ИМИТАЦИОННЫЕ МОДЕЛИ КАК ВИРТУАЛЬНАЯ СРЕДА ДЛЯ ОБУЧЕНИЯ И ТЕСТИРОВАНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ БИЗНЕС-ПРИЛОЖЕНИЙ

Андрей Борщев, Arash Mahdavi, Анатолий Жеребцов
(The AnyLogic Company, https://www.anylogic.com/)

Почему искусственный интеллект и имитационное моделирование?

В эти дни на конференциях практически любого профиля выступающие опасаются НЕ говорить про искусственный интеллект, чтобы не показаться несовременными. Это часто выглядит комично, но факт остаётся фактом: ИИ – на максимальном за свою историю хайпе, туда идут огромные инвестиции, создаются стартапы с сотнями PhD, и даже самые дремучие диктаторы пытаются что-то мямлить про важность ИИ для получения контроля над миром.

Всё это происходит, естественно, не на пустом месте; успехи ИИ очевидны. ИИ существенно продвинулся и практически применяется в распознавании образов, распознавании естественного языка, финансах, маркетинге и так далее. ИИ умеет играть в компьютерные и настольные игры. AplhaGo не только выиграл у человека в го, но и создал абсолютно новые, негуманоидные стратегии игры, которые человек теперь изучает. И сейчас разработчики ИИ пытаются совершить экспансию, в том числе, в область бизнес-задач: управление производством, логистику, цепочки поставок, бизнес-процессы, управление активами и так далее.

Далее в этой работе мы сузим область и будем говорить об ИИ, создаваемом с целью принятия решений по управлению каким-то объектом в изменяющихся условиях для достижения определённых целей, то есть о “политике”, “стратегии”, “поведении”. Здесь успехи ИИ пока скромные. Согласно Wired [https://www.wired.com/story/deepminds-losses-future-artificial-intelligence/], “Реальные проблемы ограничены не так сильно, как игры, на которых был сфокусирован Deep Mind, поэтому им еще только предстоит найти настоящее масштабное коммерческое приложение глубокого обучения с подкреплением.”

Для того, чтобы ИИ мог учиться эффективному поведению в динамичной бизнес-среде, гораздо более сложной и многообразной, чем компьютерные игры, ему всё равно необходима некая виртуальная площадка, так как:

Эксперименты для обучения в реальной среде недопустимо дороги и опасны

Они также слишком долгие: могут понадобиться миллионы и миллиарды циклов обучения

Использование накопленных исторических данных вместо реального объекта возможно, но ограничивает область обучения теми сценариями, которые случились и были записаны

Имитационное моделирование естественным образом предоставляют мощную реалистичную виртуальную среду, позволяющую проводить безопасную тренировку и тестирование обучающихся агентов. Поэтому в настоящее время мы наблюдаем всплеск спроса на имитационные модели со стороны разработчиков ИИ, что, конечно, не может нас не радовать.

Терминология

Теперь – немного терминологии ИИ. Рис. 1 взят из [https://www.argility.com/argility-ecosystem-solutions/iot/machine-learning-deep-learning/].

Рис. 1. Базовые определения искусственного интеллекта

Искусственный Интеллект (*Artificial Intelligence,* AI) – свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека [Wikipedia]. Важно, что это определение никак не ограничивает тип реализации.

Машинное Обучение (*Machine Learning, ML*) – самостоятельное (без явного программирования) получение знаний интеллектуальной системой в процессе её работы.

Так называемый "Традиционный" ИИ – подмножество алгоритмов МО (сюда входят, например, Support Vector Machines, Random Forests, K Means Clustering, Naïve Bayes Classifier). Его применение ограничено, так как таким алгоритмам нужно явно указывать на значимые особенности входных данных. Такой ИИ не масштабируется, так как его обучение существенно зависит от руководства человека.

Наконец, термин *Глубокое Обучение* (*Deep Learning, DL*) относится к искусственным нейронным сетям – структурам, копирующим работу мозга. Такие структуры способны учиться без руководства человека на необработанных данных.

(*Обучающимся*) *Агентом* (*Learning Agent*) называется собственно интеллектуальная система, алгоритм, который учится тем или иным способом. (можно спутать с агентом в агентном моделировании, но в общем, при обучении и тестировании с помощью ИМ это и будет один из агентов в модели).

Далее нам нужно рассмотреть основные парадигмы машинного обучения (как обычного, так и глубокого). Их три, см. Рис. 2.

