УДК 004.75+004.89 Научная статья
https://doi.org/10.32603/2071-8985-2022-15-5/6-22-31

Акторная модель построения нейро-нечетких систем

С. М. Морозов✉, М. С. Куприянов

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Санкт-Петербург, Россия
✉ frostsergei01@gmail.com

Аннотация. Рассматривается метод построения нейро-нечетких систем на основе модели акторов. Представлены возможные способы реализации децентрализованных и асинхронных систем искусственного интеллекта. Модель акторов, основанная на взаимодействии отдельных, независимо функционирующих узлов, позволяет создавать гибридные интеллектуальные вычислительные системы. Свойства акторов и особенности их взаимодействия дают широкие возможности для обеспечения самодиагностики состояния системы и ее самоорганизации. Представлен сравнительный анализ эффективности работы децентрализованной нейро-нечеткой системы, построенной на основе модели акторов, и монолитной системы, построенной как единый вычислительный узел. Для этого сравнивалось время выполнения обработки определенного количества наборов входных данных на обоих видах систем. Также были оценены распределение памяти и структура взаимодействия вычислительных узлов при построении децентрализованных гибридных систем искусственного интеллекта.

Ключевые слова: акторные системы, нейро-нечеткие системы, гибридные системы, искусственный интеллект, асинхронные вычисления

Для цитирования: Морозов С. М., Куприянов М. С. Акторная модель построения нейро-нечетких систем // Изв. СПбГЭТУ «ЛЭТИ». 2022. Т. 15, № 5/6. С. 22–31. doi: 10.32603/2071-8985-2022-15-5/6-22-31.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Actor model of building neuro-fuzzy systems

S. M. Morozov✉, M. S. Kupriyanov

Saint Petersburg Electrotechnical University, Saint Petersburg, Russia
✉ frostsergei01@gmail.com

Abstract. Development of actors-based model of neuro-fuzzy systems is considered. Different ways of distributed asynchronous artificial intelligent systems. Actor model, which is based on an interaction of separate and independent units, allows creating of hybrid intelligent calculating systems. Properties of actors and the specifics of their communication give a number of ways for providing system's self-diagnostics and self-organization. The comparison of efficiency of distributed neuro-fuzzy system, based on actors model, and a monolith system, created as one unit, is provided. Time of processing a specific number of datasets was performed on both if these types of systems for this purpose. Memory distribution and calculating units' interaction of distributed hybrid artificial intelligence systems were estimated.

Keywords: actor systems, neuro-fuzzy systems, hybrid systems, artificial intelligence, asynchronous computing

For citation: Morozov S. M., Kupriyanov M. S. Actor model of building neuro-fuzzy systems // LETI Transactions on Electrical Engineering & Computer Science. 2022. Vol. 15, no. 5/6. P. 22–31. doi: 10.32603/2071-8985-2022-15-5/6-22-31.

Conflict of interest. The authors declare no conflicts of interest.

Введение. Современные вычислительные системы предлагают широкий набор технологий и методов для решения различных задач. Большое распространение получают системы, основанные на концепциях искусственного интеллекта (ИИ). Интеллектуальные методы решают широкий спектр задач, связанный с выполнением сложных вычислений: обработка изображений и видео, классификация и кластеризация объектов, и др.

Существуют различные технологии, которые используются в интеллектуальных системах. Из этих технологий можно выделить нейронные сети и нечеткую логику. Нейронные сети – основная технология для реализации систем машинного обучения. С их помощью реализуются обучаемые системы различного назначения. Нечеткая логика – математический аппарат, с помощью которого интерпретируются вычислительные системы. Это осуществляется с помощью систем нечеткого вывода.

Объединение нейронных сетей и нечеткой логики представляет собой нейро-нечеткую систему, которая обладает хорошей интерпретируемостью и обучаемостью. Классическая нейронечеткая система – система нечеткого вывода, которая использует нейронную сеть для реализации нечетких правил.

Также широко распространены технологии, связанные с построением распределенных и асинхронных систем. Эти технологии позволяют оптимизировать вычислительный процесс для повышения эффективности систем. Благодаря применению параллельных вычислений можно оптимизировать асинхронные операции для повышения эффективности систем.

Один из способов построения асинхронных вычислительных систем – реализация мультиагентных или акторных систем. Основная концепция этих систем заключается в их децентрализованности. Мультиагентная и акторная системы представляют собой ряд подпрограмм (агентов и акторов соответственно), которые получают определенную информацию и взаимодействуют друг с другом. В этом взаимодействии заключается основное отличие этих систем от монолитных, в которых есть только один основной вычислительный узел.

