Содержание

ЧАСТЬ I

Основы обучения без учителя

Мы начнем первую часть книги с обсуждения текущего положения дел в области машинного обучения и способов внедрения методов обучения без учителя. Мы реализуем готовый проект с нуля, что позволит понять, как настроить программную среду, загрузить и подготовить данные, выбрать алгоритмы машинного обучения и функцию потерь, а также оценить полученные результаты.

ГЛАВА 1

Обучение без учителя как один из видов машинного обучения

> Большую часть знаний люди и животные приобретают в ходе самообучения. Обучение без учителя можно представить как торт, глазурь на котором — обучение с учителем, а вишенка на торте — обучение с подкреплением. Мы знаем, как подготовить глазурь и вишенку, но не знаем, как испечь торт. Прежде чем задумываться о том, как приблизиться к созданию истинного ИИ, нам нужно справиться с задачей обучения без учителя.
>
> *Ян Лекун*

Данная глава посвящена изучению различий между обучением с учителем и без учителя, а также сильных и слабых сторон каждого из этих подходов. Кроме того, мы обсудим многие популярные алгоритмы обучения с учителем и без учителя и кратко рассмотрим такие методы, как обучение с частичным привлечением учителя и обучение с подкреплением.

Базовая терминология машинного обучения

Прежде чем углубиться в изучение различных типов машинного обучения, рассмотрим простой пример, который поможет лучше понять вводимые понятия: фильтрация спама. Предположим, требуется создать простую программу, которая получает сообщения электронной почты и классифицирует их либо как "спам", либо как "не спам". Это типичная задача классификации.

Освежим в памяти ключевые термины машинного обучения. В этой задаче *входными переменными* служат тексты сообщений электронной почты. Входные переменные также называют *независимыми переменными*, *признаками* или *предикторами*. В нашем случае *выходная переменная* — т.е. то, что мы пытаемся предсказать, — имеет значения "спам" или "не спам". Такую переменную также называют *целевой*, *зависимой* или *ответной* (а еще *классом*, поскольку это задача классификации).

Набор примеров, на которых обучается ИИ, называют *обучающим* (*тренировочным*) *набором*, а каждый отдельный пример называют *обучающим* (*тренировочным*) *примером* или *образцом* (выборкой). В процессе обучения ИИ пытается минимизировать свою *функцию потерь* (cost function), или *частоту ошибок* (error rate), или же (в более позитивной формулировке) максимизировать свою *функцию значения* (value function), в данном случае — процент корректной классификации электронных сообщений. Частоту ошибок вычисляют путем сравнения предсказанной метки с истинной.

Однако нас больше всего интересует то, насколько хорошо ИИ способен обобщать опыт, приобретенный в процессе обучения, на данные, которые ему прежде не встречались. Истинным тестом будет следующий: способен ли ИИ корректно классифицировать сообщения, которые не вошли в состав примеров, образующих тренировочный набор? Именно *ошибка обобщения* (generalization error), или *ошибка за пределами выборки* (out-of-sample error), является основной величиной, которую мы используем для оценки эффективности систем машинного обучения.

Набор не предоставленных ранее примеров известен как *тестовый* (*отложенный*) *набор*, поскольку эти данные не привлекаются для обучения ИИ. Если мы решим использовать несколько тестовых наборов (возможно, с целью калибровки ошибки обобщения в процессе обучения, что рекомендуется делать), то у нас могут быть промежуточные наборы, которые применяются для оценки прогресса еще до того, как будет задействован финальный тестовый набор. Такие промежуточные наборы называются *валидационными*.

Подведем итоги. Итак, ИИ обучается на тренировочном наборе данных (*опыт*) для снижения частоты ошибок (*производительность*) при маркировании спама (*задача*), а окончательным критерием успеха служит то, насколько хорошо опыт, приобретенный ИИ, обобщается на новые, еще не встречавшиеся данные (*ошибка обобщения*).

Обучение, основанное на наборе правил, и машинное обучение

Используя подход на основе правил, мы могли бы спроектировать фильтр с помощью явно сформулированных правил, помечающих как спам такие, например, сообщения электронной почты, в которых встречаются фразы наподобие “купите прямо сейчас” и т.п. Однако обеспечить долговременную поддержку подобной системы будет трудно, поскольку спамеры, изменив со

временем свою тактику, смогут обойти установленные нами правила фильтрации спама. Применяя систему на основе правил, мы будем вынуждены часто изменять их вручную, приспосабливаясь к новым обстоятельствам. К тому же настройка такой системы займет очень много времени. Подумайте только, сколько правил придется создать, чтобы все работало эффективно.

Вместо описанного подхода мы можем использовать машинное обучение, тренируя систему на наборе сообщений электронной почты и автоматически конструируя правила для пометки спама. Такая система обеспечит автоматическую подстройку с течением времени, а ее обучение и сопровождение обойдутся гораздо дешевле.

