Нейро-динамическое программирование автономных агентов

Сергей А. Терехов
Лаборатория искусственных нейронных сетей *ALIFE*
г. Троицк, Московская обл.

В лекции рассматриваются задачи многошаговой оптимизации и управления сложными динамическими системами. Обсуждаются постановки типовых задач и особенности алгоритмов нейродинамического программирования, использующих для аппроксимаций функций оптимального поведения аппарат искусственных нейронных сетей. Показана связь задачи целевой адаптации автономного агента при обучении с подкреплением и задачи прогностического оптимального управления. Приведены примеры приложений методов нейродинамического программирования.

This lecture is devoted to multistage optimization methods in the control of complex dynamic systems. Statements of typical control tasks are discussed. The key technology - neuro-dynamics programming method – is based on the approximation of cost-to-go function by artificial neural networks. Examples show the relation between reinforced adaptation of autonomous agents and predictive control problems.

ВВЕДЕНИЕ: ПРИМЕРЫ СИСТЕМ С ДИНАМИЧЕСКИ-ОПТИМАЛЬНЫМ УПРАВЛЕНИЕМ 1
КЛАССИЧЕСКАЯ ЗАДАЧА ОПТИМИЗАЦИИ С ОГРАНИЧЕНИЯМИ 3
УПРАВЛЯЕМАЯ ОПТИМАЛЬНАЯ ДИНАМИКА 4
ПРОГРАММНОЕ УПРАВЛЕНИЕ 4
ПРОГНОСТИЧЕСКОЕ УПРАВЛЕНИЕ НА ОСНОВЕ НЕЙРОСЕТЕВОЙ МОДЕЛИ СИСТЕМЫ 5
НЕЙРО-ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ. 7
МАРКОВСКИЙ ПРОЦЕСС ПРИНЯТИЯ РЕШЕНИЯ 8
УРАВНЕНИЕ БЕЛЛМАНА 11
ИГРОВАЯ СТРАТЕГИЯ И ОЦЕНКА ПОЗИЦИИ 12
ИТЕРАЦИИ В УРАВНЕНИЯХ БЕЛЛМАНА 13
НЕЙРОСЕТЕВАЯ АППРОКСИМАЦИЯ ФУНКЦИИ ЦЕННОСТИ 13
АЛГОРИТМ SARSA И Q-ОБУЧЕНИЕ 16
ПРИМЕРЫ ПРИКЛАДНЫХ РАЗРАБОТОК 17
ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ ПОРТФЕЛЕМ ФИНАНСОВЫХ АКТИВОВ 17
ДИНАМИЧЕСКОЕ РЕЗЕРВИРОВАНИЕ КАНАЛОВ СЕТИ СОТОВОЙ СВЯЗИ. 18
ИТОГИ 19
БЛАГОДАРНОСТИ 19
ЛИТЕРАТУРА 19

Введение: Примеры систем с динамически-оптимальным управлением

Рациональная деятельность по ведению хозяйства, использованию ресурсов, оптимальной организации деловой активности является основной отличительной чертой устройства современной экономики и управления. Формальная сторона проблемы

сводится к определению, в некотором смысле, наилучшего, оптимального варианта решения в текущих условиях.

В последние десятилетия сформулированы практически важные постановки задач поиска оптимума, а также предложены базовые алгоритмы для их решения. Наибольший прогресс в математических методах достигнут при рассмотрении статических задач, в которых решение принимается “одномоментно”, и оптимизируемой функцией является оценка последствий выполнения такого решения.

Однако, в условиях ускоряющегося темпа деловой активности и усложнения режимов конкуренции, лицо, принимающее решения все чаще сталкивается с необходимостью поиска оптимума в многошаговых задачах.

Предметом рассмотрения в данной лекции являются системы, в которых решение принимается поэтапно. При этом оптимальным должен быть весь многоэтапный процесс смены состояний системы.

Приведем примеры многоэтапных задач.

1. Менеджмент проекта. Управление ресурсами производственного или исследовательского проекта предполагает принятие серии последовательных решений. Способ организации управления на основе многошаговых решений называют “real options”.
2. Управление оптовыми закупками в торговых сетях (супермаркетах, специализированных магазинах), сетях быстрого питания, сетях косметических салонов и др. Определение объема и сроков поставок партий товаров на каждый момент времени должно учитывать будущие потребности и дифференцированные риски от нехватки или переизбытка товара. Управляющие решения (закупки) проводятся периодически, и процесс их оптимальной выработки является многошаговым.
3. Управление динамикой инвестиционного портфеля. Регулярное участие в торгах и реструктуризация состава активов портфеля позволяют рассматривать задачу управления им, как многошаговую. Управляющие воздействия состоят в покупке и продаже некоторых активов портфеля с целью достижения выбранного критерия эффективности (например, обеспечение максимальной доходности на указанном масштабе времени, при заданном уровне риска).
4. Динамическое планирование загрузки каналов сети сотовой связи [1]. Цель состоит в обеспечении качественного сервиса для максимального числа пользователей при заданном диапазоне выделенных частот. Сложность состоит в необходимости резервирования части ресурсов для обеспечения непрерывности связи при перемещении абонента из зоны одной соты к другой. При этом слишком большой резерв равносилен большому числу отказов новым абонентам при новых вызовах (т.е. блокировок). Задача решается многократно, в реальном времени с учетом текущей и прогнозируемой динамики запросов к сети.
5. Поиск информации в распределенной сети. Маршрутизация запросов в распределенном хранилище информации представляет собой задачу динамической оптимизации. Здесь потенциальный выигрыш от расширения масштаба поиска в сети противоречит готовности выполнять новые поисковые запросы. Одновременно с задачей поиска может решаться задача динамического оптимального размещения информации в сети.
6. Динамическое составление расписаний. Одно из актуальных применений – динамическое назначение пилотов на рейсы в крупных авиакомпаниях. Задача содержит сложные динамические ограничения, вызванные фактическим выполнением воздушных перевозок.

7. Широкий круг “проблемных” задач, таких, как обучение в многошаговых играх, адаптивное управление в распределенных и автономных системах, активное обучение (обучение агента, сопровождающееся активным отбором информации для дальнейшего обучения), адаптация робота в сложной окружающей среде.

Все перечисленные задачи обладают одной общей фундаментальной сложностью – оптимальный на данном шаге вклад в функцию ценности может соответствовать решению, далекому от оптимума для всего многошагового процесса в целом.

Особенно четко это наблюдается в прототипических задачах поиска выхода их лабиринта: глобально оптимальное решение включает локальные отрезки, на которых происходит фактическое удаление от точки выхода. Следствием этого феномена в общественной жизни являются, например, факты “не популярных” и, на первый взгляд, неоптимальных решений со стороны правительств, при выходе их стран из кризисов.

Одним из современных подходов к многошаговым проблемам служат приближенные методы поиска адаптационной стратегии, включающие аппроксимации функций оптимального поведения искусственными нейронными сетями. Общее семейство таких нейросетевых методов получило специальное название – нейродинамическое программирование.