В научном мире ведутся работы, связанные с построением децентрализованных интеллектуальных систем. Существуют модели нечетких мультиагентных систем [1], [2] и нейронных сетей [3], [4]. Также создаются интеллектуальные системы на основе акторов [5]–[8]. Так как нейронечеткая система представляет собой объединение интеллектуальных систем различного вида, то логичным развитием концепции гибридного искусственного интеллекта являются мультигентные и акторные нейро-нееткие системы. Целью служит разработка обобщенной модели нейронечетких систем на основе акторов и проверка ее эффективности.

Преимущества акторных нейро-нечетких систем. Основные преимущества акторных нейронечетких систем – это их асинхронность и децентрализованность. Гибридные интеллектуальные системы задействуют несколько отдельных вычислительных элементов, поэтому их можно распределить на разные вычислительные узлы. Таким образом повышается независимость технической реализации отдельных компонентов друг от друга: если в одном из них произойдет технический сбой, то это не приведет к сбою всей системы. Как правило, акторные системы действуют в рамках одного физического вычислителя, однако ряд систем (например, Akka) предоставляют возможность построения и развертывания систем на нескольких вычислителях в одной сети.

Асинхронность позволяет построить более эффективные модели взаимодействия компонентов разрабатываемых систем. В асинхронных системах компоненты работают независимо друг от друга, что позволяет реализовать параллельные вычисления и обеспечивать независимость отдельных вычислительных узлов. Также становится возможным производить ряд вычислений в фоновом режиме. Например, в фоновом режиме можно выполнять обучение нейронной сети. Недостаток асинхронных систем заключается в более высокой сложности разработки. Для нейронечетких систем совокупность этих преимуществ существенна.

При разработке гибридных систем ИИ используются различные типы интеллектуальных вычислений. С помощью перехода на децентрализованную платформу становится возможным создавать отдельные компоненты интеллектуальных систем независимо друг от друга. Модули нейронной сети и нечеткой логики могут функционировать параллельно, реализуя потоковую вычислительную систему.

Еще одно преимущество заключается в возможности моделирования системы для других реализаций. Акторы реализуют определенный функционал, который может быть выполнен как

отдельная программа или устройство, моделируется функционал отдельных компонентов. При необходимости аппаратной или микросервисной реализации интеллектуальной системы можно взять за основу разработанную модель для определения взаимодействия узлов.

Акторная реализация повышает устойчивость системы к сбоям. Сбой одного актора не приводит к сбою всей системы, что обеспечивает стабильность работы программы. При этом программа может диагностировать всю систему, отслеживая состояния отдельных акторов, контролировать наличие сбоев в отдельных узлах и восстанавливать поврежденную функциональность, а также при необходимости легко заменять компоненты. Этот функционал акторной системы можно рассматривать как базу построения самоорганизующихся нейро-нечетких систем.

Следует отметить, что децентрализованная реализация иногда приводит к потере производительности. Это связано с необходимостью взаимодействия между акторами и межпоточному взаимодействию. Прирост производительности для децентрализованной и асинхронной архитектур может быть получен при выполнении параллельных вычислений на большом количестве параллельно выполняемых потоков. Нейро-нечеткие системы, как правило, последовательные, поэтому при разовом использовании параллельных вычислений нет.

Модель акторной нейро-нечеткой системы. Реализация акторной модели предлагает разбиение нейро-нечеткой системы на отдельные составляющие: фаззификация входных данных, обработка полученных значений нейронной сетью и дефаззификация полученных результатов. Для создания этих компонентов нужно решить ряд вспомогательных задач – вычисление функций активации при фаззификации, обучение нейронной сети. Этот функционал возможен в виде отдельных акторов.

Особенность акторных и мультиагентных систем состоит в том, что компоненты системы взаимодействуют с помощью специально определенных сообщений. По этой причине нужно определить формат сообщений, которые будут передаваться в системе.

В первую очередь необходимо определить акторы для начала и завершения процесса. Поскольку акторы взаимодействуют только посредством сообщений, должны быть созданы: актор, который запустит обработку данных, и актор, который будет обрабатывать результат.

При реализации основных составляющих нейро-нечеткой системы нужны 3 актора, которые выполняют функционал ее слоев: фаззификатор, нейронная сеть и, при необходимости, дефаззификатор. Каждый из этих слоев может быть реализован отдельным актором. Так как эти составляющие обрабатывают данные в виде чисел и числовых массивов, сообщения должны передавать их. Могут быть использованы и другие акторы в зависимости от поставленной задачи – например, отдельным актором можно реализовать задачу подготовки данных для обучения нейронной сети.

Для диагностики системы иногда нужен дополнительный актор, который осуществляет мониторинг состояния вычислительных узлов. Обнаружив сбой, этот узел восстанавливает поврежденный функционал.

Примеры реализации моделей. Для тестирования представленной модели разработаны несколько систем с использованием акторного фреймворка. Система выполняет задачу классификации на основе двух параметров. Для программной реализации использовался фреймворк акторов Actix. Выбор обусловлен скоростью работы: на базе Actix построен веб-сервер Actix Web – один из самых быстрых серверов. Другая причина выбора этого фреймворка заключается в том, что он разработан для языка программирования Rust, который дает возможность напрямую работать с памятью, при этом не позволяя строить небезопасные системы. Это позволит обеспечить потокобезопасность системы и оценить распределение памяти в ней.

