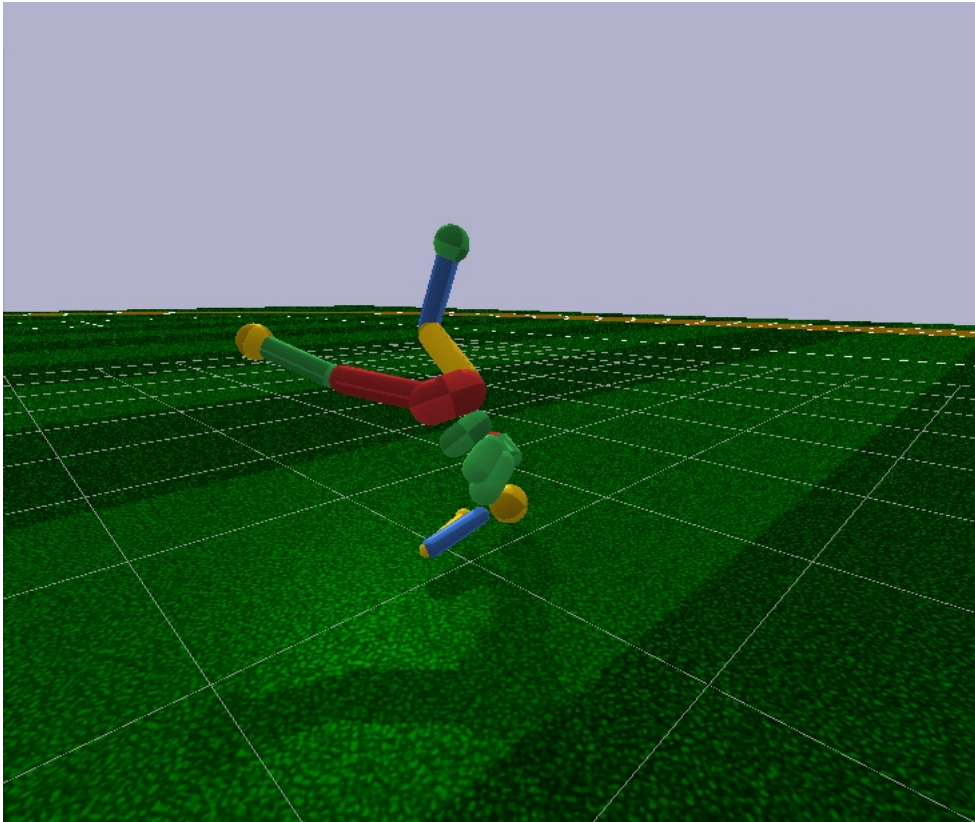


 DesertFlow сегодня в 02:25

Что не так с обучением с подкреплением (Reinforcement Learning)?

Искусственный интеллект, Машинное обучение



Еще в начале 2018 года вышла статья *Deep Reinforcement Learning Doesn't Work Yet* ("Обучение с подкреплением пока не работает"). Основными претензиями которой сводилась к тому, что современные алгоритмы обучения с подкреплением требуют для решения задачи примерно столько же времени, сколько и обычный случайный поиск.

Изменилось ли что-то с того времени? Нет.

Обучение с подкреплением считается одним из трех основных путей к созданию сильного ИИ. Но трудности, с которыми сталкивается эта область машинного обучения, и методы, которыми ученые пытаются бороться с этими трудностями, наводят на мысль, что, возможно, с самим этим подходом имеются фундаментальные проблемы.

Постойте, что значит один из трех? А остальные два какие?

С учетом успеха нейронных сетей в последние годы и анализом того, как они работают с высокоуровневыми когнитивными способностями, считавшимися ранее характерными только для человека и высших животных, на сегодняшний день в научном сообществе сложилось мнение, что можно выделить три основных подхода к созданию сильного ИИ на основе нейронных сетей, которые можно считать более менее реалистичными:

1. Обработка текстов

В мире накоплено огромное количество книг и текста в интернете, в том числе учебников и справочников. Текст удобен и быстр для обработки компьютером. Теоретически, этого массива текстов должно хватить для обучения сильного разговорного ИИ.

При этом подразумевается, что в этих текстовых массивах отражено полное устройство мира (как минимум, оно описано в учебниках и справочниках). Но это совершенно не факт. Тексты как вид представления информации сильно оторваны от реального трехмерного мира и течения времени, в котором мы живем.

Хорошими примерами ИИ, обученными на текстовых массивах, являются чат-боты и автоматические переводчики. Так как для перевода текста нужно понять смысл фразы и пересказать его новыми словами (на другом языке). Существует распространенное заблуждение, что правил

грамматики и синтаксиса, включая описание всех возможных исключений, полностью описывают конкретный язык. Это не так. Язык — это вспомогательный инструмент в жизни, он легко меняется и адаптируется под новые ситуации.

Проблема обработки текста (хоть экспертными системами, хоть нейронными сетями) в том, что **не существует** набора правил, какие фразы в каких ситуациях нужно применять. Обратите внимание — не правил построения самих фраз (чем занимается грамматика и синтаксис), а и какие фразы в каких жизненных ситуациях. В одной и той же ситуации люди произносят на разных языках фразы, которые вообще никак друг с другом не связаны с точки зрения структуры языка. Сравните фразы при крайней степени удивления: "о, боже!" и "o, holy shit!". Ну и как между ними провести соответствие, зная языковую модель? Да никак. Так случайно сложилось исторически. Нужно знать ситуацию и что в ней обговаривают на конкретном языке. Именно из-за этого автоматические переводчики пока такие несовершенные.

Можно ли выделить эти знания чисто из массива текстов — неизвестно. Но если автоматические переводчики станут идеально переводить, делая глупых и нелепых ошибок, то это будет доказательством, что создание сильного ИИ только на основе текста возможно.

2. Распознавание изображений

Посмотрите на это изображение



Глядя на эту фотографию мы понимаем, что съемка велась ночью. Судя по флагам, ветер дует справа налево. А судя по правостороннему движению, дело не происходит в Англии или Австралии. Никакая эта информация не указана явно в пикселях картинке, это внешние знания фото есть лишь признаки, по которым мы можем воспользоваться знаниями, полученными из других источников.