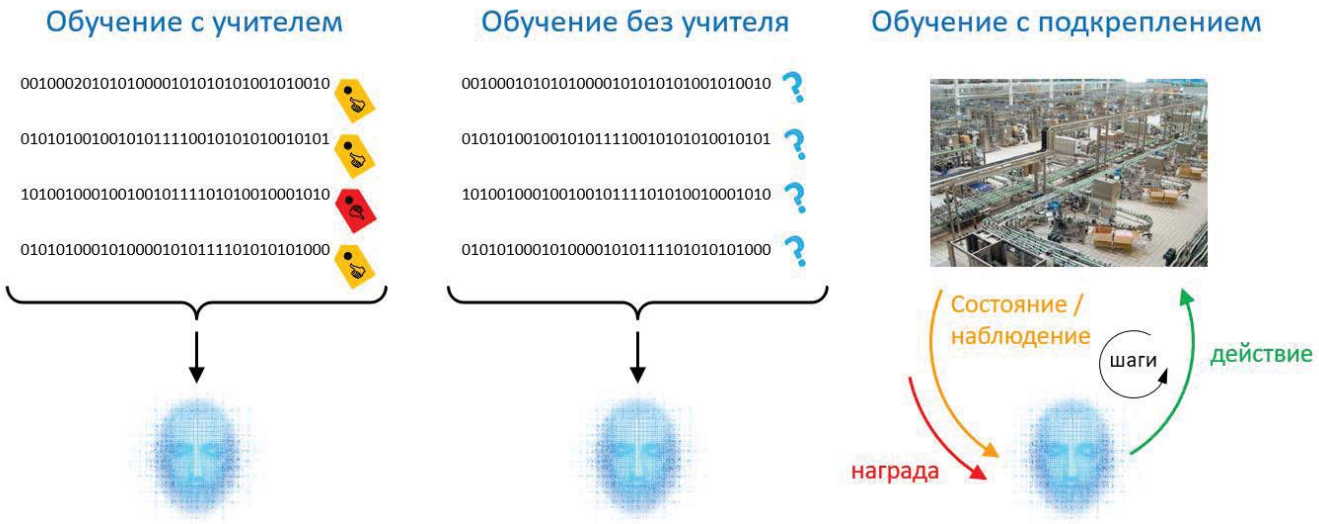

Рис. 2. Парадигмы машинного обучения (как традиционного, так и глубокого)

Обучение с учителем (*Supervised Learning*) – агенту предоставляются заранее "помеченные" данные: пары <вход, правильный ответ>, на которых он учится. Например, так можно обучить спам-фильтр, предоставляя ему заранее отсортированный на спам и не-спам поток почты.

Обучение без учителя (*Unsupervised Learning*) – агенту предоставляются необработанные ("непомеченные") данные, в которых он должен обнаружить скрытые взаимосвязи. Так можно, получить, например, кластеризацию – например, разбиение клиентов по предпочтениям.

Обучение с Подкреплением (*Reinforcement Learning, RL*) – здесь данных нет, агент учится, взаимодействуя со средой: наблюдает состояние, совершает действие, получает обратную связь – награду (которая может быть как поощрением, так и наказанием). Результатом такого обучения будет некая политика (стратегия) управления, то есть способ выбора агентом действия, исходя из наблюдения среды. После обучения политика сохраняется.

Дополнительно, по обучению с подкреплением нам будут полезны ещё несколько терминов:

Среда (*Environment*) – реальный или виртуальный мир, в котором работает обучающийся агент, его игровая площадка.

Состояние (*State*) – всё, что известно о среде в каждый момент времени

Наблюдение (*Observation*) – подмножество, или выжимка из состояния, которое видит агент.

Действие (*Action*) – то, что агент может сделать со средой, управляющее воздействие. Набор действий ограничен правилами.

Награда (*Reward*) – обратная связь к агенту, вычисляемая на основании состояния среды (или его изменения), которая используется для поощрения или наказания. Изменяет поведение агента.

Политика / *Стратегия* (Policy / *Strategy*) – способ выбора агентом действия, исходя из наблюдения среды. После обучения политика сохраняется.

Как ИМ может помочь в разработке ИИ?

Мы видим три варианта использования имитационных моделей (Рис. 3).

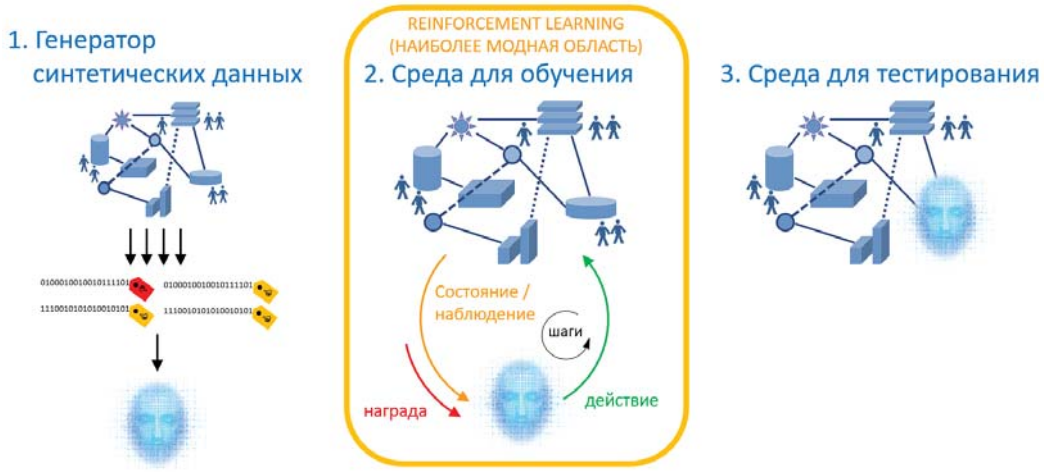

Рис 3. Как имитационное моделирование может быть полезно в разработке ИИ

Во-первых, имитационная модель может быть использована как генератор любого количества чистых, детальных, помеченных данных, покрывающих любое нужное множество вариантов развития событий. Разработчики ИИ часто страдают как от недостатка данных, так и от их зашумленности. Поскольку в ИМ мы полностью контролируем детальность и случайность, а также можем измерить и оценить происходящее, модель может быть источником данных в схеме обучения с учителем.