Если с отсевом спама, представляющим собой сравнительно простую задачу, мы еще могли бы справиться, задавая правила вручную, то во многих других случаях реализация такого подхода вообще невозможна. В качестве примера рассмотрим беспилотные автомобили. Представьте, какое количество правил нужно продумать, чтобы охватить все возможные ситуации, с которыми может столкнуться такой автомобиль. Решить подобную задачу практически нереально, если только не наделить автомобиль способностью обучаться и адаптироваться к окружению, основываясь на собственном опыте.

Системы машинного обучения также можно использовать в качестве инструмента для исследования данных с целью более глубокого понимания сути задачи, которую мы пытаемся решить. Так, в примере с электронной почтой можно изучить, какие слова или фразы чаще всего оказываются характерными признаками нежелательных сообщений, и использовать эту информацию для распознавания новых шаблонов спама.

Обучение с учителем и обучение без учителя

Существуют две основные методологии машинного обучения: *обучение с учителем* (supervised learning), или *контролируемое обучение*, и *обучение без учителя* (unsupervised learning), или *неконтролируемое обучение* (*самообучение*). Есть еще множество методик, которые являются своеобразными мостиками между ними.

При обучении с учителем интеллектуальный агент имеет доступ к *меткам*, или *маркерам*, которые могут быть использованы для улучшения производительности в ряде задач. В задаче фильтрации спама у нас имеется набор сообщений электронной почты с полными текстами каждого из них. Нам также известно (посредством меток), какие сообщения являются спамом, а

какие — нет. Метки ценны тем, что они помогают ИИ отделять спам от остальных сообщений при обучении с учителем.

В случае обучения без учителя метки отсутствуют. Поэтому задача ИИ не является четко определенной, что осложняет точное измерение эффективности обучения. Вновь обратимся к задаче фильтрации спама, но на этот раз без использования меток. Теперь интеллектуальный агент будет пытаться понять базовую структуру электронных сообщений, разбивая данные на группы, в каждой из которых сообщения сходны между собой, но отличаются от сообщений из других групп.

Задача обучения без учителя формулируется менее четко по сравнению с обучением с учителем, и интеллектуальному агенту труднее ее решать, и в то же время хорошо продуманный план действий позволяет получать более мощные решения. Это обусловлено следующим обстоятельством. ИИ может найти несколько групп, которые он пометит как "спам", но при этом могут быть обнаружены также группы, которые впоследствии будут помечены как "важное" или категоризированы как "семья", "работа", "новости", "покупки" и т.п. Другими словами, поскольку задача не задана строго, интеллектуальный агент может обнаружить новые интересные закономерности помимо тех, которые мы первоначально пытались найти.

Более того, система обучения без учителя лучше справляется с обнаружением новых закономерностей в предоставляемых ей неизвестных данных, чем система, основанная на обучении с учителем, что делает ее в перспективе куда более эффективной. В этом и заключается преимущество обучения без учителя.

Сильные и слабые стороны обучения с учителем

Обучение с учителем отлично справляется с оптимизацией в случае достаточно четко определенных задач с множеством меток. Предположим, имеется очень большой набор изображений, каждое из которых снабжено меткой. Если этот набор достаточно велик, а тренировка осуществляется с использованием алгоритмов машинного обучения (например, с помощью сверточных нейронных сетей) на достаточно мощных компьютерах, то мы сможем получить весьма неплохую систему классификации изображений на основе обучения с учителем.

Поскольку в обучении с учителем ИИ тренируется на данных, он будет способен измерить свою эффективность (посредством функции потерь), сравнивая предсказанную метку изображения с истинной, которая хранится в файле. ИИ будет пытаться минимизировать функцию потерь таким образом,

чтобы ошибка классификации изображений, которые еще не предоставлялись системе (например, изображений из тестового набора), была как можно меньшей.

Вот почему метки играют столь важную роль — они обеспечивают возможность измерения ошибки интеллектуальным агентом. ИИ использует эту информацию для повышения производительности с течением времени. В отсутствие меток ИИ не будет знать, насколько успешно (или неуспешно) он справляется с классификацией изображений.

Однако стоимость ручной разметки изображений довольно высокая. Даже предварительно подготовленные наборы данных содержат всего несколько тысяч меток. Это становится источником проблем, поскольку системы обучения хорошо справляются с классификацией изображений, для которых имеются метки, и плохо — с классификацией тех изображений, для которых метки отсутствуют.

Какими бы мощными ни были системы обучения с учителем, они ограничены в своих возможностях обобщения полученных знаний на изображения помимо тех, которые были включены в размеченный тренировочный набор. Поскольку бо́льшая часть доступных в мире данных не размечена, в случае обучения с учителем возможности ИИ по эффективному использованию приобретенного опыта применительно к новым данным довольно ограничены.

Иначе говоря, обучение с учителем отлично подходит для решения задач слабого ИИ, но хуже справляется с решением более амбициозных, но менее четко заданных задач ИИ сильного типа.

Сильные и слабые стороны обучения без учителя

Обучение с учителем превосходит обучение без учителя при решении четко сформулированных задач, для которых имеются четко определенные шаблоны, не сильно изменяющиеся с течением времени, и при условии, что у нас есть доступ к достаточно большим наборам размеченных данных.