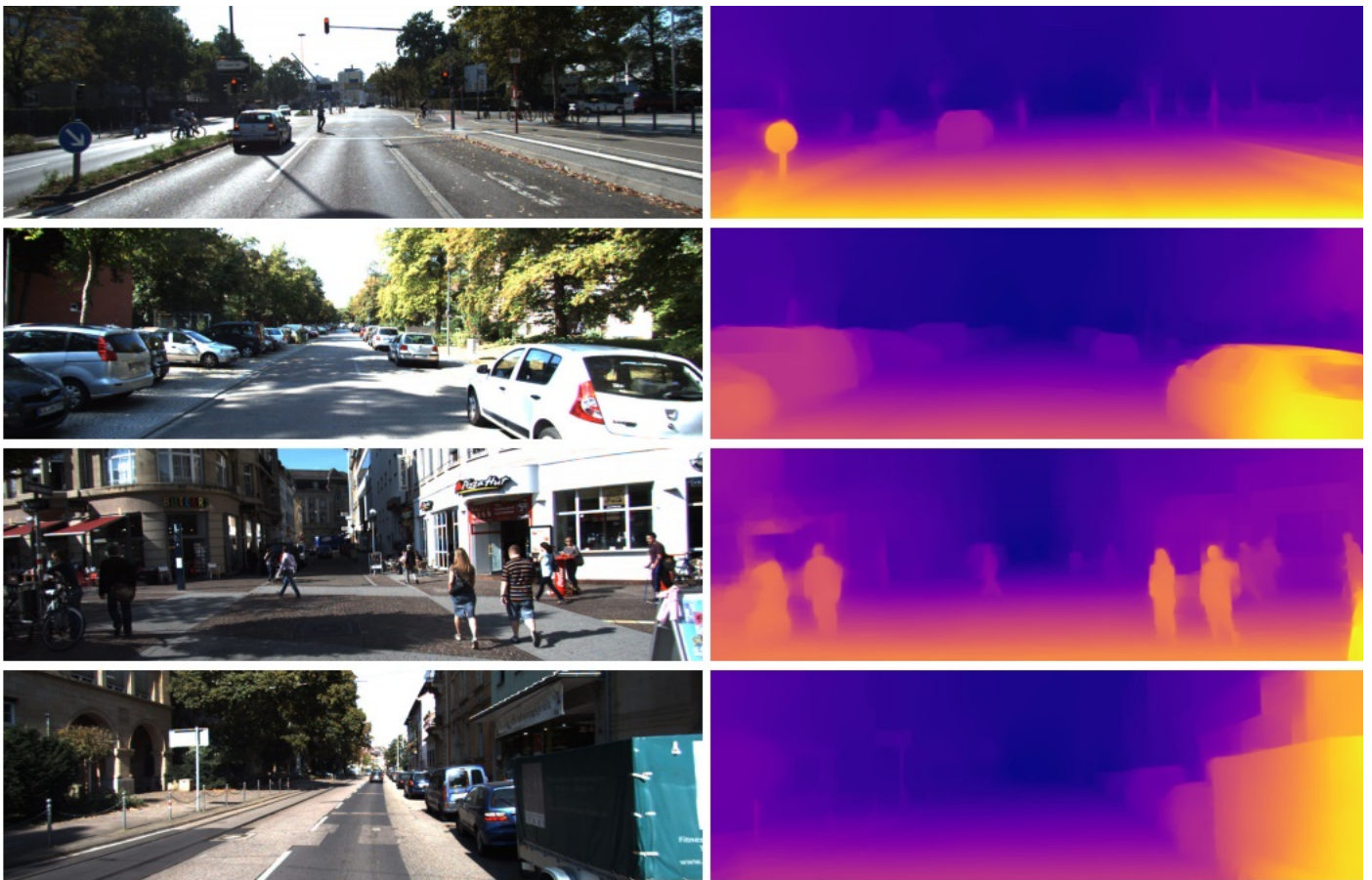
Вы знаете еще что-то, глядя на эту картинку?

О том и речь... И найдите себе девушку, наконец

Поэтому считается, что если обучить нейронную сеть распознавать объекты на картинке, то у нее сложится внутреннее представление о том, как устроен реальный мир. И это представление, полученное по фотографиям, уж точно будет соответствовать нашему реальному и настоящему миру. В отличие от массивов текстов, где это не гарантировано.

Ценность нейронных сетей, обученных на массиве фотографий ImageNet (а теперь и OpenImages V4, COCO, KITTI, BDD100K и другие) во всем факте распознавания котика на фото. А в том, что хранится в предпоследнем слое. Именно там находится набор высокоуровневых features: описывающих наш мир. Вектора в 1024 числа достаточно, чтобы из него получить описание 1000 разных категорий объектов с 80% точностью: в 95% случаев правильный ответ будет в 5 ближайших вариантах). Только подумайте в это.

Именно поэтому эти features из предпоследнего слоя так успешно используются в совершенно различных задачах по компьютерному зрению. Через Transfer Learning и Fine Tuning. Из этого вектора в 1024 числа можно получить, например, карту глубины по картинке



(пример из работы, где используется практически не измененная предобученная сеть Densenet-169)

Или определять позу человека. Применений много.



Как следствие, распознавание изображений потенциально можно использовать для создания сильного ИИ, так как оно действительно отражает модель нашего реального мира. От фотографии к видео один шаг, а видео — это и есть наша жизнь, так как около 99% информации мы получаем зрительно.

Но по фотографии совершенно непонятно, как мотивировать нейронную сеть думать и делать выводы. Ее можно обучить отвечать на вопросы вроде "сколько карандашей лежит на столе?" (этот класс задач называется Visual Question Answering, пример такого датасета: <https://visualqa.org/>) Или давать текстовое описание тому, что происходит на фото. Это класс задач Image Captioning.



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."

Но является ли это интеллектом? Развив этот подход, в недалеком будущем нейронные сети смогут отвечать по видео на вопросы вроде "И проводках сидело два воробья, один из них улетел, сколько осталось воробьев?". Это уже настоящая математика, в чуть более усложненных случаях недоступная животным и находящаяся на уровне человеческого школьного образования. Особенно, если кроме воробьев, там ряд будут сидеть синички, но их не нужно учитывать, так как вопрос был только про воробьев. Да, это определенно будет интеллект.

3. Обучение с подкреплением (Reinforcement Learning)

Идея очень проста: поощрять действия, ведущие к награде, и избегать ведущих к неудаче. Это универсальный способ обучения и, очевидно со всей определенностью может привести к созданию сильного ИИ. Поэтому к Reinforcement Learning такой большой интерес в последние

Смешать, но не взбалтывать

У обучения с подкреплением есть один большой плюс. В симуляторе можно создать упрощенную модель мира. Так, для фигурки человека достаточно всего 17 степеней свободы, вместо 700 в живом человеке (примерное число мышц). Поэтому в симуляторе можно решать зада очень маленькой размерности.

Забегаая вперед, современные алгоритмы Reinforcement Learning не способны произвольно управлять моделью человека даже с 17 степеней свободы. То есть не могут решить задачу оптимизации, где на входе 44 числа и на выходе 17. Удастся это сделать только в очень простых случаях, с тонкой ручной настройкой начальных условий и гиперпараметров. И даже в этом случае, например чтобы научить модель гуман 17 степенями свободы бегать, причем начиная с положения стоя (что намного проще), нужно несколько суток расчетов на мощном GPU. А более сложные случаи, например научиться вставать из произвольной позы, может вообще никогда не обучиться. Это провал.

Кроме того, все Reinforcement Learning алгоритмы работают с удручающе маленькими нейронными сетями, а с обучением больших не справляются. Крупные сверточные сети используются только чтобы снизить размерность картинки до нескольких features, которые и подак вход алгоритмам обучения с подкреплением. Тот же бегающий гуманоид управляется Feed Forward сетью с двумя-тремя слоями по 128 не! Серьезно? И на основе этого мы пытаемся построить сильный ИИ?

Чтобы попытаться понять, почему так происходит и что не так с обучением с подкреплением, надо сначала ознакомиться с основными архитектурами в современном Reinforcement Learning.

Физическое устройство мозга и нервной системы настроено эволюцией под конкретный вид животного и его условия обитания. Так, у мухи процессе эволюции развилась такая нервная система и такая работа нейромедиаторов в ганглиях (аналог мозга у насекомых), чтобы быст уворачиваться от мухобойки. Ну хорошо, не от мухобойки, а от птиц, которые их ловили 400 миллионов лет (шутка, птицы сами появились млн лет назад, скорее от лягушек 360 млн лет). А носорогу достаточно такой нервной системы и мозга, чтобы медленно повернуться в стор цели и начать бежать. А там, как говорится, у носорога плохое зрение, но это уже не его проблемы.