Исследователи в области адаптации автономных агентских систем [2] в сложно устроенном окружении заинтересованы в методах нейродинамического программирования, поскольку эти методы не требуют построения прямых моделей среды, и позволяют агентам начинать действовать, не дожидаясь завершения обучения.

Дальнейшее изложение лекции построено по следующему плану. В начале мы кратко остановимся на классической постановке задачи оптимизации с ограничениями и перейдем к задаче оптимального управления системой, модель которой известна. Далее будет рассмотрена задача управления в условиях, когда модель системы недоступна. Затем мы откажемся от требования детерминированности системы и перейдем к Марковскому процессу принятия решения для агента во внешней среде. Будут сформулированы постановки задач динамического программирования и уравнение Беллмана.

Оставшаяся часть лекции обозревает методы нейродинамического программирования. Отдельно особое внимание будут обращено отличию задачи обучения нейронной сети на основе локальных подкреплений (reinforcement learning) от обычной задачи обучения с учителем. В завершение будут более подробно рассмотрены некоторые приложения.

Классическая задача оптимизации с ограничениями

Пусть исследуемая система описывается набором (вектором) переменных $x = \{s, a\}$, включающим переменные состояния s и переменные, описывающие возможные целенаправленные изменения (управления) в системе. Пусть далее определена функция ценности (цели) $F(x)$, которую желательно максимизировать.

В классической постановке эта оптимизация целиком выполняется во временной точке, без учета характера последующей динамики системы. В результате применения управлений a система сразу переходит в окончательное состояние $s(a)$, которое и оценивается целевой функцией. Задача оптимизации может формулироваться, например, следующим способом[1]:

Найти a^*, *такое, что*

[1] Литература по методам оптимизации крайне обширна. См. например [3-5]

$$a^* = \arg\max_{a \in A} F\left(s\left(a\right), a\right),$$

где A *- множество допустимых управлений, определяемое ограничениями*

$$g\left(s\left(a\right), a\right) = 0$$
$$f\left(s\left(a\right), a\right) \le 0$$

Постановки задач такого типа оказываются весьма содержательными на практике. Достаточно отметить, что даже "простейший" вариант этой задачи, в котором все функции предполагаются линейными, достаточен для формулирования большинства предметных задач исследования операций. Обзор тематики задач оптимизации выходит за рамки данной лекции. Качественно представление о современном состоянии в этой области можно получить на сайте *optimization-online.org*.

При описании задачи оптимизации предполагалось, что система мгновенно переходит в конечное состояние исключительно вследствие управляющих воздействий. Для задач, в которых существенна динамика системы, это является серьезным упрощением. Отказ от него приводит к проблеме поиска оптимальных динамических траекторий, которая относятся к классу задач оптимального управления.

Управляемая оптимальная динамика

Поиск оптимального управления находится на следующей ступеньке по сложности, т.к. в задаче оптимизации неизвестным являлся конечный набор чисел – оптимальных значений переменных, а здесь мы имеем дело с неизвестной *функцией* – законом управления. Поэтому иногда задачу управления называют задаче оптимизации в "бесконечномерном" пространстве.

Программное управление

Относительно просто может быть сформулирована так называемая задача *программного* управления. В ней предполагается, что управляющие сигналы воздействуют на систему на всем протяжении процесса, вплоть до достижения конечного состояния, однако эти сигналы не зависят от фактической траектории управляемой системы. Автомат программируется на следование некоторой, заранее синтезированной программе, которая выполняется независимо от того, что происходит с системой. Другими словами, система *не наблюдается* агентом-автоматом.

Математически задача программного управления формулируется, как задача поиска минимума функционала, который часто предполагается аддитивным:

Найти $a^*\left(t\right)$*, такое, что*

$$\int_0^T F\left(s^*\left(t\right), a^*\left(t\right)\right) \cdot dt + F_{t=T}\left(s^*\left(T\right)\right) \to \max,$$

где $a^* \in A$ *- множество допустимых управлений, определяемое ограничениями на всей траектории*

$$g\left(s\left(a\left(t\right)\right), a\left(t\right)\right) = 0$$
$$f\left(s\left(a\left(t\right)\right), a\left(t\right)\right) \le 0$$

В выражении для оптимизируемого функционала выделен член, отвечающий оценке конечного (терминального) состояния системы. В приложениях [6] рассматриваются также различные упрощения в постановке этой задачи (например, квадратичные функционалы и линейные ограничения).

Отметим важную специфику задачи оптимального управления, которая сохранится в дальнейшем при постановке других задач на протяжении всей лекции. Изменение функции управления влечет за собой изменение траектории системы $s\left(a\left(t\right)\right)$, поэтому и оптимальная траектория, и оптимальное управление должны определяться *одновременно*. Итерационное варьирование траектории или управления при поиске оптимума может быть затруднено, поскольку в отсутствии решения не ясно, какие вариации являются допустимыми.

Для численного решения задачи программного управления класс управлений ограничивают параметрическим семейством функций, и сводят задачу к конечномерной задаче поиска оптимальных значений параметров (т.е. к уже рассмотренной задаче оптимизации). Часто ограничиваются выбором кусочно-постоянных управлений или гладкими сплайнами.

Представляет большой интерес возможность использования нейронной сети в качестве пробной функции, т.к. она обладает свойствами контролируемой гладкости и ограниченными производными.

Прогностическое управление на основе нейросетевой модели системы

Программное управление является приемлемым подходом во многих прикладных ситуациях. На этом принципе основаны, например, простые металлорежущие станки или бытовые стиральные машины. В этих случаях система полностью подчинена управлению[2].

В более сложных случаях поведение системы под воздействием программного управления может оказаться *неустойчивым*, и для его стабилизации необходима возможность наблюдения за системой (измерения состояния) и использования этой *обратной связи* в законе управления. Оптимальное управление с обратной связью так же, как и задачи оптимизации, имеет свою глубокую историю и многообразие постановок задач и методик их решения [7].

Здесь будет рассмотрен только один из подходов этого класса – прогностическое управление на основе модели (MPC-Model Predictive Control). Метод MPC особенно ценен для тематики этой лекции по трем основным причинам:

- Метод основан в явной форме на многошаговом поиске решения
- В качестве базового инструмента может использоваться искусственная нейронная сеть, что повышает технологичность подхода
- Метод разработан до прямых промышленных приложений с большим коммерческим эффектом.

Алгоритмы MPC используют численную оптимизацию для нахождения оптимального управления на некотором временном горизонте в будущем, основываясь на модели процесса. В современных приложениях, таких, как очистка нефти, нефтехимия, целлюлозная и пищевая промышленность, отклик управляемой системы часто является существенно нелинейным. Это, наряду с более высокими ограничениями на качество продукта и увеличивающимися требованиями производительности, экологическими требованиями и, не в последнюю очередь, экономическими соображениями требует, чтобы система эксплуатировалась в режимах, близких к технологически допустимым границам. Это, в свою очередь требует разработки точных моделей не только для отклика системы, но и для ограничений.

Нейронные сети де-факто стали очень популярной методикой для моделирования нелинейной многомерной динамики систем, вследствие их уникальных аппроксимационных свойств [8].