Но в тех случаях, когда шаблоны неизвестны, постоянно меняются или же мы не имеем доступа к достаточно большим наборам размеченных данных, на помощь приходит обучение без учителя.

Вместо того чтобы руководствоваться метками, система обучения без учителя изучает базовую структуру данных, используемых в процессе тренировки модели. Это достигается за счет попыток представить тренировочные данные с помощью набора параметров, размер которого значительно меньше количества примеров, доступных в наборе данных. Благодаря этому

обучение без учителя позволяет идентифицировать различные шаблоны в наборе данных.

В примере с набором изображений (на этот раз не снабженных метками) ИИ может идентифицировать и группировать изображения, исходя из того, насколько они схожи между собой и отличаются от остальных изображений. Например, будут объединены в отдельные группы все изображения, похожие на стул, все изображения, похожие на собаку, и т.п.

Разумеется, ИИ не пометит эти группы как "стулья" или "собаки", но теперь, когда сходные изображения сгруппированы, человеку будет намного проще расставить метки. Вместо того чтобы помечать миллионы изображений вручную, специалисты могут вручную присваивать метки различным группам, после чего эти метки будут автоматически применены ко всем элементам группы.

Если по завершении начальной тренировки интеллектуальный агент обнаружит изображения, которые не принадлежат ни к одной из помеченных групп, то он создаст отдельные группы для таких неклассифицированных изображений, переложив на человека задачу последующей разметки новых групп.

Обучение без учителя превращает задачи, которые ранее не удавалось решать, в такие, которые допускают возможность их решения, и позволяет намного быстрее находить скрытые закономерности (шаблоны поведения) в исторических данных, как доступных в процессе тренировки, так и будущих, с которыми система до сих пор не сталкивалась. Более того, тем самым ИИ прокладывает путь к обработке огромных хранилищ неразмеченных данных, существующих в мире.

Несмотря на то что обучение без учителя менее пригодно для решения специфических задач со строгой формулировкой по сравнению с обучением с учителем, оно лучше справляется с "размытыми" задачами, стоящими перед сильным ИИ, и обобщением приобретенных знаний.

Также немаловажен тот факт, что обучение без учителя способно оказать помощь при решении многих обычных задач, с которыми часто приходится сталкиваться аналитикам при создании приложений машинного обучения.

Использование обучения без учителя для улучшения систем машинного обучения

Недавние успехи в области машинного обучения обусловлены расширением возможностей доступа к большим объемам данных, резким повышением вычислительных мощностей, появлением облачных ресурсов, а также прорывом в разработке соответствующих алгоритмов. Но эти успехи главным образом достигнуты при решении задач узкого ИИ, таких как классификация изображений, компьютерное зрение, распознавание речи, обработка естественного языка и машинный перевод.

Решение более амбициозных задач ИИ требует привлечения обучения без учителя. Рассмотрим наиболее распространенные проблемы, с которыми приходится сталкиваться исследователям при построении приложений, и попытаемся понять, какую помощь в этом может оказать обучение без учителя.

Нехватка размеченных данных

> Построение ИИ можно сравнить с созданием космического корабля. Располагая мощным двигателем, но мизерным запасом топлива, вы не сможете вывести ракету на орбиту. В то же время ракета с тоннами топлива, но маломощным двигателем не сможет оторваться от земли. Чтобы запустить ракету в космос, необходим мощный двигатель и достаточно большой запас топлива.
>
> *Эндрю Ын*

Если провести аналогию между машинным обучением и космическим кораблем, то данные можно уподобить топливу — без них наш аппарат не смог бы взлететь. Но не все данные равноценны. Чтобы использовать алгоритмы обучения с учителем, нам требуется множество размеченных данных, генерирование которых — трудоемкий и затратный процесс.

В случае обучения без учителя мы можем автоматически помечать неразмеченные образцы. Вот как это делается. Мы кластеризуем все образцы, а затем применяем метки из размеченных примеров к неразмеченным, принадлежащим к одному кластеру. Неразмеченные образцы получат метки тех образцов, с которыми они имеют наибольшее сходство. (Кластеризация подробно исследуется в главе 5.)

Переобучение

Если тренировочные данные используются для того, чтобы обучить алгоритм машинного обучения чрезмерно сложной функции, то он может плохо работать на примерах, которые ему еще не предоставлялись, скажем, на примерах из наборов, зарезервированных в качестве валидационных или тестовых. В этом случае в результате извлечения чересчур большого объема информации из шума, содержащегося в данных, велика вероятность проявления эффектов *переобучения* (overfitting), приводящих к значительному ухудшению обобщающей способности модели. Иными словами, алгоритм *запоминает* обучающие данные, а не учится тому, как обобщать приобретенные знания на другие случаи[1].

Для преодоления этой проблемы можно использовать обучение без учителя в качестве регуляризатора. *Регуляризация* — это процесс снижения сложности алгоритма машинного обучения, способствующий извлечению сигнала из зашумленных данных. Одной из форм регуляризации служит предварительное обучение без учителя. Вместо того чтобы передавать оригинальные входные данные алгоритму обучения с учителем, мы можем генерировать их новое представление, которое и будет передаваться алгоритму.