Но помимо эволюции, у каждой конкретной особи, начиная с рождения и в течении всей жизни, работает именно обычный механизм обуче подкреплением. В случае млекопитающих, да и насекомых тоже, эту работу выполняет дофаминовая система. Ее работа полна тайн и нюк но все сводится к тому, что в случае получения награды, дофаминовая система, через механизмы памяти, как-то закрепляет связи между нейронами, которые были активны непосредственно до этого. Так формируется ассоциативная память.

Которая, в силу своей ассоциативности, потом используется при принятии решений. Проще говоря, если текущая ситуация (текущие активы нейроны в этой ситуации) по ассоциативной памяти активируют нейроны памяти об удовольствии, то особь выбирает действия, которые ои делала в похожей ситуации и которые запомнила. "Выбирает действия" — это плохое определение. Выбора нет. Просто активированные н памяти об удовольствии, закрепленные дофаминовой системой для данной ситуации, автоматически активируют моторные нейроны, приводящие к сокращению мышц. Это если необходимо немедленное действие.

Искусственному обучению с подкреплением, как области знаний, необходимо решить обе эти задачи:

1. Подобрать архитектуру нейросети (что для нас уже сделала эволюция)

Хорошая новость в том, что высшие когнитивные функции, выполняющиеся в неокортексе у млекопитающих (и в полосатом теле у вранов) выполняются в примерно однородной структуре. Видимо, для этого не нужно какой-то жестко прописанной "архитектуры".

Разноплановость областей мозга, вероятно, объясняется чисто историческими причинами. Когда по мере эволюции новые части мозга нар на поверх базовых, оставшихся от самых первых животных. По принципу работает — не трогай. С другой стороны, у разных людей одинаковы части мозга реагируют на одинаковые ситуации. Это может объясняться как ассоциативностью (features и "нейроны бабушки" естественны образом сформировались в этих местах в процессе обучения), так и физиологией. Что сигнальные пути, закодированные в генах, ведут им этим областям. Единого мнения тут нет, но можно почитать, например, эту недавнюю стью: "Biological and artificial intelligence".

2. Научиться обучать нейронные сети по принципам обучения с подкреплением

Именно этим, в основном, и занимается современный Reinforcement Learning. И какие успехи? Не очень.

Наивный подход

Казалось бы, обучать нейросеть с подкреплением очень просто: делаем случайные действия, и если получили награду, то считаем сделанные действия "эталонными". Ставим их на выход нейросети как стандартные labels и обучаем нейронную сеть методом обратного распространения ошибки, чтобы она выдавала именно такой выход. Ну, самое обычное обучение нейросети. А если действия привели к неудаче, то либо игнорируем этот случай, либо подавляем эти действия (ставим эталонными на выходе какие-нибудь другие, например любое другое случайное действие). В общем и целом, эта идея повторяет дофаминовую систему.

Но если вы попытаетесь так обучать любую нейронную сеть, неважно насколько сложной архитектуры, рекуррентную, сверточную или обычную прямого распространения, то... Ничего не выйдет!

Почему? Неизвестно.

Считается, что полезный сигнал настолько мал, что теряется на фоне шума. Поэтому стандартным методом обратного распространения сеть не обучается. Награда случается очень редко, может один раз из сотен или даже тысяч шагов. А даже LSTM запоминает максимум 10 точек истории, и то лишь в очень простых задачах. А на более сложных если будет 10-20 точек истории, то уже хорошо.

Но корень проблемы именно в очень редких наградах (по крайней мере в задачах, представляющих практическую ценность). На данный момент мы не умеем обучать нейросети, которые запоминали бы единичные случаи. С чем мозг справляется с блеском. Можно что-то, случившееся всего один раз, запомнить на всю жизнь. И, кстати, большая часть обучения и работы интеллекта строится именно на таких случаях.

Это что-то вроде жуткого дисбаланса классов из области распознавания изображений. Способов бороться с этим просто нет. Лучшее, что смогли придумать — это просто подавать на вход сети наравне с новыми ситуациями, сохраненные в искусственном специальном буфере удачные ситуации из прошлого. То есть, постоянно обучать не только новым случаям, но и удачным старым. Естественно, нельзя бесконечно увеличивать такой буфер, да и непонятно что именно в нем хранить. Еще пытаются как-то на время фиксировать пути внутри нейросети, чтобы активными во время удачного случая, чтобы последующее обучение их не перезаписывало. Довольно близкая аналогия к происходящему в мозге, на мой взгляд, хотя особых успехов в этом направлении тоже пока не добились. Так как новые обученные задачи в своем расчете используют и результаты выхода нейронов из замороженных путей, то в итоге сигнал только по этим замороженным интерферирует с новыми старыми задачами перестают работать. Есть еще один любопытный подход: обучать сеть новым примерам/задачам только в ортогональном направлении к предыдущим задачам (<https://arxiv.org/abs/1810.01256>). Это не перезаписывает предыдущий опыт, но резко ограничивает ее возможности.

Отдельным классом алгоритмов, призванных бороться с этой бедой (а заодно дарящих надежду достичь сильного ИИ), идут разработки в Meta-Learning. Это попытки обучить нейросеть сразу нескольким задачам. Не в смысле, что распознавать разные картинки в одной задаче, а именно разным задачам в разных доменах (каждый со своим распределением и ландшафтом решений). Скажем, распознавать картинки и одновременно ездить на велосипеде. Успехи пока тоже не очень, так как обычно все сводится к тому, чтобы заранее подготовить нейросеть с общими универсальными весами, а потом быстро, всего за несколько шагов градиентного спуска, доадаптировать их к конкретной задаче. Пример алгоритмов метаобучения — MAML и Reptile.

В общем, только эта проблема (невозможность учиться на единичных удачных примерах) ставит крест на современном обучении с подкреплением. Вся мощь нейросетей перед этим печальным фактом пока бессильна.

Этот факт, что самый простой и очевидный способ не работает, заставил исследователей вернуться к классическому табличному Reinforcement Learning. Который как наука появился еще в седой древности, когда нейросети не были даже в проекте. Но теперь, вместо ручного подсчета значений в таблицах и в формулах, давайте в качестве целевых функций использовать такой мощный аппроксиматор, как нейронные сети! вся суть современного Reinforcement Learning. И главное его отличие от обычного обучения нейросетей.

Q-learning и DQN

Reinforcement Learning (еще до нейросетей) зародился как довольно простая и оригинальная идея: давайте делать, опять же, случайные действия, а потом для каждой ячейки в таблице и каждого направления движения, посчитаем по специальной формуле (получившей название уравнение Беллмана, это слово вы будете встречать практически в каждой работе по обучению с подкреплением), насколько хороша эта я выбранное направление. Чем выше получится это число, тем с большей вероятностью этот путь ведет к победе.

-0.08	-0.04	-0.04	1.00
-0.08 0.88	-0.07 0.92	-0.04 0.96	
-0.09	-0.04	-0.03	
0.84		0.51	-0.88
-0.10 -0.10		-0.05 -0.41	
-0.10		-0.05	
0.79	-0.10	0.08	-0.27
-0.12 -0.12	-0.12 -0.10	-0.09 -0.08	-0.08 -0.08
-0.12	-0.10	-0.09	-0.08

В какой ячейке вы бы ни появились, двигайтесь по нарастанию зеленого цвета! (в сторону максимального числа по бокам текущей ячейки).

Это число получило название Q (от слова quality — качество выбора, очевидно), а метод — Q-learning. Заменяв формулу расчета этого числа нейронную сеть, а точнее обучая нейронную сеть по этой формуле (плюс еще пара трюков, связанных чисто с математикой обучения нейросетей), в Deepmind получили метод DQN. Это который в 2015 году победил в куче Atari игр и положил начало революции в Deep Reinforcement Learning.