Перед разработкой акторной нейро-нечеткой системы нужно определить общий принцип функционирования нейро-нечетких систем для последующего определения принципов отправки сообщений в акторах. Первым шагом является фаззификация, т. е. преобразование четких входных данных в нечеткую форму, реализованная как вычисление заранее определенных функций активации нечетких множеств. Результаты фаззификации для всех переменных объединяются в один массив и подаются на вход предварительно обученной нейронной сети. Результат работы нейронной сети при корректно выполненном обучении представляет собой массив нечетких значений, которые можно дефаззифицировать, т. е. перевести в четкую форму из нечеткой. В ряде

задач достаточно получить нечеткие значения в качестве результата. На рис. 1 представлена общая структура построения нейро-нечеткой системы. Эта структура представляет собой общий алгоритм выполнения нейро-нечетких вычислений в системе. Простейший алгоритм задействует один вычислительный поток.

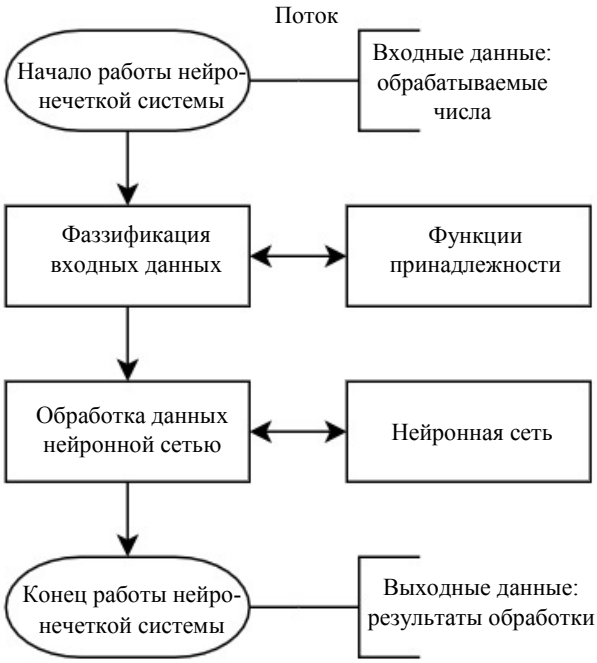


Рис. 1. Структура работы нейро-нечеткой системы
Fig. 1. Structure of neuro-fuzzy systems

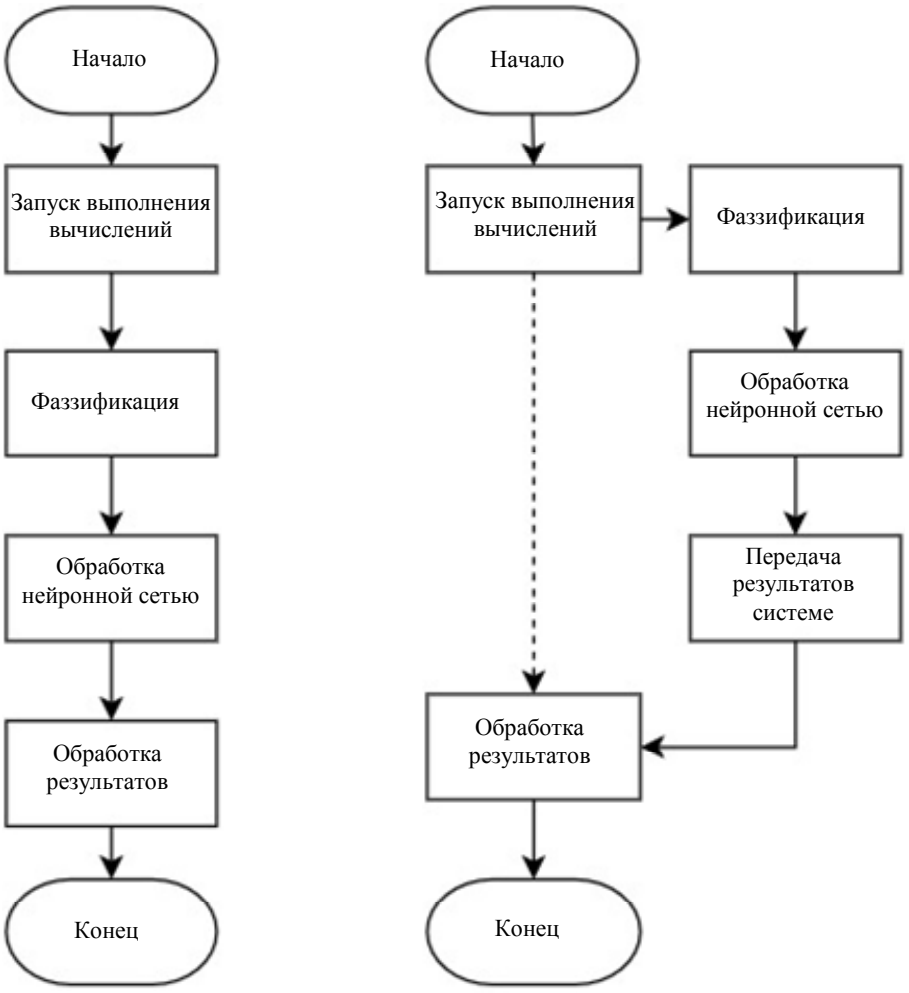
В зависимости от задачи, решаемой нейро-нечеткой системой, может быть реализован один из двух подходов к получению результатов вычислений: прямого выполнения и режима ожидания. При прямом выполнении каждый актор после завершения своей части вычислений передает сообщение с необходимыми для вычисления данными следующему актору в цепочке вызовов. Таким образом, в последнем акторе оказывается собран результат всей последовательности вычислений. Этот же актор выполняет действия, необходимые после вычислений. Такой подход хорошо подходит для выполнения фоновых вычислений, результат которых не нужно ожидать для корректного выполнения программы. Этот подход также можно назвать синхронным. Он ближе по своей структуре к монолитной реализации нейро-нечетких систем.