Это новое представление захватывает наиболее существенную составляющую исходных данных — их истинную базовую структуру, одновременно избавляя нас от не содержащего значимой информации шума. Алгоритму обучения с учителем, которому передается новое представление, будет легче справляться с оставшимися шумами и извлекать полезный сигнал, что приведет к улучшению его обобщающей способности. (Более подробному обсуждению этой темы посвящена глава 7.)

Проклятие размерности

Несмотря на невиданный рост доступных вычислительных мощностей обработка больших данных все еще доставляет трудности алгоритмам машинного обучения. Вообще говоря, добавление большего количества примеров не

[1] Другая проблема, с которой можно столкнуться в приложениях машинного обучения, — *недообучение*, но она решается легче. Причиной недообучения становится излишняя простота модели — алгоритму не удается построить достаточно сложную аппроксимирующую функцию, позволяющую принимать решения, которые хорошо подходили бы для текущей задачи. Избавиться от этой проблемы можно либо за счет расширения алгоритма (путем ввода дополнительных параметров, увеличения количества итераций обучения и т.п.), либо за счет применения более сложного алгоритма обучения.

является проблемой, поскольку, используя такие современные модели распределенных вычислений, как Spark, мы можем распараллеливать выполнение операций. В то же время, чем больше имеется признаков, тем более трудоемким становится обучение.

В случае пространств очень большой размерности алгоритмы обучения с учителем должны обучаться тому, как разделять точки и строить функцию-аппроксиматор, обеспечивающую принятие правильных решений. Если признаков очень много, то эта процедура оказывается дорогостоящей как в отношении времени, так и в отношении требуемых вычислительных ресурсов. В некоторых случаях данный фактор вообще делает невозможным получение достаточно эффективных и быстрых решений.

Эта проблема известна как *проклятие размерности*, и для ее решения хорошо подходит обучение без учителя. Используя инструменты снижения размерности, мы можем найти наиболее значимые признаки в исходном наборе и снизить, пусть даже за счет потери незначительной части важной информации, число измерений до приемлемого уровня, а затем применить алгоритмы обучения с учителем для более эффективного поиска подходящей аппроксимирующей функции. (Снижение размерности обсуждается в главе 3.)

Конструирование признаков

Конструирование признаков (feature engineering) — одна из самых важных задач, которые должны решать исследователи. Не располагая достоверными признаками, алгоритм машинного обучения окажется неспособным достаточно надежно разделять точки в пространстве для того, чтобы принимать корректные решения относительно незнакомых ему примеров. Однако, как правило, конструирование признаков — весьма трудоемкая задача, требующая креативного вмешательства человека для создания признаков нужного типа в ручном режиме. Вместо этого мы можем использовать представление, полученное с помощью алгоритмов обучения без учителя, для автоматического обучения признакам подходящего типа, помогающим решить конкретную задачу. (Автоматическое извлечение признаков обсуждается в главе 7.)

Выбросы

Важную роль играет также качество данных. Если алгоритмы машинного обучения тренируются на данных, содержащих редкие *выбросы*, которые несколько искажают истинную картину, то ошибка обобщения будет меньше, чем если бы мы игнорировали их или обрабатывали отдельно. В случае

обучения без учителя мы можем обнаруживать выбросы, используя механизмы снижения размерности, и создать два приложения: одно, специально предназначенное для обработки выбросов, и другое, предназначенное для обработки нормальных данных. (Построением системы обнаружения аномалий мы займемся в главе 4.)

Дрейф данных

Модели машинного обучения также должны считаться с *дрейфом данных* (data drift). Если данные, которые используются моделью в целях прогнозирования, статистически отличаются от данных, на которых она обучалась, то может потребоваться повторно обучить ее на тех данных, которые лучше соответствуют текущей задаче. Если модель не подвергается повторной тренировке или ей не удается распознать дрейф, то ее предсказательная способность на текущих данных от этого пострадает.

Создавая распределения вероятностей с помощью обучения без учителя, мы можем оценить, насколько текущие данные отличаются от данных тренировочного набора. И если эти различия существенны, то мы можем автоматически запускать повторную тренировку модели. (Построению такого типа систем посвящена глава 12.)

Краткий обзор алгоритмов машинного обучения с учителем

Прежде чем углубиться в изучение систем обучения без учителя, целесообразно кратко ознакомиться с алгоритмами обучения с учителем и тем, как они работают. Это поможет нам очертить границы, в которые вписывается обучение без учителя.

Существуют два основных типа задач, относящихся к сфере обучения с учителем: *классификация* и *регрессия*. В случае классификации ИИ должен корректно относить элементы к одному из двух или нескольких классов. Если есть всего два класса, то мы имеем дело с так называемой *бинарной классификацией*. Если же классов три или более, то задачу такого рода называют *многоклассовой* (*мультиклассовой*) *классификацией*.

Задачи классификации также называются *дискретными предсказаниями*, поскольку каждый класс образует отдельную группу. Другое их название — *качественный*, или *категориальный*, анализ.

В случае регрессии ИИ должен предсказывать значения *непрерывной* переменной, а не дискретной. Другое название регрессионных задач — *количественный анализ*.