[2] В более современных устройствах имеются аварийные блокировки – пример управления по обратной связи.

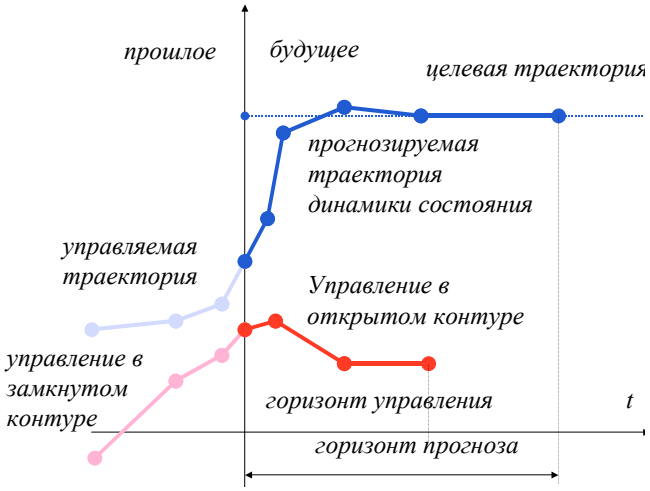

Рис. 1 Прогностическое управление на основе модели динамики системы

MPC включает следующие шаги:

- Нейронная сеть обучается на исторических данных и данных, полученных в результате испытаний управляемой системы в условиях программного управления. После обучения, нейронная сеть может использоваться для предсказания динамики системы при заданном законе управления на N последовательных шагов по времени (горизонт прогнозирования).
- На каждом шаге управления новое состояние системы измеряется в петле обратной связи.
- Управляющие воздействия на последующих M шагах рассматриваются как неизвестные переменные[3]. Набор этих переменных вместе с нейросетевой моделью отклика системы синтезируют теоретическую траекторию изменения состояния системы. Оптимальные значения управляющих переменных далее определяются итерационно путем минимизацией стоимости прогнозируемой траектории.
- Только *первое* управляющее действие (из N последовательных значений управления) фактически применяется к управляемой системе. Далее алгоритм тогда повторяется для следующего шага.

В рамках MPC-подхода могут быть рассмотрены две важные прикладные проблемы управления:

- *Адаптивное* управление, при котором состояние управляемой системы следует заданной траектории. Целевая функция штрафует отклонение теоретической траектории от желательной траектории.
- *Оптимальное* управление – нахождение траектории системы, ведущей к указанному множеству конечных состояний, вдоль которой стоимость управления минимальна. Стоимость может включать как прямые материальные затраты, так и косвенные требования к управлению (технологическое качество кривой управления, стоимость модификации управляющей системы, робастность, и т.д.).

В работе [9] приведено описание одной из промышленных разработок – MPC контроллера фирмы Pavilion Technologies (США). Особенно успешны применения таких контроллеров в задаче оптимально управления химическими реакторами по получению полипропилена и других полимеров. Контроллер применяется в около 50-ти других прикладных задач.

MPC-подход не свободен от недостатков. В частности, могут возникать проблемы с устойчивостью, вызванные конечностью горизонта управления. Конечно, они стоят менее остро, чем при применении обычных линейных систем управления к существенно нелинейной системе, работающей в нескольких операционных режимах.

[3] Обычно горизонт поиска управления M выбирается меньше горизонта прогнозирования N, с целью упрощения решения задачи оптимизации. За горизонтом управление полагается постоянным.

Требует также специального обоснования адекватность нейросетевой модели в условиях нехватки обучающих данных о системе. Однако, для этой проблемы в современной нейроинформатике уже накоплен мощный арсенал средств.

Нейро-динамическое программирование.

Применение описанного в предыдущем разделе подхода к оптимизации динамики системы ограниченно несколькими обстоятельствами:

- Необходимость наличия адекватной математической модели управляемого объекта. В некоторых важных приложениях математической модели системы может и не быть, либо ее использование требует неприемлемых вычислительных ресурсов. Для наблюдения может быть доступна лишь локальная оценка управляющего решения, так называемое reinforcement или "подкрепление на шаге". Особенно ярко эта ситуация проявляется в автономных системах (роботы, автопилоты и т.д.). Поиск оптимального управления в них сопровождается экспериментальным исследованием окружения. При рассмотрении таких систем лучше говорить о задаче оптимальной адаптации.
- Поиск многошагового оптимального управления путем прямой оптимизации возможен лишь в задачах с небольшим числом переменных и временных шагов. При увеличении числа состояний и возможных управлений сложность задачи оптимизации резко возрастает. Это явление названо[4] Р. Беллманом "проклятием размерности" [10].
- В общем случае объект управления демонстрирует вероятностное поведение, которое может быть вызвано как неполнотой описания сложной системы, так и объективными характером процессов. Возможность использования вероятностного описания сложной системы и соответствующий математический аппарат – нейросетевые аппроксимации плотности вероятности – обсуждались в одной из предыдущих лекций автора [11]. Для представления вероятности перехода из состояния в состояние при заданном управлении могут использоваться Байесовы сети (см., например, лекцию [12]).

Заметим, что в подходе MPC задача оптимизации на каждом этапе решается заново, найденные оптимальные управления для прошлых состояний системы никак не используются. Выдающейся заслугой Р. Беллмана явилось открытие способа решения многошаговой задачи оптимизации, при котором :

эффективно используются ранее полученные оценки состояний системы

большая задача разбита на серию более простых одношаговых задач, решаемых на каждом этапе

Метод получил название "динамического программирования" [10]. Более подробное изложение его приведено в следующем разделе, а здесь мы проиллюстрируем основную идею на простом примере максимизации многомерной аддитивной функции с ограничениями, предложенном самим Р. Беллманом.

Решить задачу оптимизации

$$F\left(x_1, x_2, ..., x_N\right) = \sum_{i=1}^{N} g_i\left(x_i\right) \to \max,$$

при ограничениях

$$x_i \geq 0,\ \sum_{i=1}^{N} x_i = c$$

Здесь функции g непрерывны при неотрицательных x. Так как максимум F зависит только от числа N элементов в сумме (при сохранении порядка) и величины c,

[4] Понятие, введенное Р. Беллманом для описания катастрофического роста числа состояний при масштабировании многомерной задачи.

обозначим его за $f_N(c)$. Легко заметить, что для набора введенных функций f справедливо рекуррентное соотношение:

$$f_N(c) = \max_{0 \le x \le c} \left[g_N(x) + f_{N-1}(c-x) \right]$$

поскольку, по определению, величина f_{N-1} описывает оптимальное значение для суммы из (*N*-1) упорядоченных слагаемых, а аргументы x и $(c-x)$ гарантируют выполнение ограничений задачи. Рекурсия замыкается соотношением

$$f_1(c) = g_1(c)$$

Таким образом, исходная многомерная задача оптимизации свелась к последовательности одномерных задач, которые можно рассматривать как многошаговый процесс поиска оптимума. В некоторых приложениях индексу *N* придается смысл дискретного времени.