При выполнении вычислений в режиме ожидания данные после получения результатов передаются в отложенную функцию (callback function) актора, который инициировал начало вычислений. Таким образом, результаты обрабатывает начальный актор. Такая реализация удобна, если нужно обеспечить очередность в системе при отправке сообщений. Этот подход можно назвать асинхронным, поскольку актор, инициировавший запуск вычислений, ожидает завершения ее работы для получения результата. Такой подход удобен для построения систем управления, так как позволяет при необходимости модифицировать имеющиеся данные системы, после чего повторить выполнение вычислений, создавая таким образом обратную связь.

Структуры акторных систем прямого выполнения и режима ожидания представлены на рис. 2. Акторы выполняют фаззификацию входных данных и обработку нечетких значений нейронной сетью. Основной критерий выбора механизма построения – это необходимость ожидания результатов или ее отсутствие.

Акторные фреймворки широко используются в промышленных программных системах для организации асинхронных систем. Вычислительные узлы в этих системах работают независимо друг от друга, что позволяет легко реализовывать параллельные вычисления, а также обработку данных и запросов на их основе. Для этого создаются средства управления контекстом создаваемой акторной системы. В выбранном фреймворке Actix для осуществления параллельных вычислений используется арбитр системы, создающий для акторов асинхронный контекст выполнения, в котором осуществляются все требуемые операции.

На рис. 3 представлена структура реализации нейро-нечеткой системы в асинхронном контексте. В этой системе актор, который запустил цепочку вычислений, ожидает результатов обработки данных. Обработку результатов можно перенести в завершающий работу актор, поэтому передача данных начальному актору необязательна (обозначена штриховой стрелкой). Через этот актор осуществляется взаимодействие с остальными акторами, поэтому через него можно выполнять и другие функции. Например, ему можно отправить команду для завершения работы действующей акторной системы, которая перенаправляется нужному вычислителю (в представленном примере этот же актор получает результат работы нейронной сети и обеспечивает его отправку актору, который ожидает результат работы системы). Также следует отметить, что можно запустить акторную систему как на основном потоке про-



а *б*

Рис. 2. Структура вычислений в авторах:
а – прямого порядка действий; *б* – режима ожидания
Fig. 2. Structure of actors calculations: *а* – direct mode; *б* – awaiting mode

граммы, так и на отдельном (фоновом). На рисунке представлена модель, в которой акторы запускаются на отдельном потоке, так как это решение освобождает основной поток для потенциального взаимодействия с пользовательским интерфейсом. В этом случае у пользователя системы появляется возможность взаимодействовать с акторами через отправку определенных сообщений (например, сигнал завершения работы системы).

Actix позволяет создавать не только асинхронный контекст выполнения работы, но и синхронный. Для этого существует синхронный арбитр, который создает синхронный контекст для акторов. С его помощью параллельно запускаются несколько одинаковых вычислителей, доступ к которым будет осуществляться по одному общему адресу. Таким образом, создается балансировка нагрузки при обмене сообщениями. Этот инструмент подходит для выполнения параллельных запросов, так как задействует компонент системы, отвечающий за передачу сообщений одному из параллельно работающих акторов. Синхронный контекст обеспечивает устойчивость отдельных акторов, так как восстанавливает поврежденные узлы системы при сбое. Но такой подход требует очень много памяти, поскольку создается несколько экземпляров одного актора, которые должны хранить экземпляры фаззификаторов и нейронной сети для обеспечения потокобезопасности выполнения вычислений. Структура реализации нейро-нечеткой системы в синхронном контексте представлена на рис. 4. Синхронизатор акторов выполняет функции параллельной передачи запросов в системе. При этом можно создать различное количество акторов фаззификации и обработки нейронной сети для оптимизации памяти системы.