Во-вторых, ИМ может быть интегрирована с обучающимся агентом в схеме обучения с подкреплением. Здесь моделирование периодически прерывается для цикла обучения: агенту предоставляется некая информация о состоянии модели, он отвечает действием, меняя модель, на основе оценки этого действия ему приходит поощрение или наказание, которое меняет агента. В настоящее время это – наиболее модная область исследования, куда направлены основные усилия.

Наконец, ИИ, обученный произвольным способом (как внутри, так и вне модели), может быть включён в имитационную модель просто как компонент по принятию решений – так мы сможем протестировать ИИ до его разворачивания на реальном объекте.

Подробный пример глубокого обучения с подкреплением

Мы подробно, по шагам рассмотрим очень простой пример *глубокого обучения с подкреплением* (*Deep Reinforcement Learning*,*DRL*), в котором нейронная сеть будет учиться управлять светофором на перекрёстке. Цель – объяснить “кухню” этой технологии. Пример разработан AnyLogic совместно с компанией Skymind [https://skymind.ai/].

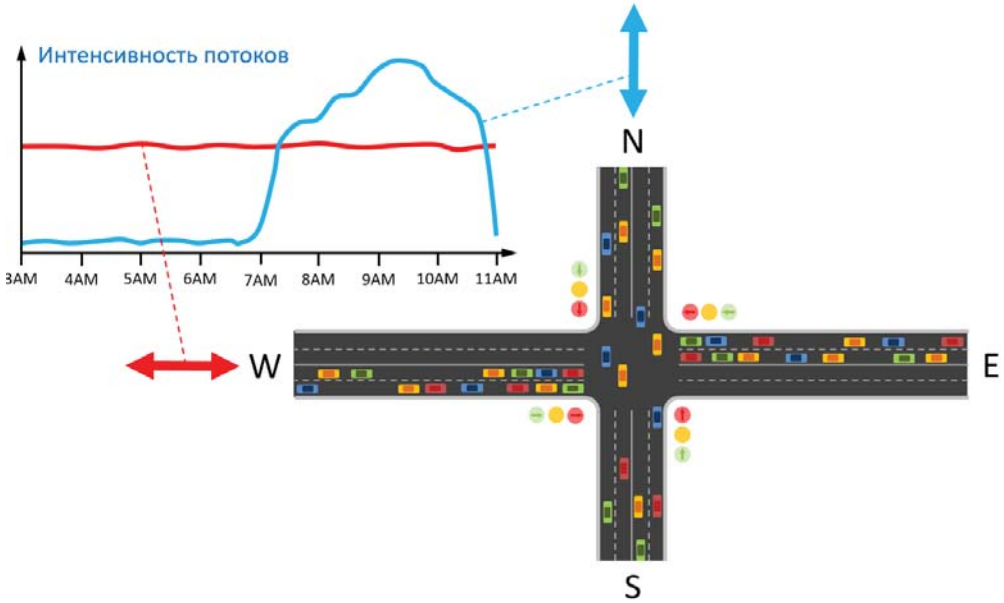

Рис. 4. Конфигурация перекрёстка в задаче управления светофором

Итак: самый простой перекрёсток, любые повороты запрещены, оборудован и всё время управляется светофором. Известны потоки машин по направлениям север-юг и запад-восток (Рис. 4), эти потоки непостоянные: направление север-юг имеет ярко выраженный час пик. Цель – минимизировать среднее время проезда перекрёстка. Доступные для варьирования параметры – длительность двух зеленых фаз светофора.

Имитационная модель была разработана в AnyLogic с помощью библиотеки Road Traffic Library (Рис. 5), она элементарна и не требует комментариев. Для нас важно, впрочем, что в этой модели мы имеем доступ ко всему и можем измерить всё: координаты каждой машины, её скорость, направление движения, ускорение, количество машин в конкретной зоне и т.д.