Марковский процесс принятия решения

Рассмотренная выше многошаговая задача оптимизации может быть положена в основу более широкого класса задач оптимального управления и поведения.

До этого момента в лекции неявно предполагалось, что управляемая система представляет собой некоторый технический объект, который рационально используется путем применения к нему управляющих воздействий. Оптимальное управление может быть специально синтезировано для этой системы, экспериментально отработано и внедрено. Модификация системы управления в технических приложениях – относительно редкий процесс, который рассматривается, скорее, как исключение, нежели правило.

В дальнейшем изложении мы перейдем к другой метафоре, которая служит мостиком от теории оптимального управления к теории адаптивного обучения в изменяющихся внешних условиях. Примером служит построение оптимальных стратегий поведения *автономного* агента, самостоятельно принимающего решения. Агент находится в недетерминированной среде, модель которой ему, вообще говоря, недоступна.

Эта задача в сравнении с рассмотренными ранее в лекции постановками еще на ступеньку сложнее. Чтобы перейти к ней введем сначала основные понятия, сопровождающие процесс принятия решения.

Под *агентом* будем понимать, автономную сущность (компьютерную программу, робототехническую систему, или некоторую формальную систему - предметный смысл может уточняться в контексте), которая имеет возможность получать информацию о своем окружении и действовать, изменяя состояние (свое и окружения). Общая структура взаимодействия агента и окружения приведена на Рис.2

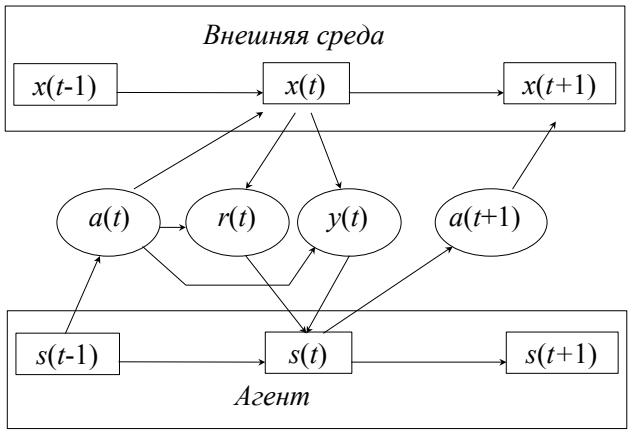

Рис 2. Структура взаимодействия агента со средой

Все временные изменения происходят в дискретные моменты времени[5]. Внешняя среда описывается состоянием среды $x(t)$, которое вероятностным образом изменяется во времени. Будем предполагать, что функция плотности распределения условной вероятности зависит только от состояния и действия агента в предыдущий момент времени:

$$x(t) \sim P\left(x(t) \middle| x(t-1), a(t-1)\right)$$

В общем случае агент не может наблюдать все компоненты состояния среды (например, в задаче ориентации в лабиринте агент наблюдает лишь непосредственное окружение).

Наблюдаемая функция $y(t)$ также описывается вероятностной моделью:

$$y(t) \sim P\left(y(t) \middle| x(t), a(t)\right)$$

Кроме наблюдения $y(t)$ агент получает локальное подкрепление[6] (reward)

$$r(t) = E\left(R(t) \middle| x(t), a(t)\right)$$

То, что агент наблюдает, зависит от того, что он делает. Таким образом, учитывается, что восприятие агента является активным процессом.

Сам агент обладает памятью о своем текущем состоянии. Новое состояние $s(t)$ он волен менять по некоторому закону:

$$s(t) = f\left(s(t-1), y(t), r(t), a(t)\right)$$

В новом состоянии агент вырабатывает управляющее действие, придерживаясь своей стратегии $\pi(s)$:

$$a(t) = \pi\left(s(t)\right)$$

[5] Рассматриваются также и континуальные постановки задач, обычно это приводит к серьезным математическим усложнениям.
[6] Или наказание ☺

Целью агента является максимизация суммарного поощрения в процессе адаптации в среде путем научения некоторой оптимальной стратегии и функции изменения состояния:

$$J(x,a) = E\left(\sum_t \gamma^t r(t)\right) \to \max$$

В последнем выражении использована дисконтированная функция поощрения, что гарантирует конечность суммы даже в случае, если агент адаптируется неопределенно долго. Возможна также другая постановка, в которой в системе имеются выделенные, достижимые с конечной вероятностью *терминальные* состояния, и процесс взаимодействия со средой разбивается на отдельные временные эпохи.

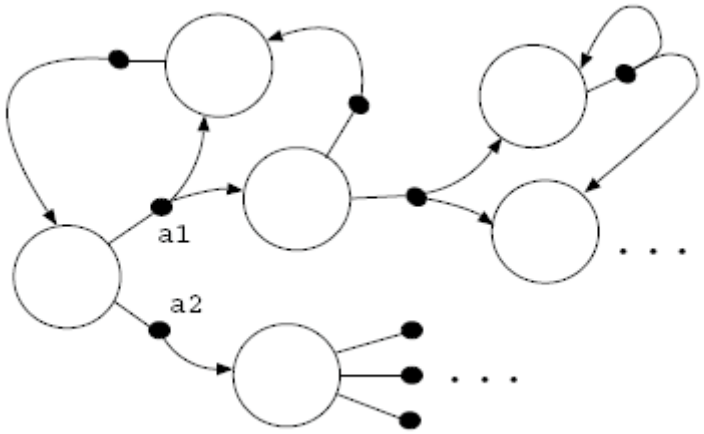

Рис 3. Граф Марковского процесса решений.
Кружки соответствуют состояниям, точки – возможным решениям.

Хорошей метафорой для класса рассматриваемых задач может служить задача адаптации выделенного агента в лабиринте, в котором также действуют другие агенты, со своими целями. Агент должен найти выход из лабиринта, не столкнувшись по пути с другими агентами. *Клеточный мир* лабиринта стационарен, однако наличие множества участников, перемещающихся в лабиринте по неизвестным нашему агенту законам, делает задачу его перемещения вероятностной и нестационарной.

На языке дискретных клеточных миров (и в пространстве, и во времени) может формулироваться весьма широкий класс задач логистики, планирования, и исследования операций[7].

Для простоты, будем рассматривать ситуацию, когда взаимодействие агента со средой разбивается на множество повторяющихся эпизодов. Часто задачу ограничивают специальным случаем полной наблюдаемости $y(t) = x(t)$, как, например, в некоторых игровых постановках (поле игры доступно полностью). Тогда необходимость в выделении собственного состояния агента s отпадает, и можно использовать термин "состояние" для описания состояния совокупной системы "агент + среда".

Агент должен научиться оптимальному поведению только на основе элементарных поощрений, возникающих в актах взаимодействия со средой, без внешнего учителя и без возможности непосредственного наблюдения отдаленных последствий своих действий. Другими словами, агент имеет представление о том, *что* нужно сделать (*что* будет

[7] Язык клеточных миров оказывается еще более широким, поскольку на нем могут рассматриваться задачи оптимизации для двух и более агентов, относящиеся к *теории игр*.