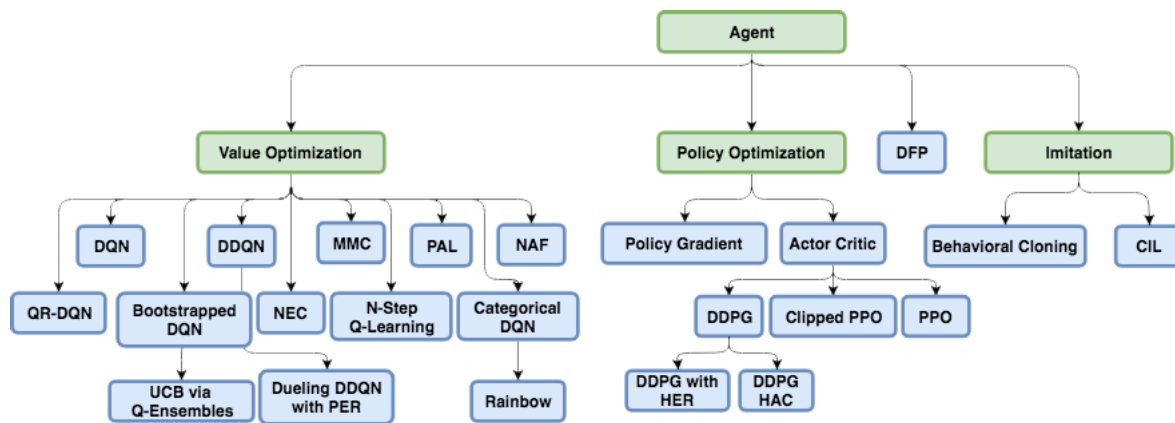
К сожалению, этот метод по своей архитектуре работает только с дискретными discrete действиями. В DQN на вход нейросети подается текущий state (текущая ситуация), а на выходе нейросеть предсказывает число Q. А так как на выходе сети перечислены сразу все возможные действия (каждый со своим предсказанным Q), то получается что нейросеть в DQN реализует классическую функцию $Q(s,a)$ из Q-learning. Выдает Q state и action (поэтому обозначение $Q(s,a)$ как функции от s и a). Мы просто ищем обычным `argmax` по массиву среди выходов сети ячейку с максимальным числом Q и делаем действие, которое соответствует индексу этой ячейки.

Причем можно всегда выбирать действие с максимальным Q, тогда такая политика будет называться детерменистской. А можно выбирать действие как случайное из доступных, но пропорционально их Q-значениям (т.е. действия с высоким Q будут выбираться чаще, чем с низким). Такая политика называется стохастической. У стохастического выбора плюс в том, что автоматически реализуется поиск и исследование (Exploration), так как каждый раз выбираются разные действия, иногда не кажущиеся самыми оптимальными, но могущие в будущем привести к большой награде. И тогда мы обучимся и повысим этим действиям вероятность, чтобы теперь они чаще выбирались согласно их вероятности.

Но что делать, если вариантов действий бесконечно? Если это не 5 кнопок на джойстике в Atari, а continuous момент управления двигателем робота? Конечно, момент в диапазоне $-1..1$ можно разбить на поддиапазоны по 0.1, и в каждый момент времени выбирать один из этих поддиапазонов, словно нажатие джойстика в Atari. Но не всегда нужное число можно дискретизировать на интервалы. Представьте, что вы на велосипеде по горному пику. И можете поворачивать руль только на 10 градусов влево или вправо. В какой-то момент пик может стать настолько узким, что повороты на 10 градусов в обоих направлениях приведут к падению. Это принципиальная проблема дискретных действий. Еще DQN не работает с большими размерностями, и на роботе даже с 17 степенями свободы просто не сходится. Все хорошо, но есть нюанс, как говорится.

В дальнейшем было разработано много оригинальных и местами гениальных алгоритмов на основе DQN, позволивших, в том числе, работать с continuous действиями (за счет хитростей и введения дополнительных нейросетей): DDQN, DuDQN, BDQN, CDQN, NAF, Rainbow. Пожалуй, можно также отнести Direct Future Prediction (DFP), который роднится с DQN архитектурой сети и дискретными действиями. Вместо предсказания числа Q для всех действий, DFP напрямую предсказывает сколько на следующем шаге будет здоровья или патронов, если выбрать это действие. Причем на один шаг вперед и на несколько шагов вперед. Нам остается лишь перебрать все выходы сети и найти максимальное значение интересующего нас параметра и выбрать соответствующее этому элементу массива действие, в зависимости от текущих приоритетов. Например, если мы ранены, то можем среди выходов сети искать действие, ведущее к максимальному увеличению здоровья.

Но что еще важнее, за последующее время были разработаны новые архитектуры специально для Reinforcement Learning.



Policy Gradient

Давайте на вход сети подавать текущий state, а на выходе сразу предсказывать действия (либо сами действия, либо распределение вероятностей для них в стохастической политике). Мы просто действуем, применяя actions, предсказанные нейросетью. А потом смотрим, награду R набрали за эпизод. Эта награда может быть либо выше начальной (когда выиграли в игре), либо ниже (проиграли в игре). Также награду сравнивать со некоей средней наградой. Выше она средней или ниже.

Собственно, динамику полученной награды R в результате действий, которые подсказала нейросеть, можно использовать для вычисления градиента по специальной формуле. И применить этот градиент к весам нейросети! И дальше использовать обычное обратное распространение ошибки. Просто вместо "эталонных" действий на выходе сети в качестве labels (мы ведь не знаем какие они должны быть), используем изм награды для расчета градиента. По этому градиенту сеть обучится, чтобы предсказывать действия, которые ведут к увеличению награды F

Это классический Policy Gradient. Но у него есть недостаток — надо ждать окончания эпизода, чтобы посчитать куммулятивную награду R , чем изменять веса сети согласно ее изменению. А из преимуществ — гибкая система поощрений и наказаний, которая не только работает стороны, но также зависит от величины награды. Большая награда сильнее поощряет действия, которые к ней привели.

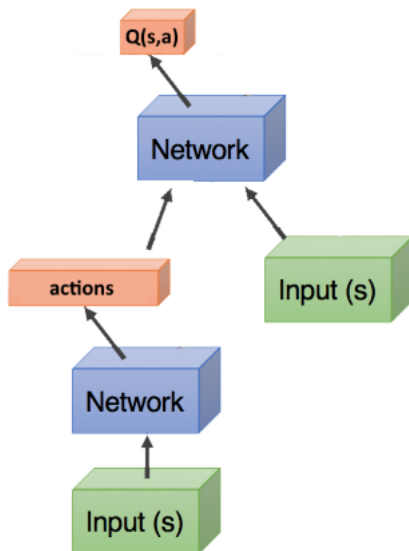
Actor-critic, DDPG

А теперь представьте, что у нас есть две сети — одна предсказывает какие действия надо совершить, а вторая оценивает насколько эти действия хороши. То есть, выдает Q -число для этих действий, как в алгоритме DQN. На вход первой сети подается state, а она предсказывает action. Вторая сеть на вход тоже получает state, но еще и действия action, предсказанные первой сетью, а на выходе выдает число Q как функцию от обоих: $Q(s,a)$.

Собственно, это число $Q(s,a)$, выданное второй сетью (ее называют critic, критик), точно также можно использовать для вычисления градиента которым обновлять веса первой сети (которую называют актером, actor), как мы делали выше с наградой R . Ну а вторая сеть обновляется обычным путем, согласно реальному прохождению эпизода. Этот метод получил название actor-critic. Его плюс по сравнению с классическим Policy Gradient, что веса сети можно обновлять на каждом шаге, не дожидаясь окончания эпизода. Что ускоряет обучение.