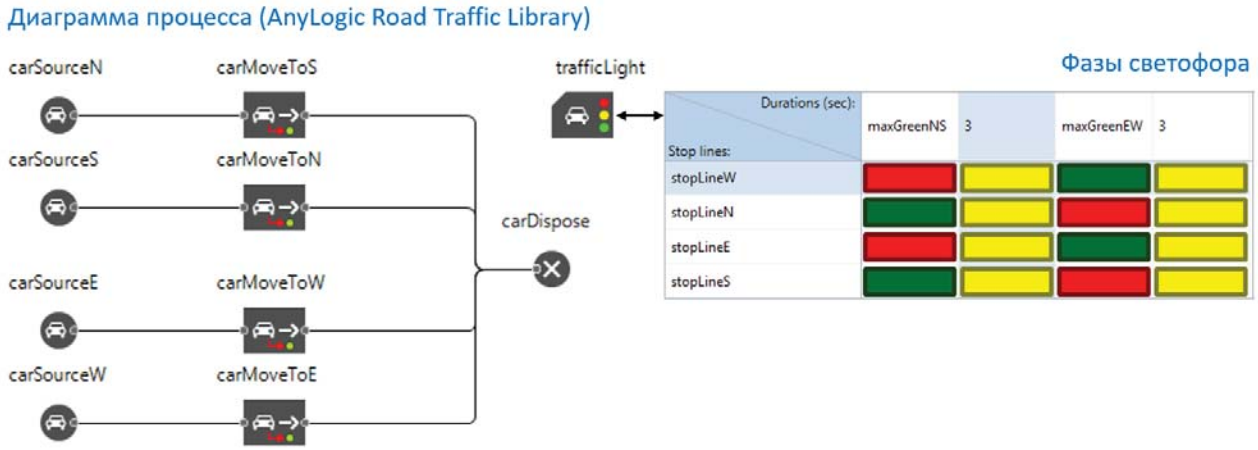

Рис. 5. Имитационная модель перекрёстка в AnyLogic

Мы собираемся поместить в эту модель обучающегося агента. Поэтому нам нужно определиться с несколькими вещами:

1) Что, какую часть состояния системы мы будем передавать агенту в качестве наблюдения?
2) Как часто мы будем это делать?
3) Как мы формализуем возможные действия агента?
4) Как будет определяться обратная связь (награда) от системы к агенту?

Были приняты следующие решения.

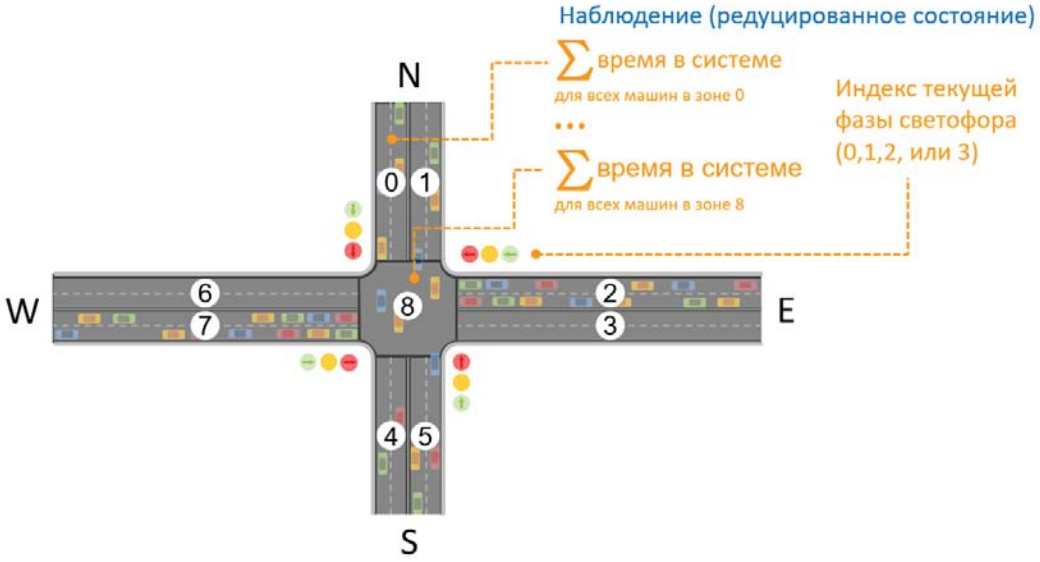

Рис. 6. Вычисления наблюдения агента на основе состояния системы

Весь перекрёсток был разбит на девять зон (Рис. 6). По каждой зоне мы вычисляем сумму времени, проведённого в системе, для всех находящихся там машин и нормализуем его в промежуток [0..1]. К этому мы добавляем значение текущей фазы светофора, их четыре, включая желтые: 0, 1, 2, 3. Эти десять значений и будут наблюдаемыми для агента. Передавать наблюдение мы будем каждые 10 моделируемых секунд.

Возможные действия агента будут такими, их два: 0 – ничего не делать, 1 – поменять фазу, то есть переключить светофор между направлениями север-юг и запад-восток.

А в качестве награды мы используем улучшение состояния системы между двумя последовательными наблюдениями, конкретно, нормализованную до [-1..1] разницу между суммами времени в системе для всех машин в зонах до и на перекрёстке (то есть, исключая машины, которые его уже проехали) в новом и предыдущем измерении.

Для самого агента выбрана простая полносвязная нейронная сеть с 10 входными нейронами (по размерности наблюдения), 2 выходными нейронами (по числу вариантов действий) и двумя скрытыми слоями по 300 нейронов в каждом (Рис. 7).