оценено высоким уровнем поощрения), но не *как* этого достичь. Это является принципиальным моментом в проблеме, и порождает следующую проблему выбора (дилемму рациональности-иррациональности):

- В процессе поиска оптимума агент должен действовать максимально целесообразно в соответствии с текущим пониманием ситуации
- Вместе с тем, агент должен исследовать совершенно новые варианты поведения, в противном случае он остановится в локальном минимуме.

В случае, когда состояние системы содержит всю существенную информацию для описания дальнейшей динамики, задача сводится к Марковскому процессу, в котором вероятности переходов зависят от управляющих переменных (действий агента). Более строго, вероятности переходов между состояниями должны удовлетворять условию:

$$P\left(s\left(t+1\right)=s',r\left(t+1\right)=r\middle|s\left(t\right),a\left(t\right),s\left(t-1\right),a\left(t-1\right),\ldots s\left(0\right),a\left(0\right)\right)=$$
$$=P\left(s\left(t+1\right)=s',r\left(t+1\right)=r\middle|s\left(t\right),a\left(t\right)\right)$$

для всех состояний и историй системы. На практике, требование марковости часто можно выдержать, если расширить размерность пространства состояний путем введения дополнительной памяти об истории динамики системы.

Процессы, в которых вероятности переходов зависят от управляющих переменных, принято называть Марковскими процессами решений (MDP – Markov Decision Process). Далее будет показано, что нахождение оптимальных стратегий для таких процессов может описываться компактными уравнениями, решаемыми итерационными алгоритмами.

Уравнение Беллмана

Введем обозначения

$T\left(s,a,s'\right)=P\left[S\left(t+1\right)=s'\middle|S\left(t\right)=s,A\left(t\right)=a\right]$ - вероятности переходов между состояниями. В случае детерминированных переходов будем говорить о модели смены состояний $s'=f\left(s,a\right)$.

$r\left(s,a,s'\right)=E\left[R\left(t+1\right)\middle|S\left(t\right)=s,A\left(t\right)=a,S\left(t+1\right)=s'\right]$ - функция поощрения[8] на шаге.

Обозначим далее $V^{\pi}\left(s\right)$ *цену состояния* s - т.е. то суммарное поощрение, которое получит агент, следующий своей стратегии π, в эпизоде, начинающемся в состоянии s. Заметим, что цена состояния существенно зависит от стратегии:

$$V^{\pi}\left(s\right)=E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r\left(t+k+1\right)\middle|S\left(t\right)=s\right]$$

Максимально возможная цена состояния

$$V^{*}\left(s\right)=\max_{\pi}V^{\pi}\left(s\right)$$

соответствует оптимальной стратегии π^{*}. Аналогично цене состояния введем также *цену действия* $Q^{\pi}\left(s,a\right)$ - суммарное поощрение агента, предпринявшего в состоянии s управляющее решение a, и далее следующего избранной стратегии.

[8] Поскольку целью агента является максимум математического ожидания суммы поощрений, удобно перейти от полного вероятностного рассмотрения стохастического процесса поощрения к описанию достаточной статистики – матожидания поощрения на шаге.

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r(t+k+1)\middle| S(t)=s, A(t)=a\right]$$

С переходом от представления в виде оценки состояния к цене хода в данном состоянии, задача оптимизации на каждом шаге упрощается, однако число переменных задачи возрастает.

Цены состояния и действия связаны соотношением, вытекающим из определений:

$$Q^{\pi}(s,a) = \sum_{s'} T(s,a,s')\cdot\left[r(s,a,s') + \gamma V^{\pi}(s')\right]$$

Оптимальная стратегия соответствует максимальному поощрению:

$$V^{*}(s) = \max_{a} Q^{*}(s,a) = \max_{a}\sum_{s'} T(s,a,s')\left[r(s,a,s') + \gamma V^{*}(s')\right]$$

Аналогичное соотношение имеет место и для цены действия:

$$Q^{*}(s,a) = \sum_{s'} T(s,a,s')\left[r(s,a,s') + \gamma \max_{a'} Q^{*}(s',a')\right]$$

Последние два уравнения играют фундаментальную роль в теории Марковских процессов решений и динамическом программировании. Они носят название *уравнений Беллмана* (для оценок состояния и оценки действия). Решение уравнения Беллмана дает оптимальное распределение цены по всему пространству состояний задачи, и тем самым, решает сразу семейство задач оптимизации, отвечающих различным начальным состояниям.

Игровая стратегия и оценка позиции

Уравнение Беллмана является наиболее общим соотношением для оценки позиции в играх, обладающих марковским свойством[9]. Решение уравнения Беллмана, формально, дает оценку каждого состояния (т.е. игровой позиции), поэтому оптимальная стратегия игры легко определяется – на каждом шаге следует перемещаться в состояние с более высокой ценой, в качестве которой может выступать, например, вероятность выигрыша. Выбранное действие в игровых задачах часто называют *ходом* игрока.

Буквальное описание состояния, как совокупности расположения всех фигур, наталкивается на большие вычислительные сложности. На практике выбираются более компактные представления [13], агрегирующие большое число элементарных состояний (как-то, позиция фигуры в центре, число угроз и защит фигур, и т.д.). В свою очередь, возникают трудности при выборе представления – противоречие между подробностью описания и "проклятием размерности". Сама проблема извлечения существенных *черт* для данной задачи является крайне сложной, и должна рассматриваться в совокупности с задачей оптимизации стратегии. Тем не менее, переход к оцениванию состояния, вместо необходимости перебора и оценки каждой из возможный стратегий, является существенным шагом к упрощению задачи.

[9] Шахматы, например, обладают таким свойством с точностью до отслеживания возможности рокировки короля. Правила запрещают рокировку, если король двигался на более *ранних* ходах в партии. Однако, в случае шахмат, эту проблему легко обойти, если расширить понятие состояния, введя дополнительную логическую переменную "двигался ли король?". В игре *Го* ситуация более сложная вследствие правила *Ко*, запрещающего повтор расположения камней на игровой доске. Поэтому набор допустимых ходов в некоторой позиции, формально, зависит от всей предыстории партии.

Итерации в уравнениях Беллмана

Одним из прямых подходов к поиску оптимальной адаптации в (конечных и дискретных) игровых позициях служит итерационное вычисление оценки позиции (value iteration).

$$V_{j+1}^{*}(s) \leftarrow \max_{a} \sum_{s'} T(s,a,s') \left[r(s,a,s') + \gamma V_{j}^{*}(s') \right]$$

Сходимость алгоритма value iteration легко устанавливается для показателя дисконтирования γ, меньшего единицы – итерационное отображение является сжимающим. Если дисконтирование отсутствует, то для сходимости требуются дополнительные условия на матрицу вероятностей и на достижимость состояний (т.е. при неограниченных итерациях каждое состояние посещается бесконечное число раз). Обсуждение сходимости итераций в случае непрерывных переменных и управлениях выходит за рамки этой лекции[10].