Алгоритмы машинного обучения с учителем расширяют этот спектр задач, но все они нацелены на минимизацию некоторой функции потерь (либо максимизации функции значения), которая ассоциирована с метками, имеющимися в наборе данных.

Как уже упоминалось ранее, больше всего нас должно заботить то, насколько хорошо система машинного обучения обобщается на ранее не предоставленные образцы. Для минимизации ошибки обобщения очень важно сделать правильный выбор алгоритма обучения с учителем.

Для обеспечения как можно меньшей ошибки обобщения необходимо, чтобы сложность алгоритмической модели соответствовала сложности истинной функции, заложенной в данных. Мы не знаем, что в действительности представляет собой эта функция. Если бы это было известно, то нам не потребовалось бы задействовать машинное обучение для создания модели — мы могли бы просто использовать эту функцию для получения корректного ответа. Но поскольку истинная функция неизвестна, мы выбираем алгоритм машинного обучения для тестирования гипотез и находим модель, которая наилучшим образом аппроксимирует истинную функцию (т.е. приводит к наименьшей возможной ошибке обобщения).

Если сложность того, что моделирует алгоритм, меньше сложности истинной функции, то мы столкнемся с проблемой *недообучения* (underfitting). В таком случае ошибку обобщения можно уменьшить посредством выбора алгоритма, способного моделировать более сложную функцию. Но если алгоритм порождает чересчур сложную модель, то мы столкнемся с *переобучением* (overfitting) и получим плохие результаты для ранее не встречавшихся данных, увеличив ошибку обобщения.

Другими словами, отдавать предпочтение более сложным алгоритмам по сравнению с более простыми не всегда будет правильным выбором — иногда простое лучше сложного. Каждому алгоритму свойствен свой набор сильных и слабых сторон, а также допущений, и знание того, какие из них следует применять для решения конкретной задачи с имеющимися данными, является очень важным условием овладения искусством машинного обучения на профессиональном уровне.

Далее мы рассмотрим наиболее популярные алгоритмы машинного обучения с учителем (включая их практические применения), а затем — алгоритмы обучения без учителя[2].

Линейные методы

Самые простые алгоритмы обучения с учителем моделируют линейное соотношение между входными признаками и выходной переменной, значение которой мы хотим предсказать.

Линейная регрессия

Простейшим из всех алгоритмов является *линейная регрессия*, в которой используется модель, предполагающая линейное соотношение между входными переменными (X) и единственным выходным значением (y). Если соотношение между входами и выходом действительно линейное, а между входными переменными отсутствует значительная корреляция (ситуация, известная как *коллинеарность*), то линейная регрессия станет неплохим выбором. Если же истинное соотношение носит более сложный или нелинейный характер, то использование линейной регрессии приведет к недообучению[3].

Учитывая простоту самого алгоритма, интерпретация моделируемого им соотношения также не представляет никакого труда. *Интерпретируемость* результатов — очень важный фактор машинного обучения, поскольку решение должно быть понято и воплощено в жизнь как техническими, так и нетехническими отраслевыми специалистами. В отсутствие возможности интерпретации решения оно превращается в загадочный черный ящик.

Сильные стороны

Линейная регрессия отличается простотой, интерпретируемостью и несклонностью к переобучению, поскольку она не в состоянии моделировать чрезмерно сложные отношения. Она великолепно подходит для тех случаев, когда соотношение между входными и выходными переменными в основном является линейным.

[2] Этот список не является исчерпывающим, но включает большинство алгоритмов, которые обычно используются в машинном обучении.

[3] К числу других потенциальных проблем, делающих линейную регрессию неудачным вариантом выбора, относятся выбросы, а также наличие корреляции между составляющими ошибки и непостоянство их дисперсии.

Слабые стороны

Линейная регрессия будет недообучаться на данных, если соотношение между входными и выходными переменными нелинейное.

Применение

Поскольку вес человека в целом линейно зависит от его роста, линейная регрессия будет хорошо предсказывать вес человека, используя его рост в качестве входных данных, и наоборот.

Логистическая регрессия

Простейшим алгоритмом классификации является *логистическая регрессия*, которая также относится к линейным методам, но ее предсказания преобразуются с помощью логистической функции. Выходом такого преобразования являются *вероятности классов* — иначе говоря, вероятности принадлежности образца к тому или иному классу, причем сумма всех вероятностей по всем образцам должна равняться единице. Затем каждому примеру приписывается класс, вероятность принадлежности к которому максимальная.

Сильные стороны

Как и линейная регрессия, логистическая регрессия отличается простотой и интерпретируемостью. Если классы, которые мы пытаемся предсказать, не перекрываются и линейно разделимы, то логистическая регрессия — отличный выбор.

Слабые стороны

Логистическая регрессия не работает в случае линейно неразделимых классов.

Применение

Если классы — например, рост ребенка и рост взрослого человека — в целом не перекрываются, то логистическая регрессия даст хорошие результаты.