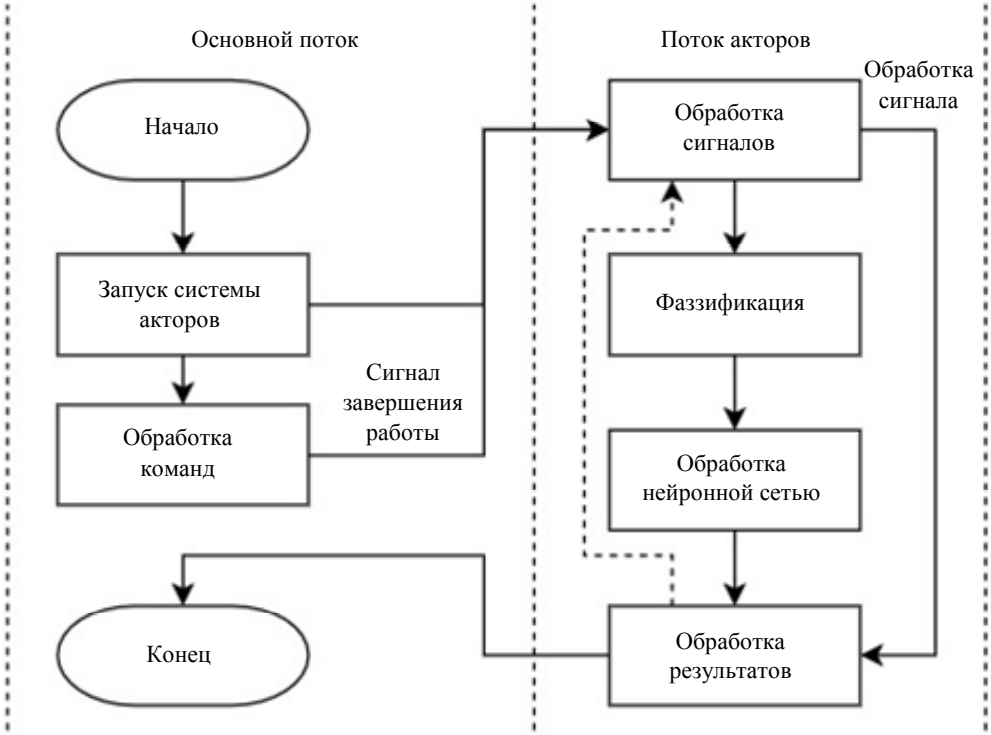


Рис. 3. Реализация нейро-нечеткой системы в асинхронном контексте акторов
Fig. 3. Asynchronous context implementation of neuro-fuzzy actors system