Другой интересной задачей является оценивание некоторой определенной стратегии π поведения агента (policy evaluation). Уравнение для цены состояния относительно заданной стратегии имеет вид:

$$V_{j+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s') \left[r(s,\pi(s),s') + \gamma V_{j}^{\pi}(s') \right]$$

Здесь состояние s' находится на допустимой траектории, ведущей из состояния s при применении действия $\pi(s)$ в данной стратегии. Если получена оценка состояний в выбранной стратегии, то ее можно использовать для улучшения самой стратегии (policy iteration). А именно, гарантированно более ценной стратегией будет стратегия, *жадная* по отношению к данной:

$$\pi_{j+1}(s) \leftarrow \arg\max_{a} \sum_{s'} T(s,a,s') \left[r(s,a,s') + \gamma V_{j}^{\pi}(s') \right]$$

При разработке алгоритмов, шаги оценивания стратегии и ее уточнения могут чередоваться. В дискретном случае возникающие при прямом итерировании уравнения Беллмана трудности, в основном, связаны с практическим обеспечением сходимости за интересные времена вычислений. Неизвестные величины цены состояния могут представляться в виде векторов, а вероятности переходов – в виде матриц. В частности, соотношения для оценивания стратегии могут рассматриваться, как система линейных уравнений

$$\vec{V} = \vec{R}^{\pi} + \gamma T^{\pi} \vec{V}$$

а задача вычисления оценки позиции может быть сформулирована, как задача решения линейных неравенств типа

$$\vec{V} \le \vec{R} + \gamma T \vec{V}$$

с дополнительными требованиями максимальности оценок.

Нейросетевая аппроксимация функции ценности

Табличное представление цен действий и состояний задачи имеет естественные ограничения по масштабируемости задачи на большую размерность. В дискретных моделях, имеющихся в литературе, число измерений весьма редко превосходит дюжину. Выход состоит в поиске приближенных аппроксимаций функций задачи.

[10] Некоторые принципиально сложные в теории обучения с подкреплением вопросы рассматриваются в работе [14].

Имеются два важных условия, влияющих на эффективность выбираемых алгоритмов аппроксимации. Во-первых, алгоритм должен основываться на функциях, обладающих богатыми возможностями по аппроксимации, и во-вторых, для выбранного алгоритма должны существовать масштабируемые алгоритмы выбора параметров аппроксимаций. Эти два требования часто являются противоречащими друг другу – богатый набор аппроксимирующих функций обычно связан с большим числом свободных параметров.

Искусственные нейронные сети являются весьма естественным инструментом для приближенного представления оценок состояний при росте размерности. Однако в стандартной постановке задача аппроксимации нейронной сетью рассматривается, как задача обучения *с учителем*, т.е. с использованием обучающих пар "вектор аргументов – значение аппроксимируемой функции".

В задаче аппроксимации ценности состояния ($V(s)$ или $Q(s,a)$) "правильные" обучающие значения отсутствуют – они являются решением задачи оптимизации, которую как раз предстоит решить. Это обстоятельство вносит новый, принципиальный уровень сложности в постановку задачи, а именно, оптимальное решение задачи, содержащей приближения в оптимизируемом функционале на классе выбранных аппроксимаций *не обязано существовать*, а если решение приближенной задачи все-таки существует, оно *не обязано приближать* собой решение исходной *точной* задачи.

На практике это приводит к тому, что приходится довольствоваться суб-оптимальными стратегиями, представленными аппроксимациями, и эти стратегии весьма расширительно трактуются, как приближенные решения задачи оптимальной адаптации (управления). Семейство приближенных алгоритмов, основанных на нейросетевых методах аппроксимации функций оценки состояний в многошаговых задачах принято называть *нейро-динамическим программированием* [15].

Рассмотрим вначале, как может быть построено обучение нейронной сети в случае задачи итерационного вычисления цены состояния. Алгоритм value iteration состоит в поиске неподвижной точки уравнения типа $V = g(V)$ методом простой итерации. Операция присваивания нового итерационного значения $V_{j+1} \leftarrow g(V_j)$ в случае нейросетевой аппроксимации не может быть применена непосредственно, и должна быть заменена некоторой операцией изменения весовых коэффициентов нейронной сети.

Простые варианты наталкиваются на методические сложности. Например, операцию присваивания можно было бы заменить одним шагом градиентного уменьшения ошибки

$$E(\vec{w}) = \tfrac{1}{2}\left(V_{\vec{w}}(s) - g(V)\right)^2$$

который приводит к следующим формулам обновления весов нейронной сети с некоторым шагом

$$\vec{w} \leftarrow \vec{w} + \eta\left(g(V) - V_{\vec{w}}(s)\right)\frac{\partial V_{\vec{w}}(s)}{\partial \vec{w}}$$

Хотя эти соотношения представляются весьма естественными, имеется несколько серьезных контрпримеров, указывающих на отсутствие сходимости такого алгоритма даже в случае простых линейных аппроксимаций [13]. Причина кроется в потере свойства безусловной сжимаемости отображения, в котором используется аппроксимация. Несмотря на то, что в литературе было предложено много практически полезных способов стабилизации итераций [16], общий вывод таков – локальные методы value iteration с аппроксимацией функции ценности, в общем случае, *не сходятся*.

Идея решения проблемы со сходимостью состоит в использовании для вычисления функции цены состояния *глобальной* информации о траектории из состояний, которые

посещались агентом. Семейство алгоритмов, основанных на этом подходе, получило название "методы разностей по времени" (temporal difference methods, $TD(\lambda)$) . В современном виде теория $TD(\lambda)$ сформулирована Sutton и Barto[1].

В дискретном случае алгоритмы с разностями по времени обходят (частично) также и другое ограничение методов value iteration, а именно, они являются online-алгоритмами и не требуют для новых приближений посещения всех состояний. Обучение происходит отдельными траекториями (эпизодами). В простейшей трактовке, если происходит изменение в оценке некоторого состояния, это изменение отражается и на предыдущих состояниях. Обновление оценки происходит синхронно.

Наиболее просто алгоритм формулируется в случае дискретных состояний. Предполагается, что если оценка текущего состояния после шага уточняется на величину δV , то оценка предыдущего состояния должна быть уточнена на $\lambda\delta V$, состояния перед ним – на $\lambda^2\delta V$ и т.д. Для накопления информации об истории состояний отводится специальная память (eligibility trace).

Алгоритм $TD(\lambda)$

Инициализировать $\vec{V} \leftarrow$ *любое*; $\vec{e} \leftarrow 0$

Для каждого эпизода

$s \leftarrow$ *случайное*

Цикл до конца эпизода

$s' \sim T(s, \pi(s), s')$

$\delta \leftarrow r(s, \pi(s)) + \gamma V(s') - V(s)$

$e(s) \leftarrow e(s) + 1$

$\vec{V} \leftarrow \vec{V} + \delta \cdot \vec{e}$

$\vec{e} \leftarrow \lambda\gamma\vec{e}$

$s \leftarrow s'$

В этом алгоритме используемая стратегия $\pi(s)$ может быть постоянной, в этом случае вычисляется функция оценки состояний относительно этой стратегии. Альтернативно, стратегия может быть выбрана *жадной* по отношению к текущей оценке V , в таком случае алгоритм оценивает значение оптимальной стратегии V^*. На практике сходящееся решение достигается для *ε-жадных* стратегий, в которых с вероятностью ε предпринимается случайное действие, а в остальных случаях – жадное (максимизирующее текущее значение V). Обычно стратегии класса ε-жадных ассоциируются с исследовательским поведением агента.