Методы на основе соседства точек

Другую группу очень простых алгоритмов образуют методы, основанные на понятии *соседства точек* (neighborhood). Методы этой группы — *ленивые ученики*, поскольку в данном случае обучение разметке новых точек основывается на их близости к уже размеченным точкам. В отличие от линейной или

логистической регрессии модели, основанные на соседстве точек, не обучаются предсказанию меток для новых точек. Вместо этого они предсказывают метки для новых точек, исключительно исходя из того, насколько новые точки удалены от существующих размеченных точек. *Ленивое обучение* (lazy learning) также относится к методам *обучения на примерах* (instance-based learning), или *непараметрическим* методам.

Метод *k*-ближайших соседей

Наиболее популярный метод этой категории — *k-ближайших соседей* (k-nearest neighbors — KNN). Для пометки каждой новой точки алгоритм осуществляет поиск k ближайших соседних помеченных точек (где k — целое число) и наделяет уже размеченных соседей правом голоса при принятии решения относительно того, какую метку следует присвоить новой точке. В качестве меры удаленности новой точки от ближайших соседей в KNN по умолчанию используется евклидово расстояние.

От выбора значения k зависит очень многое. Если оно установлено слишком низким, то метод приобретает повышенную гибкость, очерчивая границы, учитывающие множество нюансов, и переобучаясь на данных. Если же для k выбрано слишком большое значение, то метод теряет гибкость, очерчивая слишком грубые границы и потенциально недообучаясь на данных.

Сильные стороны

KNN, в отличие от линейных методов, обладает значительной гибкостью и приспособлен для обучения более сложным нелинейным соотношениям. При этом он сохраняет простоту и интерпретируемость.

Слабые стороны

KNN плохо работает в тех случаях, когда количество наблюдений и признаков достигает критической величины. В условиях заполненного большим количеством точек многомерного пространства метод становится неэффективным с вычислительной точки зрения, поскольку для предсказания признаков приходится рассчитывать расстояния от новой точки до уже размеченных многочисленных соседних точек. Он не позволяет использовать эффективную модель с уменьшенным количеством параметров для генерирования необходимых предсказаний. Кроме того, метод весьма чувствителен к выбору параметра k. При слишком низких значениях k метод может переобучаться, а при слишком высоких — недообучаться.

Применение

KNN широко применяется в рекомендательных системах, например тех, которые используются для прогнозирования пользовательских предпочтений: фильмов (Netflix), музыки (Spotify), друзей (Facebook), фотографий (Instagram), поисковых запросов (Google) и потребительских товаров (Amazon). В частности, метод может оказаться полезным при предсказании предпочтений отдельного пользователя на основании известных предпочтений пользователей с аналогичными вкусами (так называемая *коллаборативная фильтрация*) или на основании собственных предпочтений пользователя в прошлом (так называемая *фильтрация по содержимому*).

Методы на основе деревьев решений

Вместо того чтобы использовать линейный метод, мы можем построить дерево решений, в котором все примеры *сегментируются*, или *стратифицируются*, по отдельным областям на основании имеющихся меток. По завершении сегментирования каждая область соответствует определенному классу меток (для задач классификации) или диапазону предсказанных значений (для задач регрессии). Этот процесс аналогичен тому, как если бы мы поручили ИИ автоматически создать правила, ориентированные на получение наилучших решений или предсказаний.

Одиночное дерево решений

Простейший вариант — метод *одиночного дерева решений*, в котором ИИ выполняет один проход по тренировочным данным, создает правила для сегментирования данных на основании имеющихся меток и использует результирующее дерево решений для создания предсказаний в отношении не представленных ранее данных, включенных в валидационный или тестовый набор. Однако одиночное дерево решений обычно плохо обобщает то, чему обучилось в процессе тренировки, на незнакомые примеры ввиду переобучения на одной-единственной тренировочной итерации.

Бэггинг

В качестве улучшения одиночного дерева решений можно воспользоваться *бутстрэп-агрегированием*, или *бэггингом*, когда мы формируем из тренировочных данных *несколько случайных выборок примеров*, а затем создаем дерево решений для каждой выборки и предсказываем выход для каждого примера

путем усреднения предсказаний отдельных деревьев. За счет использования *рандомизации* выборок и усреднения результатов нескольких деревьев (подход, известный как *метод ансамблей*) бэггинг помогает частично справиться с переобучением, свойственным одиночному дереву решений.

Случайные леса

Мы можем дополнительно ослабить эффекты переобучения, семплируя не только примеры, но и сами предикторы. В методе *случайных лесов* мы отбираем несколько случайных выборок примеров из тренировочных данных, как это делается в бэггинге, но в каждом дереве решений мы создаем расщепление, основанное не на всех предикторах, а на *случайной выборке предикторов*. Количество предикторов, устанавливаемых для каждого дерева, обычно выбирается равным квадратному корню из общего их количества.

Семплируя предикторы описанным способом, алгоритм случайных лесов создает деревья, еще в меньшей степени скоррелированные между собой (по сравнению с деревьями бэггинга), тем самым ослабляя эффекты переобучения и уменьшая ошибку обобщения.