В таком виде это сеть DDPG. Так как она предсказывает напрямую действия actions, то прекрасно работает с continuous действиями. DDPG является прямым continuous конкурентом DQN с его дискретными действиями.

DDPG



Advantage Actor Critic (A3C/A2C)

Следующим шагом стало использование для обучения первой сети не просто предсказания критиком critic числа $Q(s,a)$ — насколько хорош действия, предсказанные актером actor, как это было в DDPG. А насколько эти предсказанные действия оказались лучше или хуже, чем мы ожидали.

Это очень близко к тому, что происходит в биологическом мозге. Из экспериментов известно, что максимальный выброс дофамина происходит во время самого получения удовольствия, а во время **ожидания**, что скоро получим удовольствие. Впрочем, если ожидания не оправдались наступают ужасные последствия, большие чем в обычном случае (в организме присутствует специальная система наказания, обратная сила вознаграждения).

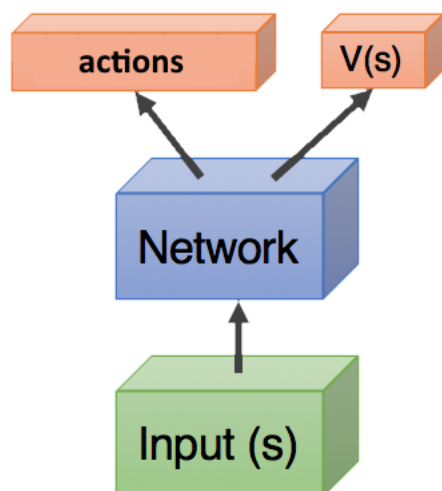
Для этого для расчета градиентов стали использовать не число $Q(s,a)$, а так называемое Advantage: $A(s,a) = Q(s,a) - V(s)$. Число $A(s,a)$ показывает не абсолютное качество $Q(s,a)$ выбранных действий, а относительное преимущество — насколько после предпринятых действий станет лучше, чем текущая ситуация $V(s)$. Если $A(s,a) > 0$, то градиент будет изменять веса нейросети, поощряя предсказанные сетью действия. Если $A(s,a) < 0$, то градиент будет изменять веса так, что предсказанные действия будут подавляться, т.к. они оказались плохие.

В этой формуле $V(s)$ показывает насколько хорош текущий state сам по себе, без привязки к действиям (поэтому зависит только от s , без a). Мы стоим в шаге от вершины Эвереста — это очень хорошая ситуация state, с большим $V(s)$. А если мы уже сорвались и падаем, то это очень плохой state, с низким $V(s)$.

К счастью, при таком подходе $Q(s,a)$ можно заменить на награду r , которую получим после совершения действия, и тогда формула преимущества для расчета градиентов получается $A = r - V(s)$.

В таком случае, достаточно предсказывать только $V(s)$ (а награду мы посмотрим уже по факту что получится в реальности), и две сети — actor и critic, можно объединить в одну! Которая получает на вход state, а на выходе разделяется на две головы head: одна предсказывает действия actions, а другая предсказывает $V(s)$. Такое объединение помогает лучше переиспользовать веса, т.к. обе сети должны на входе получать s . Впрочем, можно использовать и две отдельные сети.

Advantage Actor Critic (A3C/A2C)



Учет и предсказание сетью качества текущей ситуации $V(s)$ в любом случае помогает ускорить обучение. Так как при плохом $V(s)$, где уже нельзя исправить ни при каких действиях action (мы летим вниз головой с Эвереста), можно не искать дальше пути решения. Это используется в Dueling Q-Network (DuDQN), где $Q(s,a)$ внутри сети специально раскладывается на $Q(s,a) = V(s) + A(s,a)$, а потом собирается обратно.

Asynchronous Advantage Actor Critic (A3C) означает всего лишь, что есть сервер, собирающий результаты от множества акторов. И обновлять веса как только набирается батч batch нужного размера. Поэтому асинхронный, что не ждет каждого актора. Это вроде как разбавляет приме убирая из них ненужную корреляцию, что улучшает обучение. С другой стороны, потом появился A2C — синхронная версия A3C, в которой сервер дожидается окончания эпизодов у всех акторов и только после этого обновляет веса (поэтому синхронный). A2C тоже показывает хорошие результаты, поэтому применяются обе версии, в зависимости от вкуса разработчика.

TRPO, PPO, SAC

Собственно, на этом прогресс закончился.

Не смотря на красивое и выглядящее логичным описание, все это работает не очень. Даже лучшие Reinforcement Learning алгоритмы требуют десятки миллионов примеров, сравнимы по эффективности со случайным поиском, а самое печальное, что не позволяет с их помощью создать сильный ИИ — работают лишь на крайне низких размерностях, исчисляемых единицами. Даже не десятками.

Дальнейшее улучшение — TRPO и PPO, являющиеся сейчас state-of-the-art, являются разновидностью Actor-Critic. На PPO в настоящее время обучают большинство агентов в мире RL. К примеру, им обучали OpenAI Five для игры в Dota 2.

Вы будете смеяться, но все что придумали в методах TRPO и PPO — это ограничивать изменение нейронной сети при каждом обновлении веса резко не менялись. Дело в том, что в A3C/A2C бывают резкие изменения, которые портят предыдущий опыт. Если сделать, чтобы новая policy не слишком отличалась от предыдущей, то можно избежать таких выбросов. Что-то вроде gradient clipping в рекуррентных сетях для защиты от взрывающихся градиентов, только на другом математическом аппарате. Сам факт того, что приходится так грубо обрезать и ухудшать обучение (большие градиенты там ведь не просто так появились, они нужны для вызвавшего их примера), и что это дает положительный эффект говорит о том, что мы свернули куда-то не туда.

В последнее время возрастающей популярностью пользуется алгоритм Soft-Actor-Critic (SAC). Он не сильно отличается от PPO, только добавляет цель при обучении повышать энтропию в policy. Делать поведение агента более случайным. Нет, не так. Чтобы агент был способен действовать в более случайных ситуациях. Это автоматически повышает надежность политики, раз агент готов к любым случайным ситуациям. Кроме того требует немного меньше примеров для обучения, чем PPO, и менее чувствителен к настройке гиперпараметров, что тоже плюс. Однако для SAC, чтобы обучить бегать гуманоида с 17 степенями свободы, начиная с позиции стоя, нужно около 20 млн кадров и примерно сутки расчета на одном GPU. Более сложные начальные условия, скажем, научить вставать гуманоида из произвольной позы, может вообще не обучиться.

Итого, общая рекомендация в современном Reinforcement Learning: использовать SAC, PPO, DDPG, DQN (в таком порядке, по убыванию).

Model-Based

Существует еще один интересный подход, косвенно касающийся обучения с подкреплением. Это построить модель окружающей среды, и использовать ее для прогнозирования, что произойдет если мы предпримем какие-то действия.

Его недостатком является то, что он никак не говорит, какие действия нужно предпринять. Лишь об их результате. Но зато такую нейронную сеть легко обучать — просто обучаем на любой статистике. Получается что-то вроде симулятора мира на основе нейронной сети.

После этого генерируем огромное количество случайных действий, и каждое прогоняем через этот симулятор (через нейронную сеть). И скажем какое из них принесет максимальную награду. Есть небольшая оптимизация — генерировать не просто случайные действия, а отклоняющиеся от текущего закона от текущей траектории. И действительно, если мы поднимаем руку, то с большой вероятностью нужно продолжать ее поднимать. Поэтому в первую очередь нужно проверить минимальные отклонения от текущей траектории.