Приведенный алгоритм обобщается на вариант с нейросетевой аппроксимацией функции оценки. Шаги внутреннего цикла имеют вид

$$\delta \leftarrow r(s, \pi(s)) + \gamma V_{\vec{w}}(s') - V_{\vec{w}}(s)$$
$$\vec{e} \leftarrow \lambda\gamma\vec{e} + \partial V_{\vec{w}}(s) / \partial \vec{w}$$
$$\vec{w} \leftarrow \vec{w} + \eta \cdot \delta \cdot \vec{e}$$

Заметим, что вектор "следа памяти" $\vec{e}$ определен здесь для каждого параметра нейросетевой модели $\vec{w}$, а не для набора состояний (как в дискретном случае). К сожалению, точных результатов о сходимости алгоритмов $TD(\lambda)$ с аппроксимациями известно мало.

Алгоритм SARSA и Q-обучение

Часто на практике используются алгоритмы временных разностей, в которых аппроксимируется не ценность состояния $V(s)$, а цена действия агента $Q(s,a)$ в данном состоянии. Как уже отмечалось, функция Q позволяет обойтись без модели системы, т.к. оптимальное управление в каждом состоянии может быть найдено непосредственно:

$$a^* = \arg\max_a Q(s,a)$$

Можно выделить два варианта алгоритмов для нахождения $Q(s,a)$. В первом варианте при вычислении разности по времени используется текущее значение функции $Q(s,a)$, т.е. разности согласованы с оцениваемой стратегией (в англоязычной литературе используется термин *on-policy*).

$$Q(s(t),a(t)) \leftarrow Q(s(t),a(t)) + \alpha\left[r(t+1) + \gamma Q(s(t+1),a(t+1)) - Q(s(t),a(t))\right]$$

Алгоритм получил название SARSA (в соответствии с мнемоникой обозначений последовательности $s,a \to r \to s'a'$). Циклы обучения устроены по эпохам, на каждой пробной траектории величины управлений $a(s), a'(s')$ выбираются ε-жадными относительно текущей оценки Q.

Второй вариант, называемый Q-обучением (Q-learning) использует при вычислении разностей по времени *максимальную* текущую оценку Q в данном состоянии, независимо от текущей стратегии. Таким образом, при моделировании используется одна стратегия, а фактически оценивается другая (оптимальная относительно текущей оценки Q). Такие методы называют не согласованными (off-policy). Уравнения для обучения агента имеют вид:

$$Q(s(t),a(t)) \leftarrow Q(s(t),a(t)) + \alpha\left[r(t+1) + \gamma \max_{a'} Q(s(t+1),a') - Q(s(t),a(t))\right]$$

Сравнительная оценка этих методов приведена в таблице:

SARSA	Q-обучение
Обучение стратегии, наилучшей среди ε-жадных.	Обучение оптимальной стратегии
Высокая скорость обучения, т.к. принимаются во внимание “исследовательские” шаги (с вероятностью ε)	Высока вероятность попадания в локальный минимум.

Для обоих вариантов предложены модификации, учитывающие “следы памяти” (называемые, соответственно, $SARSA(\lambda)$ и $Q(\lambda)$) с нейросетевыми аппроксимациями.

Значительная исследовательская активность в области алгоритмов рассмотренных классов связана с поиском методов агрегации состояний в макро-состояния (существенные черты задачи), сокращающих пространство поиска задачи, а также с теоретическим обоснованием сходимости аппроксимационных алгоритмов в специальных условиях.

Примеры прикладных разработок

В завершение лекции остановимся на двух важных примерах технологий нейродинамического программирования: управление финансовым портфелем и оптимизация ресурсов в сетях сотовой связи.

Динамическое управление портфелем финансовых активов

В работе [17] описывается пример успешного использования методов многошагового обучения для задачи управления торговой системой. Рассмотрим простейший вариант, когда портфель состоит из одного актива[11] с ценой z_t, и автоматическая торговая система вырабатывает оптимальную последовательность решений $F_t \in \{-1,0,1\}$. Три варианта решений соответствуют покупке (long), пассивному ожиданию и, соответственно, продаже (short) актива. При этом, для простоты, будем считать, что объем каждой сделки фиксирован. Математически, речь идет об обмене одного актива портфеля на другой по текущей рыночной цене[12]. В момент свершения эта сделка не изменяет стоимость портфеля, однако она вредно или полезно сказывается на его цене в будущем.

Торговая система может следовать некоторой параметрической модели:

$$F_t = F\left(\theta_t; F_{-1}; \{z_t, z_{t-1}, ...; y_t, y_{t-1}, ...\}\right)$$

Здесь θ - обучаемые параметры, y_{t-k} - дополнительные внешние переменные в соответствующие моменты времени (например, значения биржевых индексов, цены базовых активов и др.). Соотношение для дохода, получаемого системой-трейдером за T шагов, имеет вид:

$$P_T = \sum_{t=1}^{T} R_t = v\sum_{t=1}^{T}\left\{r_t^f + F_{t-1}\left(r_t - r_t^f\right) - \delta\left|F_t - F_{t-1}\right|\right\}$$

Здесь v - объем сделки, δ - комиссия, $r_t = z_t - z_{t-1}$ -текущая доходность, величины с индексом f соответствуют доходности безрискового актива (например, государственной ценой бумаги). Возможны другие соотношения для дохода портфеля, учитывающие, например частичное реинвестирование прибыли [18].

Цель торговой системы может формулироваться, как достижение максимального дохода. На практике, в соответствии с представлениями в области финансов, принято оптимизировать величину отношения среднего дохода к его дисперсии (отношение Шарпа). Для целей применения методик многошаговой оптимизации в работе [17] был предложен дифференциальный вид отношения Шарпа.

Алгоритм был применен для торговли на среднемесячных значениях индекса S&P 500. В модели использовались дополнительно еще около 80 временных рядов (как макроэкономических, так и финансовых). Для стартового обучения трейдер-системы использовались исторические данные за 1950-1969 г. В игровом режиме система проверялась на 25-летнем промежутке с 1970 по 1994 г. Полученное соотношение Шарпа составило величину 0.83 в год. Для сравнения стратегия "купи и держи" дает на этом растущем рынке значение 0.34 в год, в качестве безрисковых активов рассматривались годовые векселя казначейства США (US Treasure Bills).

Авторы отмечают, что торговая система, основанная на алгоритмах нейро-динамического программирования, принимала весьма качественны решения в периоды

[11] Портфелем называется *подмножество* активов (например, ценных бумаг) из *множества* активов, оборачивающихся на рынке. Портфель логически дополняется еще одним активом – (свободными) деньгами.
[12] Обычно имеются еще издержки сделки.

нефтяного кризиса в 1974, финансовых проблем начала 80-х, рыночных поправок 1984 г. и обвала рынка в 1987 г.