Бустинг

В другом подходе, который называется *бустинг*, также создается несколько деревьев, но при этом деревья формируются *последовательно*, что позволяет использовать знания, которым ИИ обучился на предыдущем дереве, для улучшения результатов, получаемых с помощью следующего дерева. Глубина каждого дерева поддерживается на довольно мелком уровне, с небольшим количеством расщеплений, и обучение происходит последовательно, дерево за деревом. Из всех методов, основанных на деревьях решений, наилучшую производительность продемонстрировали *машины градиентного бустинга*, которые часто одерживают победы на соревнованиях по машинному обучению[4].

Сильные стороны

Методы на основе деревьев решений относятся к числу наиболее производительных алгоритмов обучения с учителем, предназначенных для решения задач прогнозирования. Эти методы способны выявлять сложные соотношения в данных путем обучения многим простым правилам,

[4] Подробнее о градиентном бустинге можно прочитать в блоге Бена Гормана (`http://bit.ly/2S1C8Qy`).

по одному правилу за раз. Кроме того, они способны обрабатывать отсутствующие данные и категориальные признаки.

Слабые стороны

Методы на основе деревьев решений с трудом поддаются интерпретации, особенно в тех случаях, когда для получения надежных предсказаний требуется использовать большое количество признаков. Кроме того, по мере увеличения количества признаков производительность также становится проблемой.

Применение

Градиентный бустинг и случайные леса отлично подходят для решения задач прогнозирования.

Метод опорных векторов

Вместо того чтобы строить деревья для разделения данных, мы можем использовать алгоритмы, предназначенные для создания гиперплоскостей, разделяющих данные на основании имеющихся признаков. Соответствующий подход получил название *метод опорных векторов* (support vector machine — SVM). SVM не гарантирует идеального разделения (не все точки в определенной области гиперпространства обязаны иметь одну и ту же метку), но расстояние между пограничными точками, с которыми ассоциирована некоторая метка, и пограничными точками, с которыми ассоциирована другая метка, должны быть как можно бо́льшими. Кроме того, границы не обязаны быть линейными — мы можем обеспечить более гибкое разделение данных, используя нелинейные ядра.

Нейронные сети

Обучение представлениям данных можно проводить с использованием *нейронных сетей*, которые состоят из входного слоя, нескольких скрытых слоев и выходного слоя[5]. Входной слой использует признаки, тогда как выходной слой пытается добиться соответствия переменной отклика (ответной переменной). Скрытые слои представляют собой вложенную иерархию

[5] Для получения более подробной информации о нейронных сетях обратитесь к книге *Глубокое обучение* Яна Гудфеллоу, Иошуа Бенджио и Аарона Курвилля (`http://www.deeplearningbook.org/`).

абстрактных понятий — каждый слой (или понятие) пытается понять, как предыдущий слой соотносится с выходным слоем.

Опираясь на эту иерархию, нейронная сеть может обучаться сложным понятиям путем их формирования на основе более простых понятий. Нейронные сети — один из наиболее мощных подходов к аппроксимации функций, однако с ним сопряжены такие проблемы, как переобучение и трудность интерпретации — недостатки, которые мы будем подробно обсуждать далее.

Краткий обзор алгоритмов машинного обучения без учителя

А теперь переключимся на рассмотрение задач, в которых мы имеем дело с неразмеченными данными. Применяемые в таких случаях алгоритмы обучения без учителя не создают предсказания, а пытаются обучиться базовой структуре данных.

Снижение размерности

Алгоритмы снижения размерности транслируют оригинальное многомерное пространство входных данных в пространство более низкой размерности, фильтруя наименее релевантные признаки и сохраняя как можно большее количество признаков, представляющих интерес. Снижение размерности позволяет эффективнее выявлять шаблоны и решать крупномасштабные, вычислительно трудоемкие задачи (чаще всего связанные с обработкой изображений, видео, речи и текста).

Линейная проекция

Существуют две основные разновидности алгоритмов снижения размерности: *линейная проекция* и *нелинейное снижение размерности*.

Анализ главных компонент

Один из подходов к изучению базовой структуры данных заключается в идентификации наиболее важных признаков, объясняющих изменчивость примеров. Не все признаки равноценны. Одни из них испытывают лишь незначительные изменения и поэтому менее полезны в плане объяснения данных. В то же время другие признаки могут варьироваться в определенных пределах, и именно они заслуживают более пристального внимания, поскольку с их помощью легче построить модель, предназначенную для разделения данных.

В методе PCA (principal component analysis — анализ главных компонент) алгоритм находит такое представление данных низкой размерности, которое обеспечивает сохранение максимально возможной доли их вариативности. Размерность этого представления значительно меньше размерности полного набора данных (т.е. полного числа признаков). Переход к пространству пониженной размерности сопровождается потерей определенной части дисперсии, но упрощает идентификацию базовой структуры данных, что обеспечивает более эффективное решение таких задач, как кластеризация.

Существует несколько вариантов метода PCA, которые мы более подробно обсудим в последующих главах. В их число входят мини-пакетные версии, такие как *инкрементный PCA*, нелинейные версии, такие как *ядерный PCA*, и разреженные версии, такие как *разреженный PCA*.