Здесь фокус с тем, что даже примитивный физический симулятор вроде MuJoCo или pyBullet выдает около 200 FPS. А если обучить нейронную сеть прогнозировать вперед хотя бы на несколько шагов, то для простых окружений легко можно за один раз получать батчи по 2000-5000 предсказаний. В зависимости от мощности GPU, в секунду можно получить прогноз для десятков тысяч случайных действий благодаря параллелизации в GPU и сжатости вычислений в нейросети. Нейросеть здесь просто выполняет роль очень быстрого симулятора реально

Кроме того, раз уж нейросеть может прогнозировать реальный мир (это и есть model-based подход, в общем смысле), то можно проводить обучение целиком в воображении, так сказать. Эта концепция в Reinforcement Learning получила название Dream Worlds, или World Models неплохо работает, хорошее описание есть тут: <https://worldmodels.github.io>. Кроме того, это имеет природный аналог — обычные сны. И многократная прокрутка недавних или планируемых событий в голове.

Imitation Learning

От бессилия, что алгоритмы Reinforcement Learning не работают на больших размерностях и сложных задачах, народ задался целью хотя бы повторить действия за экспертами в виде людей. Здесь удалось достичь неплохих результатов (недостижимых обычным Reinforcement Learning). Так, OpenAI получилось пройти игру Montezuma's Revenge. Фокус оказался прост — помещать агента сразу в конец игры (в конец показанной человеком траектории). Там с помощью PPO, благодаря близости финальной награды, агент быстро учится идти вдоль траектории. После помещаем его немного назад, где он быстро учится доходить до того места, которое он уже изучил. И так постепенно сдвигая точку "respawn" вдоль траектории до самого начала игры, агент учится проходить/имитировать траекторию эксперта в течении всей игры.

Другой впечатляющий результат — повторение движений за людьми, снятые на Motion Capture: DeepMimic. Рецепт аналогичен методу OpenAI, каждый эпизод начинаем не с начала траектории, а со случайной точки вдоль траектории. Тогда PPO успешно изучает окрестности этой то

Надо сказать, что нашумевший алгоритм Go-Explore от Uber, прошедший с рекордными очками игру Montezuma's Revenge, вообще не являлся алгоритмом Reinforcement Learning. Это обычный случайный поиск, но начиная со случайной посещенной ранее ячейки cell (огрубленной ячеей в которую попадают несколько state). И только когда таким случайным поиском будет найдена траектория до конца игры, уже по ней с помощью Imitation Learning обучается нейросеть. Способом, аналогичным как в OpenAI, т.е. начиная с конца траектории.

Curiosity (любопытство)

Очень важным понятием в Reinforcement Learning является любопытство (Curiosity). В природе оно является двигателем для исследования окружающей среды.

Проблема в том, что в качестве оценки любопытства нельзя использовать простую ошибку предсказания сети, что будет дальше. Иначе сеть зависнет перед первым же деревом с качающейся листвой. Или перед телевизором со случайным переключением каналов. Так как результат из-за сложности будет невозможно предсказать и ошибка всегда будет большой. Впрочем, именно это и является причиной, почему (люди) так любим смотреть на листву, воду и огонь. И на то, как другие люди работают =). Но у нас есть защитные механизмы, чтобы не застрять навечно.

Один из таких механизмов придумали как Inverse Model в работе Curiosity-driven Exploration by Self-supervised Prediction. Если коротко, агент (нейронная сеть) кроме того, что предсказывает какие действия лучше всего совершить в данной ситуации, дополнительно пытается предсказать что будет с миром после совершенных действий. И использует это свое предсказание мира: следующего шага, чтобы по нему и по текущему шагу обратно предсказать свои же предпринятые ранее действия (да, сложно, без поллитры разобраться).

Это приводит к любопытному эффекту: агент становится любопытным только к тому, на что он может повлиять своими действиями. На качающиеся ветки дерева он никак не может повлиять, поэтому они становятся ему неинтересны. А вот походить по округе он может, поэтому любопытно ходить и исследовать мир.

Однако если у агента будет пульт от телевизора, переключающий случайные каналы, то он может на него повлиять! И ему будет любопытно щелкать каналы до бесконечности (так как не может предсказать, какой будет следующий канал, т.к. он случайный). Попытка обойти эту проблему предпринята в Google в работе Episodic Curiosity through Reachability.

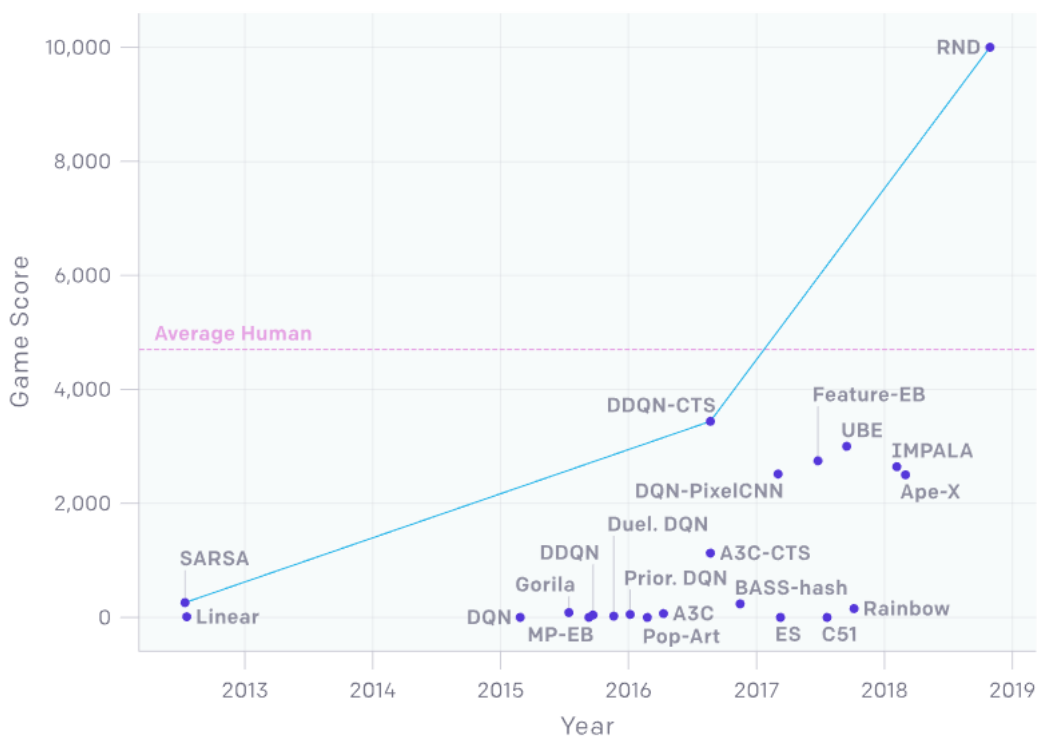
Но, пожалуй, лучший state-of-the-art результат по любопытству, на данный момент принадлежит OpenAI с идеей Random Network Distillation. Ее суть в том, что берется вторая, совершенно случайно инициализированная сеть, и ей на вход подается текущий state. А наша основная рабочая нейросеть пытается угадать выход этой нейросети. Вторая сеть не обучается, она остается все время фиксированной как была инициализирована.