Динамическое резервирование каналов сети сотовой связи.

В сетях сотовой связи остро стоит проблема распределения диапазонов частот для обеспечения максимального количества услуг в условиях стохастически изменяющихся потребностей. Рассмотрим модель сотовой сети.

Заданная географическая область разделяется непересекающиеся ячейки, и каждая ячейка обеспечивает услуги в пределах своих границ. Весь доступный диапазон частот разбит на каналы. Каждый канал может использоваться одновременно в нескольких ячейках, при условии, что эти ячейки географически далеки друг от друга (так, что нет интерференции). Когда пользователь запрашивает услугу, ему либо назначается один из свободных каналов, либо вызов блокируется. Также, при пресечении границ зоны, активный вызов передается от сотовой ячейки к соседней ячейке, где ему выделяется канал. Если такого канала нет, то сервис прерывается.

Таким образом, критерий оптимальности должен учитывать как сокращение числа блокировок, так и уменьшение вероятности прерывания сервиса.

Многие сотовые системы используют фиксированное распределение каналов. Более эффективной является стратегия динамического распределения, которая позволяет использовать в ячейке *любой* канал, который свободен и *в данный момент* не используется в соседних ячейках.

Сформулируем эту задачу в терминах многошаговой оптимизации [15]. Смена состояния происходит, когда канал освобождается (пользователь покидает ячейку), или когда новый пользователь запрашивает канал (новый вызов), либо пользователь переходит из ячейки в ячейку (поддержка вызова). Состояния в каждый момент описывается списком занятых каналов в каждой ячейке, а также типом события, вызвавшего смену состояния (вход, выход, поддержка).

Управление состоит в оптимальном переназначении каналов, когда пользователь покидает систему. Цель переупаковки – обеспечение ‘бо́льших возможностей по предоставлению свободных каналов, не противоречащему ограничениям на соседство ячеек. Когда пользователь входит в систему, управляющее действие состоит в выделении канала, или отказе от предоставления канала, если свободных каналов нет, или если выделение имеющегося канала приведет к большому числу блокировок в следующие моменты времени. Каждый случай блокировки или отказа в сервисе приводит к увеличению штрафа на временном шаге. Цель оптимизации – найти стратегию с минимальным суммарным штрафом.

Число различных состояний для этой задачи очень велико, поэтому прямое применение методов динамического программирования для практически интересных масштабов не возможно[13]. Однако, приближенные решения можно получить методиками, использующими сокращенное описание состояний и дифференцирование по времени. Для простоты, можно считать поддержку сервиса при переходе от ячейки к ячейке двумя событиями – уход пользователя из одной ячейки и новый вызов в другой. Кроме того, для каждой ячейки ограничимся лишь агрегированной информацией – числом свободных каналов и для каждого канала сети – количество раз, сколько этот канал используется в данный момент соседями данной ячейки. В этом случае число состояний пропорционально произведению числа ячеек и числа каналов, и задача становится обозримой

[13] Но как же быстро растет мощность и память компьютеров!

В работе [15] приводятся результаты для Пуассоновского потока заказов в сети из 50 ячеек с 70-ю каналами, при нагрузке несколько сотен вызовов в час в каждой ячейке. Алгоритм $TD(\lambda = 0)$ устойчиво обеспечивает меньшее число блокировок, в сравнении с лучшими известными эвристическими алгоритмами динамического распределения, и заведомо превосходит алгоритмы с фиксированным распределением частот.

Итоги

Симбиоз нейронных сетей и методов динамического программирования для решения многошаговых задач оптимизации – широко и интенсивно развиваемая область современной прикладной математики. Основные результаты получены в течение лишь последних 10 лет. К сожалению, не для всех условий использования алгоритмов в настоящий момент имеются строгие обоснования. С другой стороны, как и в области оптимального управления, важным исследовательским инструментом является компьютерное моделирование.

Алгоритмы нейродинамического программирования крайне привлекательны не только для традиционных "тяжелых" задач оптимизации, но и для построения простых приближенных решений в мобильных устройствах, встраиваемых и робототехнических системах.

Благодарности

Автор выражает глубокую благодарность Ю.В.Тюменцеву и Н.Г.Макаренко за плодотворные комментарии и творческое содействие.

Литература

[1] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[2] David H. Wolpert, Kagan Tumer. *An introduction to collective intelligence*. NASA Ames Research Center. 2000.
URL: http://ic.arc.nasa.gov/ic/projects/coin_pubs.html
[3] Дж. Дэннис, Р.Шнабель. *Численные методы безусловной оптимизации и решения нелинейных уравнений*. М. Мир, 1988, 440 с.
[4] Ф. Гилл, У. Муррей, М. Райт. *Практическая оптимизация*, 1985, Москва, Мир.
[5] Лутманов С.В. *Курс лекций по методам оптимизации*. Ижевск: НИЦ "Регулярная и хаотическая динамика", 2001, 368 стр.
[6] В.И. Ванько, О.В.Ермошина, Г.Н. Кувыркин. *Вариационное исчисление и оптимальное управление*. М: Изд. МВТУ им. Н.Э. Баумана, 2001.
[7] Филлипс Ч., Харбор Р. *Системы управления с обратной связью*. М: Лаборатория базовых знаний. 2002
[8] Горбань А.Н. *Обобщенная аппроксимационная теорема и свойства нейронных сетей*. Соросовский образовательный журнал, 1998.
[9] Stephen Piche, Jim Keeler, Greg Martin, Gene Boe, Doug Johnson and Mark Gerules. *Neural Network Based Model Predictive Control*. NIPS'1999 Volume 12, Paper 1029.
URL: http://nips.djvuzone.org/djvu/nips12/1029.djvu
[10] Р.Беллман. *Динамическое программирование*. М. Издательство иностранной литературы, 1960.
[11] Терехов С.А. *Нейросетевые аппроксимации плотности в задачах информационного моделирования*. Лекция для школы-семинара "Современные проблемы нейроинформатики", Москва, МИФИ, 25-27 января 2002 года.
[12] Терехов С.А. Введение в Байесовы сети. Лекции по нейроинформатике. М. МИФИ, 2003 ч.1., с.149.
[13] John N. Tsitsiklis and Benjamin Van Roy. *Feature-based methods for large scale dynamic programming*. Machine Learning, 22:59-94, 1996.
[14] Richard S. Sutton. *Open Theoretical Questions in Reinforcement Learning*. AT&T.
URL: www.cs.umass.edu/~rich
[15] Dimitri P. Bertsekas and John N.Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[16] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. *Reinforcement Learning: A Survey*. JAIR (Journal of AI Research), Volume 4, 1996.
URL: http://www.cs.brown.edu/people/lpk/rl-survey/rl-survey.html
[17] J.Moody and M.Saffell. *Reinforcement Learning for Trading*. Advances in Neural Information Processing Systems 11, M.S. Kearns, S.A. Solla and D.A. Cohn, eds., MIT Press, Cambridge, MA 1999.
[18] М. Интрилигатор. *Математические методы оптимизации и экономическая теория*. М. Айрис Пресс, 2002, 567 с.