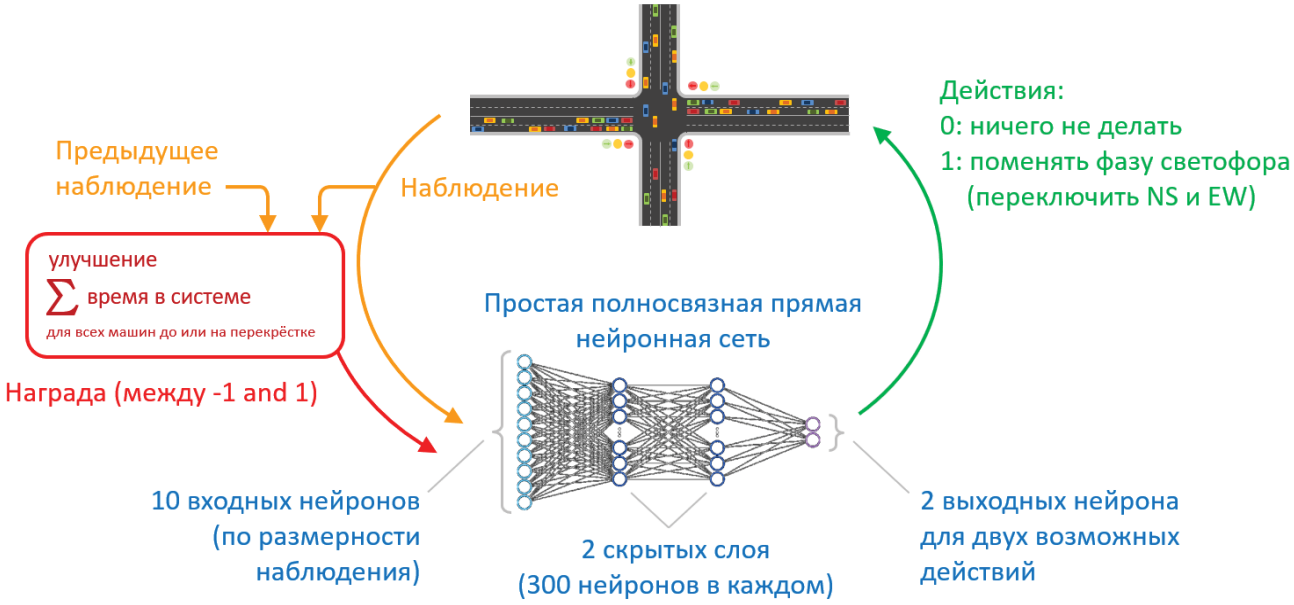

Рис. 7. Структура нейронной сети, действия и награда

Важно понимать, что все эти решения были приняты на основе интуиции и опыта, это та часть проекта, которая на настоящем этапе развития ИИ почти не формализуема и относится к области искусства, а не технологии.

Чтобы обучить нейронную сеть, мы провели 100 итераций моделирования (эпизодов в терминологии ИИ) с 2880 циклами обучения в каждой итерации; общее время обучения сети составило 30 (реальных) минут, что считается очень быстро. Множественные итерации необходимы для исследования наиболее широкого спектра сценариев. Само обучение схематически показано на (Рис. 8).

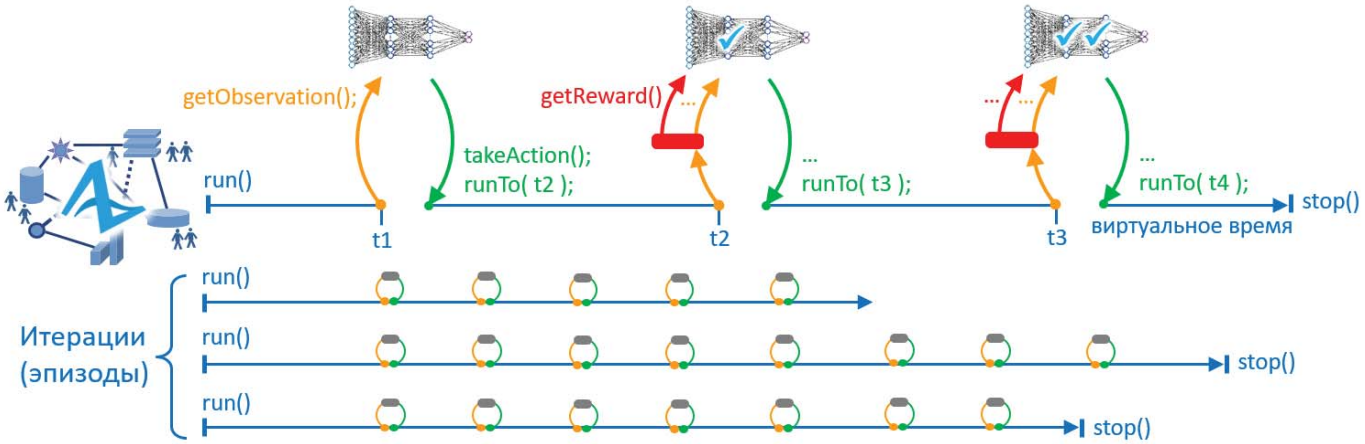

Рис. 8. Временная развёртка циклов обучения с подкреплением

Чтобы быть использованной в такой конфигурации, имитационная модель должна поддерживать простейший интерфейс:

1) **run()** – начать моделирование
2) **stop()** – завершить моделирование

3) **runTo(t)** – выполнять модель до времени **t**, затем поставить её на паузу
4) **getObservation()** – вычислить наблюдение на основе состояния модели
5) **takeAction(…)** – применить к модели управляющее воздействие агента

А также мы должны уметь считать функцию награды, для чего может понадобиться, например, частично сохранять предыдущее состояние.

Результаты нашего эксперимента следующие. Чтобы оценить качество управления светофором при помощи ИИ, мы будем сравнивать его с неким базовым сценарием, где длительности фаз светофора не меняются в течение дня и определены обычным оптимизатором. Ключевые показатели представлены на (Рис. 9). Мы видим, что нейронная сеть уже с самого начала обеспечивает лучшую пропускную способность, так как она является адаптивным алгоритмом (на графике видно, что длительность фаз всё время меняется). Во время часа пик среднее время проезда, естественно, растёт в обоих вариантах, но показатели ИИ в конце всё равно остаются лучше.