В чем смысл? Смысл в том, что если какой-либо state уже был посещен и изучен нашей рабочей сетью, то она более менее успешно сможет предсказывать выход той второй сети. А если это новый state, где мы никогда не были, то наша нейросеть не сможет предсказать выход той сети. Эта ошибка в предсказании выхода той случайно инициализированной сети используется как показатель любопытства (дает высокие награды, если в данной ситуации не можем предсказать ее выход).

Почему это работает, не совсем понятно. Но пишут, что это устраняет проблему когда цель предсказания стохастическая и когда недостаток данных, чтобы самому сделать предсказание что будет дальше (что дает большую ошибку предсказания в обычных алгоритмах любопытства). Так или иначе, но RND реально показал отличные результаты по исследованию на основе любопытства в играх. И справляется с проблемами случайного телевизора.

С помощью RND любопытства в OpenAI впервые честно (а не через предварительный случайный поиск, как в Uber) прошли первый уровень Montezuma's Revenge. Не каждый раз и ненадежно, но время от времени получается.

Progress in Montezuma's Revenge




Что в итоге?

Как видите, всего за несколько лет Reinforcement Learning прошел долгий путь. Не просто несколько удачных решений, как в сверточных сетях, где residual и skip connections позволили тренировать сети глубиной в сотни слоев, вместо десятка слоев с одной только функцией активации Relu, поборовшей проблему исчезающих градиентов в сигмоиде и tanh. В обучении с подкреплением произошел прогресс в концепциях и понимании причин, почему не заработал тот или иной наивный вариант реализации. Ключевое слово "не заработал".

Но с технической стороны все по-прежнему упирается в предсказания все тех же Q, V или A значений. Ни временных зависимостей на разных масштабах, как в мозге (Hierarchical Reinforcement Learning не в счет, уж больно примитивная в нем иерархия по сравнению с ассоциативным живым мозгом). Ни попыток придумать архитектуру сети, заточенную именно под обучение с подкреплением, как это произошло с LSTM и другими рекуррентными сетями для временных последовательностей. Reinforcement Learning либо топчется на месте, радуясь небольшим успехам, либо движется в каком-то совсем уж неправильном направлении.

Хочется верить, что однажды в обучении с подкреплением произойдет прорыв в архитектуре нейронных сетей, аналогичный тому, что произошел в сверточных сетях. И мы увидим по-настоящему работающее обучение с подкреплением. Обучающееся на единичных примерах, с работой ассоциативной памятью и работающее на разных временных масштабах.

Теги: обучение с подкреплением, reinforcement learning, искусственный интеллект, нейронные сети, машинное обучение



10,0

Карма

35,2

Рейтинг

1

Подписчики

@DesertFlow

Пользователь

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

- 26 ноября 2017 в 15:25

Текстовые капчи легко распознаются нейронными сетями глубокого обучения

+75

34,3k

178

100
- 13 марта 2017 в 13:14

Робот-собеседник на основе ИНС: рекуррентные сети

+8

3,2k

31

3
- 5 августа 2010 в 00:38

По следам интеллекта

+30


4,5k

79

68

ЗАКАЗЫ	Фриланс
Разработка дистанционных курсов на онлайн-платформе 1 отклик · 44 просмотра	20000 за проект
Проект по распознаванию документов (OpenCV, OCR, Tesseract, ML) 13 откликов · 89 просмотров	10000 за проект
Видеомейкер, скринкастмейкер с базовыми знаниями интернет-маркетинга 0 откликов · 26 просмотров	10000 за проект
Разработка программно-аппаратного устройства для людей с ОВЗ 22 отклика · 268 просмотров	95000 за проект
Разработать приложение на C++ с использованием технологии CUDA 6 откликов · 80 просмотров	60000 за проект
Все заказы	Разместить заказ

Комментарии 19


- 

lostmsu

сегодня в 08:24

#

🔖


Вывод, на мой взгляд, высосан из пальца. Статья — кликбейт для срача.
Просто 90% нетехническую статью с заголовком «Прогресс RL за последние 3 года» никто не стал бы читать.
- 

Hedgehogues

сегодня в 09:46

#

🔖

Тема порно не достаточно раскрыта. Думаю, что автору тоже можно найти себе девушку
- 

DesertFlow

сегодня в 12:25

#

🔖

📄

🔄

Этот пример для того, что направление ветра все же можно выделить из картинки. Выделив объект флаг и определив его направление. Но вот что за здание, из пикселей картинки точно никак нельзя узнать. Это внешнее знание. Без наличия этого знания с этим не справиться.

Поэтому распознавание изображений это не про поиск паттернов в пикселях. Это в первую очередь ментальная модель мира. А раз эта модель вмещает в себя такое разнообразие визуальных объектов, то в принципе, может вмещать и еще что-то. Взаимоотношения между объектами, дин (по видео) и т.д... А это уже очень близко к интеллекту.






Поэтому распознавание изображений в современной форме это хоть и слабая, но все же форма ИИ. А где слабая, там есть потенциал и для сил

 Sklott  сегодня в 09:48  

Мне кажется у автора странное понимание об интеллекте.

Можно сделать сетку (уже есть примеры), которая будет «понимать» текст или картинку. Т.е. проводить с заданным текстом/картинкой все те операи может провести с ним человек. Сейчас ограничения в основном только в объёмах текста и отсутствии опыта из реальной жизни. Но полноценным интеллектом такая сетка никогда не станет, максимум некий механизм ввода «перекодирующий» внешнюю информацию в некое внутреннее представление.

Для настоящего интеллекта в первую очередь нужны механизмы памяти короткой и длинной. И гугловцы уже что-то по этому поводу химичат, но подробностей не раскрывают, только общий смысл: [Hybrid computing using a neural network with dynamic external memory](#)
PDFничек, если кому интересно.

 DesertFlow  сегодня в 12:18    

Для настоящего интеллекта в первую очередь нужны механизмы памяти короткой и длинной.

Именно. Но ни в одном из state-of-the-art алгоритмов Reinforcement Learning нет ни короткой, ни долговременной памяти (если не считать «запомненное» при обучении). Они вообще в принципе не приспособлены к Lifelong Learning, то есть постоянному дообучению в течении жизни. : конечно хорошо, что хоть что-то удается сделать. Но существующая ситуация очень далека до настоящего обучения с подкреплением, которое существует в биологических системах.

 Sklott  сегодня в 12:30    

Но ни в одном из state-of-the-art алгоритмов Reinforcement Learning нет ни короткой, ни долговременной памяти

А при чем тут вообще Reinforcement Learning? Reinforcement Learning — это по сути методология обучения. К архитектуре оно имеет очень опосредованное отношение...

Это в той сетке которую учат таким методом должна быть такая память.

ЗЫ: И нужна такая память кстати и сеткам которые обучают и обычными методами. Например для NMT неплохо бы хранить контекст всего документа который переводишь, а не одного предложения как сейчас.

 DesertFlow  сегодня в 12:47    

Просто в обучении с подкреплением память (желательно быстрая, чтобы реагировала на единичные удачные случаи) это практически необходимое свойство. Ведь надо как-то запоминать моменты успеха. Но ее в существующем RL вообще нет. А все что есть — это аппроксимация нейросетью $Q(s,a)$ или $V(s)$, который в итоге приводится к тому же Q .

Нет, есть конечно попытки. Как например нейропластичность от Uber (ее тут можно рассматривать как аналог памяти).

Есть еще Fast Weights, вторая копия весов, которые быстрее обновляются, а потом переводятся к основным. Вот это почти прямой аналог краткосрочной памяти. Но в RL этот тип сети, кажется, ни разу не использовался. И если почитать последующие работы, там выявилась куч проблем, поэтому маловероятно что это будет работать.

 stanislavnikitin  сегодня в 11:05  

Можно ли отнести обучение автономных автомобилей к Meta-Learning алгоритмам?