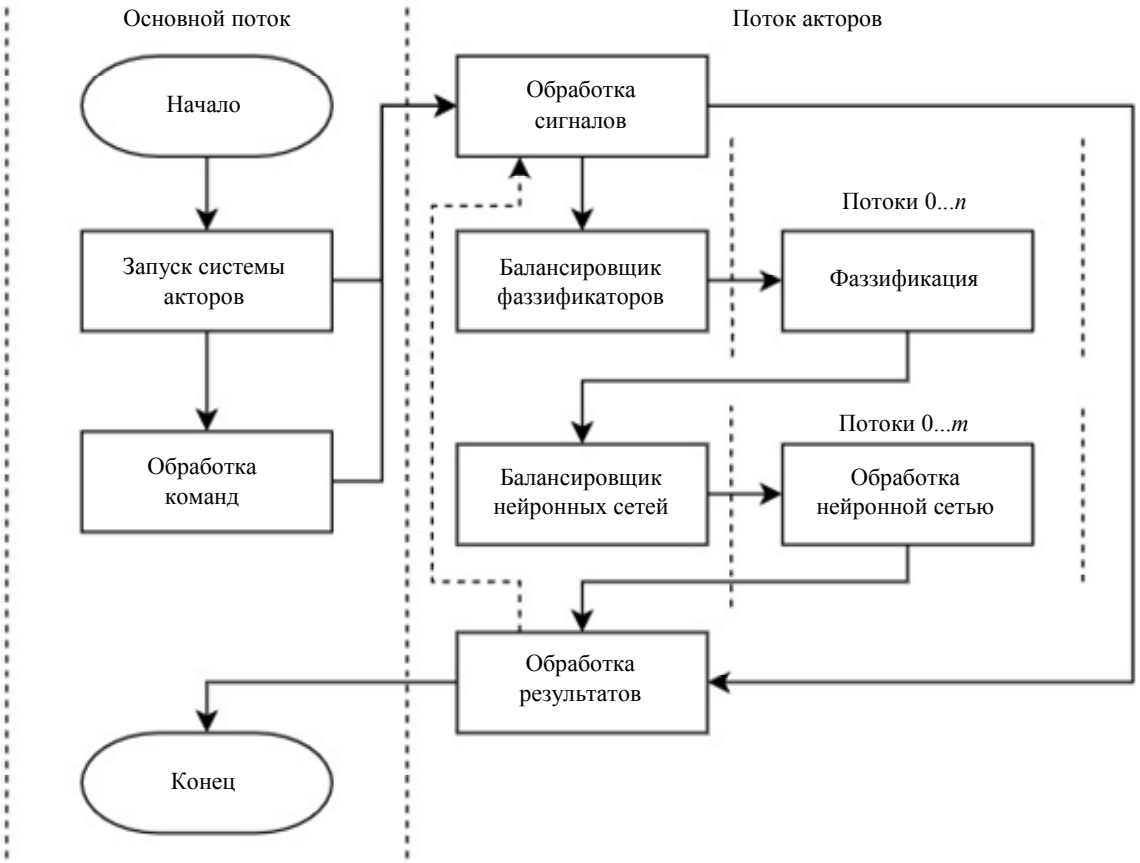


Рис. 4. Реализация нейро-нечеткой системы в синхронном контексте акторов
Fig. 4. Synchronous context implementation of neuro-fuzzy actors system

Представленные инструменты и средства позволяют построить несколько вариаций акторной модели нейро-нечеткой системы в зависимости от решаемой задачи. Все представленные системы основаны на передаче сообщений между акторами, что обеспечивает масштабируемость системы. Однако отправка, прием и передача сообщений могут сказываться на общем времени выполнения вычислений, так как они занимают процессорное время.

Важно правильно оценить время работы всех видов систем. Сначала нужно оценить влияние акторной системы на общее время вычислений, для чего измеряется время работы трех моделей нейро-нечеткого вычислителя. Сравниваются следующие виды систем: на синхронном контексте, на асинхронном контексте и без применения акторов. Актор выполнял нейро-нечеткую обработку входных данных. Система акторов строилась по модели режима ожидания как более универсальная. Значения системы обрабатываются для всех входных данных. В системе без акторов нейро-нечеткие вычисления осуществляются непосредственно в цикле, в акторной системе для начала обработки отправляется сообщение. Также было измерено время работы системы, которая запускала цикл нейро-нечеткой обработки непосредственно в акторе, однако времена работы такой системы и системы без акторов были равны, так как отправлялось всего одно сообщение. Для корректного сравнения времени вычислений задействуется один поток. Результаты сравнения представлены в табл. 1. Из общего времени выполнения вычиталось время обучения нейронной сети для того, чтобы получить результаты обработки входных данных. Временем построения системы фаззификации можно пренебречь.

Из результатов измерений (табл. 1) можно сделать вывод, что применение выбранного фреймворка акторов приводит к незначительному повышению времени работы системы. Исходя из этого, можно принять, что время передачи сообщений акторов в системе крайне мало.

Табл. 1. Результаты сравнения работы моделей
Tab. 1. Results of models' benchmarking

Размерность входных данных	Среднее время выполнения системой без акторов, мс	Среднее время выполнения в асинхронном контексте, мс	Среднее время выполнения в синхронном контексте, мс
100 × 100	97	101	107
200 × 200	375	380	402
400 × 400	1648	1672	1701
800 × 800	6343	6427	6475
1600 × 1600	27 873	28 043	28 364

Табл. 2. Результаты сравнения работы децентрализованных нейро-нечетких систем на асинхронном контексте
Tab. 2. Results of distributed neuro-fuzzy systems' benchmarking for asynchronous context

Размерность входных данных	1 поток, мс	2 потока, мс	4 потока, мс	8 потоков, мс	16 потоков, мс
100 × 100	1149	830	527	450	408
200 × 200	4463	3854	2364	1753	1583
400 × 400	17 934	13 452	8351	6520	6027
800 × 800	70 952	51 844	32 157	25 952	23 302
1600 × 1600	352 571	209 053	138 934	118 453	106 454

Табл. 3. Результаты сравнения работы децентрализованных нейро-нечетких систем на синхронном контексте
Tab. 3. Results of distributed neuro-fuzzy systems' benchmarking for synchronous context

Размерность входных данных	1 поток, мс	2 потока, мс	4 потока, мс	8 потоков, мс	16 потоков, мс
100 × 100	3962	2578	1465	1448	1338
200 × 200	16 468	9361	5254	4939	4637
400 × 400	87 636	55 386	22 462	16 384	13 193
800 × 800	325 153	175 751	85 044	64 936	49 237
1600 × 1600	1 403 562	613 807	356 481	263 932	196 832

На основе акторной модели были построены две децентрализованные версии нейро-нечеткой системы: на синхронном и асинхронном контекстах. Новая реализация отличается тем, что для фаззификатора и для нейроной сети были выделены отдельные акторы. Это увеличит число сообщений в системе и объем передаваемых данных, что приведет к повышению времени работы. Было оценено затраченное на выполнение время, а также влияние изменения количества потоков в системе. Результаты измерений представлены в табл. 2 и 3.