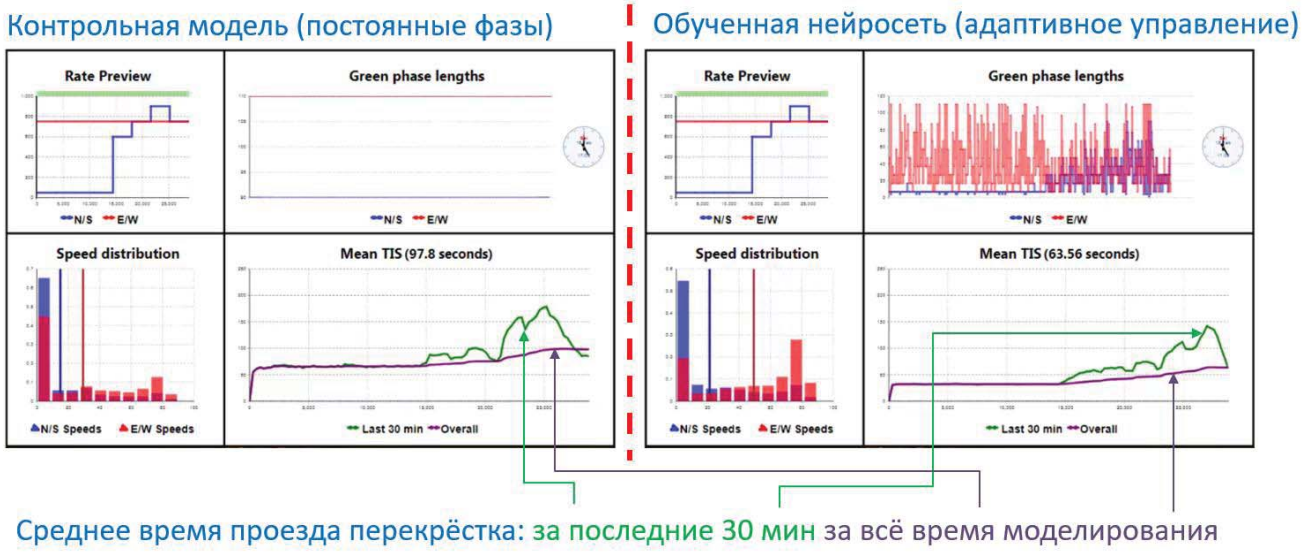

Рис. 9. Сравнение управления перекрёстком нейросетью и базового сценария

Здесь нужно сделать важное замечание. Во-первых, пример был намеренно упрощён для целей демонстрации. Во-вторых, для данной задачи несложно вручную, без всякого искусственного интеллекта сконструировать набор эвристик, которые будут управлять светофором ещё лучше. Но представьте себе систему чуть более сложную, например, десять взаимосвязанных (и поэтому взаимозависимых) перекрёстков со светофорами. На такой размерности задачи человеческий мозг начнёт давать сбои – просто по превышению количества информации, которую он может эффективно обработать, а ИИ покажет своё преимущество.