 DesertFlow  сегодня в 12:13    







Если автомобиль одной нейросетью будет и видеть картинку с камер, и управлять рулем, то да. В машинном обучении есть такое понятие — дом или характерное распределение примеров для конкретной задачи. Когда нейронная сеть преобразует данные, она переводит одно распределение (статистическое, вероятностное) в другое. А GAN, например, создают такое распределение с нуля и подгоняют его под текущую задачу. В сети ес картинки с примерами, как это происходит в динамике.

Хотя есть примеры, когда одна нейросеть хорошо работает сразу с несколькими задачами (и иногда даже лучше, чем с одной, так как переиспользуются общие веса), но обычно обучить нейронную сеть под два сильно отличающихся домена непросто.

Там еще возникает множество чисто технических проблем. Например, с нормализацией выходов. Если на одном выходе (отвечающем за положение руля, скажем) числа получаются больше по величине, то и градиент от них выходит выше. И сеть как бы больше учится рулить, а на картинки с кз забывает. В большинстве случаев вручную подгоняют специальный коэффициент, чтобы числа более менее привести к одному диапазону. Метод автоматической нормализации в нейросетях (слоев, выходов, наград в RL) начали развиваться относительно недавно и пока несовершенны.

 **Stepan555** сегодня в 12:44  

Логика статьи примерно такая — мы не можем сейчас полететь на Альфу Центавра, поэтому космонавтика не работает.

 **DesertFlow**  сегодня в 12:49    

Особенно если для того, чтобы полететь на Альфу Центавра, мы копаем тоннель к центру Земли.

Но что удивительно, это работает! Ведь когда Альфа Центавра находится с противоположной стороны, мы действительно таким образом к ней приближаемся. Вот примерная аналогия тому что происходит в мире RL. И какими способами исследователи добиваются успеха.

 **Alexey_mosc** сегодня в 13:00  




За 10 лет метод эволюционировал от возможности балансировать палку в тележке (cart-pole) до игры в стратегию с непрерывными действиями на у чемпионов (youtube). А так, да, простые нейронки и предсказание валуе/полиси. Даже странно, что работает. :)

 **JustDont** сегодня в 13:35    

до игры в стратегию с непрерывными действиями на уровне чемпионов

На уровне хороших не игравших вместе игроков.
До чемпионов там примерно как до Альфы Центавра.

Что очень характерно, на триллиарды игр, сыгранных OpenAI для обучения, людям оказалось достаточно всего лишь двух (или трех, уже не помню) публичных игр OpenAI для выработки успешной выигрышной стратегии против одного.

 **Alexey_mosc** сегодня в 14:26    

Что очень характерно, на триллиарды игр, сыгранных OpenAI для обучения, людям оказалось достаточно всего лишь двух (или трех, уже не помню) публичных игр OpenAI для выработки успешной выигрышной стратегии против одного.

Это характерно для машинного обучения в целом и очевидно почему так. Градиентный спуск не сходится за пару-тройку итераций.

 **JustDont**  сегодня в 14:41    

Да, конечно. Но это один из фундаментальных моментов, демонстрирующих идейный тупик ML в направлении достижения сильного ИИ. Пр всего экономически: гонять симуляцию с огромным количеством повторов — это очень серьезные вычислительные ресурсы, даже для таких: дичайше примитивных симуляций, как игрушки, или там, простейшая ньютоновская физика для задач «встать из произвольной позы».

 **DesertFlow**  сегодня в 14:24  


del

 **arikshtein** сегодня в 14:25  

Ответ прост. пока явных применений RL в бизнесе — нет. Т.е. заработать на нем деньги очень трудно. Сверточные сети пригодились гуглу, фб, да кому угодно у кого есть доступ к фотографиям, то же самое с LSTM, Signal Processing и так далее.

Ну предположим у тебя есть алгоритм который заставляет скелет подниматься с позиции лежа. И что? До роботостроения такого уровня в массовом производстве мы еще не дошли, ибо слишком дорого, у игр и без RL нормальный AI (ну как нормальный: на принципе сила есть — ума не надо), в трейдинге он тоже бесполезен.

Все на благотворительности работает: тот же Open AI. Покажите реальную бизнес возможность и инвестиции сразу поплывут миллионами, а там глы и муравьев начнут имитировать.

 **DesertFlow** сегодня в 14:29    

Почему до роботостроения такого уровня не дошли? Бесколлекторные электромоторы дешевы и просты в конструкции, управляются электронно ESC. Есть проблема в редукторах — там нужны огромные передаточные числа, с которым с высокой точностью справляются только волновые редукторы (применяемые массово в промышленных роботах). Но для бытового уровня и планетарных должно хватать.

По аккумуляторам — человек тратит примерно 100 Вт на тепло, 50-100 Вт при ходьбе, и около 250 Вт при беге. Нет особых причин, чтобы робот с мотором с КПД 95% должен тратить больше (а вот с редукторами хуже, хотя те же волновые до 99% бывают, т.к. являются разновидностью подшипника). 100-200 Вт/час это всего лишь 1 кг литиевых аккумуляторов. Не так уж много и не так уж дорого.

Все упирается именно в программную оболочку, а не в железо. И конкретно — в Reinforcement Learning.

Посудите сами: зрительная система, аналогичная человеческой — 50-150 млн параметров, глубина сети 20-1000 слоев. Языковая модель ~100 м параметров, глубина — десятки слоев. Лучшие образцы Reinforcement Learning — 2 слоя по 128 нейронов. Seriously? Самая большая сеть, обученной обошлось в десятки млн долларов — OpenAI Five — один слой LSTM в 1024 нейрона. Seriously?

 **arikshtein** сегодня в 14:50    

Дело не в алгоритмах, дело в железе (не буквально). Вся область RL сейчас — как обычный CNN в 80ых: концепт придумали, но реализовать могли очевидно из за чего. Алгоритмы и методы сами по себе не плохи, масштаб не тот. И если прицепить оптимизированный квантовый комп с парой сотен кубитов, то можно за те же 2-3 дня натренировать что угодно.

НО то же применимо и к Image Processing, Signal Processing: все они имеют огромный потенциал расти с расширением вычислительной мощности железа, но и сегодня на них можно хорошенько подзаработать (0.014\$ за фотографию по ценникам Гугла). От того и инвестиции постоянные, и алгоритмы, хайп и все такое.

А квантовыми компаниями сейчас кроме всяких университетов только IBM занимается ито чисто для маркетинга.

То есть либо на чем либо можно подзаработать *сейчас* — значит хайп, обсуждения, инвестиции, продвижение, либо «когда-нибудь», и всем блевать. Никто не покупал биткоины пока они за день не выросли в 3000%.

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

- Сутки
- Неделя
- Месяц

Юбилейный Android 10 (Q). Что известно уже сейчас?

+51 44,9k 29 81

Как научить людей использовать Git

+40 29,7k 453 152

Guix — самая продвинутая операционная система

+37 28,9k 107 56

Microsoft прекращает поддержку Windows 10 Mobile

+43 24,6k 23 140

Остров Пикю: как из базальтового ада сделали уютный Туссент

+87 26,5k 60 49

Аккаунт	Разделы	Информация	Услуги	Приложения
Войти	Публикации	Правила	Реклама	<div></div> <div></div>
Регистрация	Хабы	Помощь	Тарифы	
	Компании	Документация	Контент	
	Пользователи	Соглашение	Семинары	
	Песочница	Конфиденциальность		