Из этих таблиц видно, что применение акторов обеспечивает параллелизм системы, однако при большом количестве передаваемых данных может увеличиться время работы. Это объясняется увеличением работы системы, связанной с передачей сообщений. Применение многопоточности повысило совокупную скорость работы. Для синхронного контекста выполнения использование нескольких потоков оказало более сильный эффект, так как для отдельной задачи создавалось несколько акторов, т. е. обеспечивалась балансировка нагрузки.

Применение синхронного контекста также привело к увеличению времени обучения отдельной нейронной сети, однако обучение происходило параллельно. При использовании одного потока нейронная сеть обучалась около 300 мс, в то время как при большом количестве потоков обучение каждой сети длилось около 1.5 с. Но это обучение происходило параллельно и привело к тому, что менее чем за 2 с были обучены более 8 нейронных сетей, т. е. быстрее, чем если бы обучение происходило последовательно. Обучить одну нейронную сеть для передачи ее адреса в акторы не представлялось возможным, так как это нарушало бы потокобезопасность системы. Выбранные средства разработки (в частности, язык программирования Rust) не позволили бы создать систему, не обеспечивающую потокобезопасность компонентов.

Увеличение общего времени выполнения программы связано с увеличением последовательности передачи данных в системе акторов. В текущей версии после отправки сообщения цепь вычислений задействует три актора, в то время как в первой системе – один. Это приводит к росту совокупного количества сообщений в системе, что увеличило время работы. Для синхронного контекста выполнения эта проблема более серьезна, так как при этом затрачивается время на балансировку нагрузки акторов и получение данных из них. В то же время, вычислительные узлы в синхронном контексте более устойчивы к техническим сбоям – при остановке работы одного актора будет запущена его копия.

Особенность построенной модели заключается в заменяемости узлов в последовательности выполнений. При применении акторов каждый следующий этап вычислений задается с помощью адреса нужного актора. Для замены узла нужно создать новый вид акторов и подключить их к вычислительному процессу.

Заключение. В результате выполнения работы была получена модель акторной нейро-нечеткой системы. Получены синхронная и асинхронная вариации построения этой модели в зависимости от решаемых задач. Обе эти модели можно применять для решения различных задач: синхронная модель лучше подходит для настройки внутренних параметров системы с помощью нейро-нечеткой системы, в то время как асинхронная может быть использована в качестве инструмента построения систем с обратной связью.

Также были рассмотрены инструменты обеспечения параллельности нейро-нечетких вычислений. В различных акторных фреймворках существуют инструменты для запуска нескольких акторов на параллельных потоках и для обеспечения их отказоустойчивости от разовых сбоев. Эти средства могут быть использованы для выполнения самодиагностики системы и обеспечения самоорганизации на их основе.

Особенностью акторной системы является возможность реализовать отдельный функционал гибридных систем искусственного интеллекта на различных вычислительных узлах. Асинхронность и децентрализованность рассмотренных систем может быть использована для моделирования и быстрого построения прототипов гибридного искусственного интеллекта в условиях, когда предполагается разработка системы в виде отдельных сервисов. Таким образом можно определить функционал отдельных компонентов системы для реализации.

Список литературы

1. Enhancement of neural network based multi agent system for classification and regression in energy system / C. T. Yaw, K. S. Yap, S. Y. Wong, H. J. Yap, J. K. S. Paw // IEEE Access. 2020. Vol. 8. P. 163026–163043. doi: 10.1109/ACCESS.2020.3012983.

2. A neural network-based multi-agent classifier system / A. Quteishat, C. P. Lim, J. Tweedale, L. C. Jain // Neurocomputing. 2009. Vol. 72, no. 7–9, P. 1639–1647. doi: 10.1016/j.neucom.2008.08.012.

3. Balachandran B. M., Mohammadian M. Development of a fuzzy-based multi-agent system for E-commerce settings // Procedia Comp. Sci. 2015. Vol. 60. P. 593–602. doi: 10.1016/j.procs.2015.08.186.

4. logic decision-making in multi-agent systems for smart grids / R. B. Menon, S. B. Menon, D. Srinivasan, J. L. Fuzzy // Conf. Computational Intelligence Appl. In Smart Grid (CIASG). IEEE. 2013. P. 44–50. doi: 10.1109/CIASG.2013.6611497.

5. Actor neural networks for the robust control of partially measured nonlinear systems showcased for image propagation through diffuse media / B. Rahmani, D. Loterie, E. Kakkava, N. Borhani, U. Teğin, D. Psaltis, Ch. Moser // Nature Machin Intelligence. 2020. Vol. 2. P. 403–410. doi: 10.1038/s42256-020-0199-9.