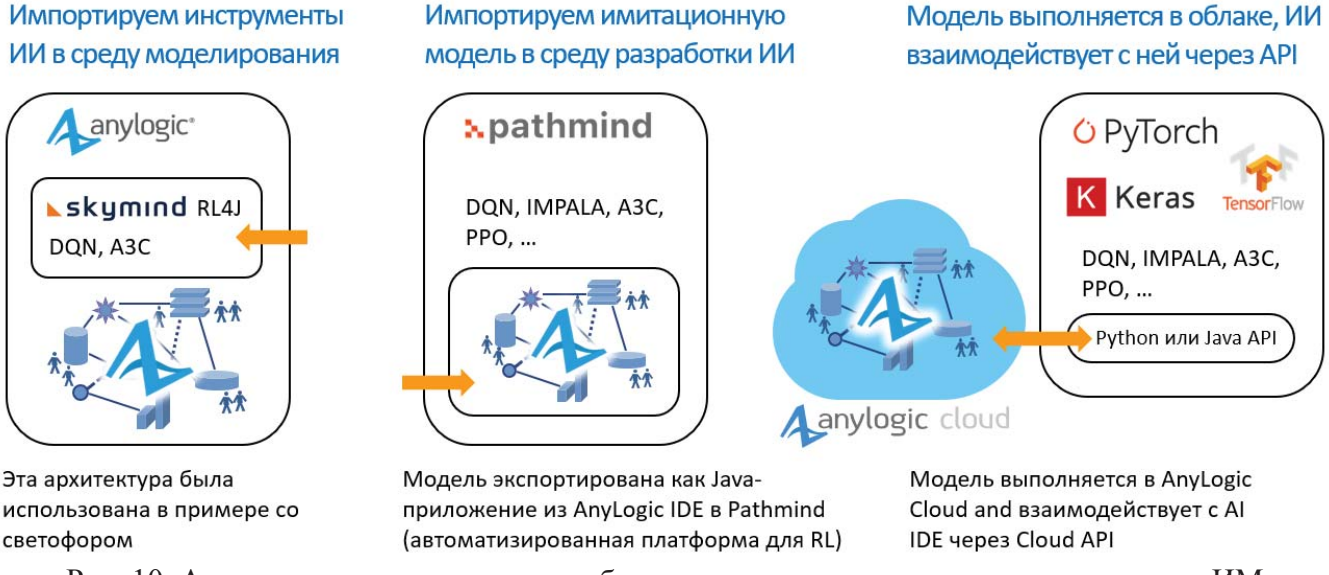

Рис. 10. Архитектурные решения для обучения с подкреплением с использованием ИМ

Мы завершим этот пример обзором архитектурных решений, которые могут использоваться для глубокого обучения с подкреплением при помощи имитационных моделей, см (Рис. 10). Для нашего простого примера мы импортировали в AnyLogic библиотеки Skymind (теперь – часть платформы Eclipse с открытым кодом), в частности RL4J [https://deeplearning4j.org/], написанные на Java, и использовали AnyLogic Custom Experiment для реализации процесса обучения. Однако, для промышленного применения, то есть для более сложных случаев мы рекомендуем делать наоборот: экспортировать имитационную модель из AnyLogic в среду разработки ИИ или в облако и использовать один из программных интерфейсов (Java, Python, JS) для взаимодействия с ней. Skymind недавно запустил платформу Pathmind [https://pathmind.com/], которая явно поддерживает имитационные модели на AnyLogic.

Пример проекта: обучение с подкреплением в управлении производством

Давайте рассмотрим пример реального использования ИИ для решения бизнес задачи. Компания Лагор (Италия) производит ферромагнитные сердечники для трансформаторов. Эти сердечники производятся на заказ и требуют различных операций на разных стадиях производственного процесса в зависимости от размера и требований клиентов. Масса сердечников может достигать 10 тонн, их перемещение по цеху между станциями обработки осуществляется на металлических паллетах посредством роликовых конвейеров и челночных рельсовых тележек.

При одновременном производстве нескольких сердечников могут возникают ситуации, когда один или несколько сердечников блокируют перемещение другого сердечника. Для продолжения производства необходимо перемещение заготовок таким образом, чтобы освободить путь для наиболее приоритетной заготовки.

Ручное планирование производства с небольшим горизонтом приводило к возникновению заторов. Зачастую наиболее эффективным способом оказывалось снятие заготовок краном и перезапуск всей линии производства.

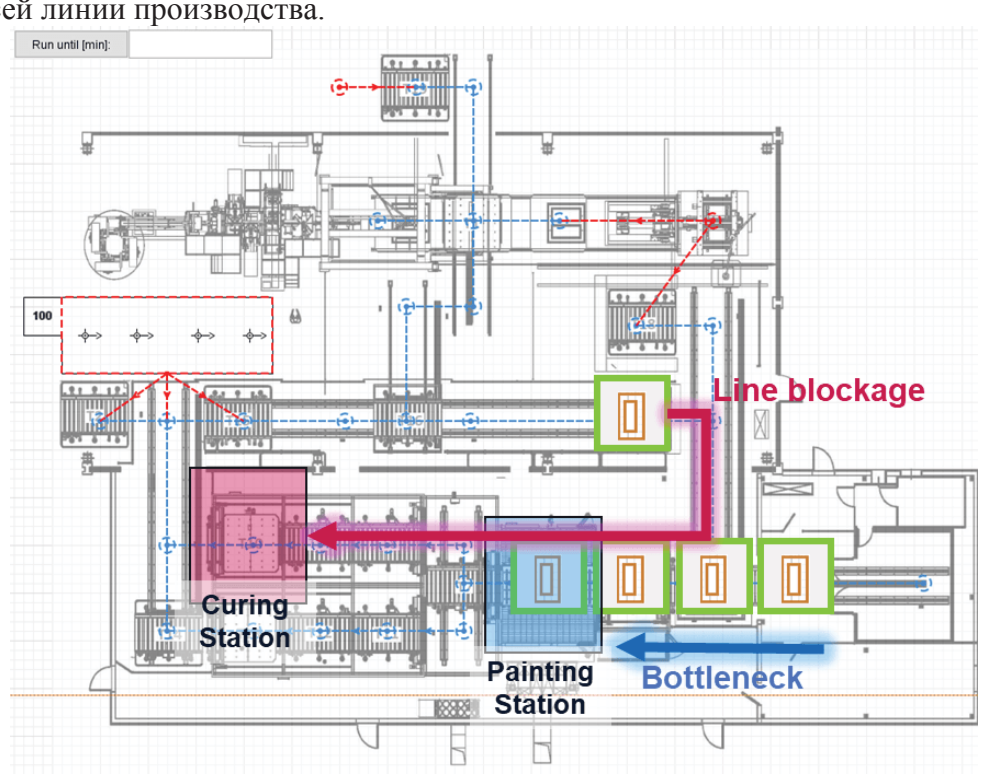

Рис. 11. Затор и узкое место на производственной линии

Для повышения эффективности производства компанией Engineering Ingegneria Informatica был разработан цифровой двойник производства, который представляет собой детальную имитационную модель, разработанную в AnyLogic, интегрированную со SCADA-системой, предающей текущее состояние производственной линии и сердечников в стадии производства. Применение эвристических алгоритмов для управления перемещением сердечников по цеху

показало существенный прирост общей эффективности производства по сравнению с ручным планированием. Однако, в сложных случаях эвристики не справлялись с поставленной задачей.