6. An integrated critic-actor neural network for reinforcement learning with application of DERs control in grid frequency regulation / J. Sun, Z. Zhu, H. Li, Y. Chai, G. Qi, H. Wang, Y. Hen Hu // Intern. J. of Electrical Power & Energy Systems, 2019. Vol. 111. P. 286–299. doi: 10.1016/j.ijepes.2019.04.011.

7. Mothku S. K., Rout R. R. Fuzzy logic based adaptive duty cycling for sustainability in energy harvesting sensor actor networks // J. of King Saud University – Comp. and Information Sci. 2022. Vol. 34, no. 1. P. 1489–1497. doi: 10.1016/j.jksuci.2018.09.023.

8. Kulla E., Elmazi D., Barolli L. Neuro-adaptive learning fuzzy-based system for actor selection in wireless sensor and actor networks // 2016 10th Intern. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS). IEEE. 2016. P. 488–493. doi: 10.1109/CISIS.2016.120.

Информация об авторах

Морозов Сергей Михайлович – аспирант кафедры вычислительной техники СПбГЭТУ «ЛЭТИ».
E-mail: frostsergei01@gmail.com

Куприянов Михаил Степанович – д-р техн. наук, профессор, заведующий кафедрой вычислительной техники СПбГЭТУ «ЛЭТИ», почетный работник высшего профессионального образования РФ.
E-mail: mikhail.kupriyanov@gmail.com
https://orcid.org/0000-0003-4695-4507

References

1. Yaw C. T., Yap K. S., Wong S. Y., Yap H. J., Paw J. K. S. Enhancement of Neural Network Based Multi Agent System for Classification and Regression in Energy System // IEEE Access. 2020. Vol. 8. P. 163026–163043. doi: 10.1109/ACCESS.2020.3012983.

2. Quteishat A., Lim C. P., Tweedale J., Jain L. C. A Neural Network-Based Multi-Agent Classifier System // Neurocomputing. 2009. Vol. 72, no. 7–9. P. 1639–1647. doi: 10.1016/j.neucom.2008.08.012.

3. Balachandran B. M., Mohammadian M. Development of a Fuzzy-based Multi-agent System for E-commerce Settings // Procedia Computer Science. 2015. Vol. 60. P. 593–602. doi: 10.1016/j.procs.2015.08.186.

4. Menon R. B., Menon S. B., Srinivasan D., Jain L. Fuzzy Logic Decision-Making in Multi-Agent Systems for Smart Grids // Conf. Computational Intelligence Appl. In Smart Grid (CIASG). IEEE. 2013. P. 44–50. doi: 10.1109/CIASG.2013.6611497.

5. Rahmani B., Loterie D., Kakkava E., Borhani N., Teğin U., Psaltis D., Moser Ch. Actor Neural Networks for the Robust Control of Partially Measured Nonlinear Systems Showcased for Image Propagation through Diffuse Media // Nature Machin Intelligence. 2020. Vol. 2. P. 403–410. doi: 10.1038/s42256-020-0199-9.

6. Sun J., Zhu Z., Li H., Chai Yi, Qi G., Wang H., Hu Yu H. An Integrated Critic-Actor Neural Network for Reinforcement Learning with Application of DERs Control in Grid Frequency Regulation // Intern. J. of Electrical Power & Energy Systems. 2019. Vol. 111. P. 286–299. doi: 10.1016/j.ijepes.2019.04.011.

7. Mothku S. K., Rout R. R. Fuzzy Logic Based Adaptive Duty Cycling for Sustainability in Energy Harvesting Sensor Actor Networks // J. of King Saud University – Comp. and Information Sciences. 2022. Vol. 34, no. 1. P. 1489–1497. doi: 10.1016/j.jksuci.2018.09.023.

8. Kulla E., Elmazi D., Barolli L. Neuro-Adaptive Learning Fuzzy-Based System for Actor Selection in Wireless Sensor and Actor Networks // 2016 10th Intern. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS). IEEE. 2016. P. 488–493. doi: 10.1109/CISIS.2016.120.

Information about the authors

Sergey M. Morozov – postgraduate student, Department of Computer Engineering, Saint Petersburg Electrotechnical University.

E-mail: frostsergei01@gmail.com

Mikhail S. Kupriyanov – Dr Sci. (Eng.), Professor, Head of the Department of Computer Engineering, Saint Petersburg Electrotechnical University.
E-mail: mikhail.kupriyanov@gmail.com
https://orcid.org/0000-0003-4695-4507

Статья поступила в редакцию 16.04.2022; принята к публикации после рецензирования 21.04.2022; опубликована онлайн 30.06.2022.

Submitted 16.04.2022; accepted 21.04.2022; published online 30.06.2022.