В качестве решения специалисты компании применили глубокое обучение с подкреплением агентов, управляющих перемещением сердечников на производственной линии.

Для обучения агентов использовалась имитационная модель, разработанная в AnyLogic с использованием библиотеки для глубокого машинного обучения DL4J [https://deeplearning4j.org/]. Основная задача управления процессом производства была разбита на более простые подзадачи (перемещение сердечника к конкретной целевой позиции). Далее, для каждой подзадачи производилось обучение соответствующего агента.

В качестве обучаемого алгоритма был выбран DDQN, так как он демонстрирует хорошие результаты при конечном наборе действий, а в данной задаче допускается только 64 возможных действия при любой начальной конфигурации производственной линии. [https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389/11847]

В качестве наблюдаемого состояния агент получает положение всех перемещаемых объектов на линии производства (сердечников и рельсовых тележек).

Функция награды R за выполнение действий устанавливается следующим образом:

1) $R_p = 1000$ если действие перемещает сердечник в целевую позицию;
2) $R_i = 1$ если в результате совершенного действия перемещается сердечник или рельсовая тележка помещается рядом с сердечником;
3) $R_n = -1$ во всех остальных случаях.

R_i - небольшую награду за действия, которые потенциально повышают вероятность достижения целевой позиции. При этом R_i достаточно мало, чтобы многократное перемещение сердечника между двумя соседними положениями (сумма наград R_i) не могло превысить награду от достижения целевой позиции (R_p) и привести к ложной интерпретации функции награды [https://openai.com/blog/faulty-reward-functions/].

В начале процесса обучения агент предлагает случайно выбранные действия, многие из которых неосуществимы (например, перемещение объекта в позицию, занятую другим объектом), однако в результате многократно повторяющихся циклов обучения агент под воздействием функции награды формирует эффективную политику достижения цели.

В результате, ансамбль обученных агентов успешно решает задачу перемещения сердечников по цеху и повышения эффективности производства.

Обсуждение: трудности, проблемы, вопросы

Нерешенных проблем много, но нерешаемых среди них нет, что хорошо.

Во-первых, имитация реальных бизнес-задач может быть вычислительно трудоёмкой. Время на моделирование между двумя последовательными циклами обучения может быть слишком большим с точки зрения ИИ, которому могут понадобиться миллионы и миллиарды циклов. Даже секунда в этом контексте может быть недопустимо долгой. Это осложняется тем, что в реальных задачах как пространство состояний, так и пространство действий могут быть огромными, что, естественно, замедляет сходимость процесса обучения, так как требует большего размера нейронной сети и более долгого обучения. Один из эффективных способов борьбы с этим – параллельное обучение [например, https://arxiv.org/abs/1802.01561 или https://arxiv.org/pdf/1802.01561.pdf] и легко масштабируемые облачные решения.

Во-вторых, модель, как мы все знаем, это неточное отражение реальности. То, насколько хорошо модель отражает проблему, целиком и полностью зависит от разработчика модели. Плохая модель – плохо обученный ИИ. Но моделей есть и преимущество по сравнению с реальностью: это чистота данных, отсутствие лишних деталей.

В-третьих, специалисты по ИИ, кстати, обычно не знакомы с технологиями и инструментами ИМ. Мало того, они привыкли использовать бесплатный открытый код или писать на Python’е с нуля. Однако, когда речь идет о моделировании бизнес-задач, никакого даже минимально приличного бесплатного софта просто не существует, а написать его с нуля нереально. Поэтому нужно изменение парадигмы использования ПО и сотрудничество с консультантами по ИМ.

И наконец, сама по себе область ИИ в настоящее время представляет собой практически алхимию, полуслепой поиск, где полученным результатам радуются, но не всегда понимают, почему они появились. Выбор состояния и наблюдения, частоты циклов обучения, функции награды, структуры нейросети и её гиперпараметров – всё это делается путем проб и ошибок, является искусством. Надежду, правда, вселяют новые инструменты автоматизации разработки ИИ, например тот же Pathmind от Skymind или Bons.ai (теперь Microsoft, [https://www.bons.ai/]). В любом случае, на этом фоне старое доброе имитационное моделирование, конечно, выглядит хорошо разработанной технологией: здесь вы, по крайней мере, понимаете, какие действия приводят к какому результату.

Полезные ресурсы

Машинное обучение и имитационная модель – пример. Блог AnyLogic. https://www.anylogic.ru/blog/mashinnoe-obuchenie-i-imitatsionnaya-model-primer/

How To Build a Reinforcement Model With AnyLogic. Ben Schumann's blog. https://www.benjamin-schumann.com/blog/2018/11/9/how-to-build-a-reinforcement-learning-model-in-anylogic

Тренировка алгоритмов ИИ с помощью ИМ и глубокого обучения. Вебинар AnyLogic (видео на английском). https://www.anylogic.ru/blog/video-vebinara-trenirovka-algoritmov-ii-s-pomoshchyu-im-i-glubokogo-obucheniya/

Artificial Intelligence and Simulation. Panel discussion at AnyLogic Conference 2019. https://www.anylogic.com/resources/conference/#ai-panel