Сингулярное разложение

Суть другого подхода к изучению базовой структуры данных заключается в снижении ранга исходной матрицы признаков до меньшего значения с сохранением возможности восстановления исходной матрицы путем образования линейной комбинации некоторых из векторов полученной матрицы более низкого ранга. Это и есть метод SVD (singular value decomposition — сингулярное разложение). Генерируя матрицу меньшего ранга, метод SVD оставляет те из векторов исходной матрицы, которые содержат наибольшую часть информации (обеспечивают наибольшее сингулярное значение). Эта матрица пониженного ранга захватывает наиболее важные элементы исходного пространства признаков.

Случайная проекция

Аналогичный алгоритм снижения размерности включает проецирование точек из многомерного пространства в пространство более низкой размерности таким образом, чтобы сохранить масштаб расстояний между точками. Для этого используют либо *случайную гауссовскую* либо *случайную разреженную* матрицу.

Обучение на многообразиях

Как PCA, так и случайная проекция подразумевают линейное проецирование данных из многомерного пространства в пространство низкой размерности. Результаты можно улучшить за счет применения нелинейного преобразования вместо линейного — такой подход известен как *обучение на многообразиях* (manifold learning), или *нелинейное снижение размерности*.

Isomap

Этот алгоритм обучается внутренней геометрии данных путем оценки *геодезических* (*искривленных*) *расстояний* между каждой точкой и ее ближайшими соседями, а не евклидовых. Isomap использует эту информацию для последующего вложения многомерного пространства в пространство пониженной размерности.

Стохастическое вложение соседей с t-распределением

Другой подход к снижению размерности, называемый *t-SNE* (t-distributed stochastic neighbor embedding), основан на вложении многомерных данных в пространство, имеющее всего лишь два-три измерения, что позволяет визуализировать преобразованные данные. В этом двух- или трехмерном пространстве схожие примеры располагаются на небольших, а несхожие — на больших расстояниях друг от друга.

Словарное обучение

Подход, известный как *словарное обучение*, подразумевает обучение разреженному представлению базовых данных. Этими репрезентативными элементами служат простые бинарные векторы (состоящие из нулей и единиц), и каждый пример в наборе данных может быть реконструирован в виде взвешенной суммы репрезентативных элементов. Матрица (*словарь*), генерируемая в процессе обучения без учителя, заполнена главным образом нулями и содержит лишь небольшое количество ненулевых весов.

Благодаря созданию подобного словаря алгоритм способен эффективно идентифицировать наиболее существенные репрезентативные элементы исходного пространства признаков — те, которые содержат наибольшее количество ненулевых весов. Менее важные элементы содержат меньшее количество ненулевых весов. Как и PCA, словарное обучение отлично подходит для разделения данных и идентификации интересующих нас шаблонов.

Анализ независимых компонент

Одной из общих проблем, порождаемых неразмеченными данными, становится наличие множества независимых сигналов, которые вложены в имеющиеся в нашем распоряжении признаки. Используя *анализ независимых компонент* (independent component analysis — ICA), мы можем разделить эту смесь сигналов на независимые компоненты. После такого разделения можно реконструировать любой из оригинальных признаков, образуя линейные комбинации сгенерированных индивидуальных компонент. Метод ICA широко

применяется для обработки сигналов (например, для идентификации отдельных голосов в аудиозаписи из шумного кафетерия).

Латентное размещение Дирихле

Обучение без учителя также может быть использовано для объяснения набора данных путем изучения факторов, которые обусловливают сходство отдельных частей набора между собой. Это требует обучения ненаблюдаемым элементам набора данных — подход, получивший название *латентное размещение Дирихле* (latent Dirichlet allocation — LDA). Предположим, имеется текстовый документ, содержащий множество слов. Эти слова не являются случайными, они подчиняются определенной структуре.

Такую структуру можно смоделировать как ненаблюдаемые элементы — так называемые *темы*. После проведения соответствующей тренировки метод LDA может объяснить данный документ с помощью ограниченного набора тем, для каждой из которых имеется небольшой набор часто используемых слов. Это и есть скрытая структура, которую LDA способен захватывать, что позволяет лучше объяснять ранее неструктурированный текстовый корпус.

Снижение размерности сводит оригинальный набор признаков к меньшему набору, включающему лишь наиболее важные признаки. Далее мы можем запускать другие алгоритмы обучения без учителя на этом меньшем наборе признаков с целью поиска шаблонов, представляющих интерес (см. следующий раздел, посвященный кластеризации), или, при наличии меток, для ускорения цикла обучения алгоритмов обучения с учителем путем передачи им меньшей матрицы признаков вместо использования оригинальной матрицы.

Кластеризация

После редуцирования набора оригинальных признаков до набора меньшего размера мы можем приступить к поиску интересующих нас закономерностей путем группирования схожих примеров. Этот процесс, называемый *кластеризацией*, можно реализовать с помощью целого ряда алгоритмов обучения без учителя и использовать в таких задачах, как сегментирование рынка.

Метод *k*-средних

Чтобы успешно справиться с кластеризацией, мы должны идентифицировать отдельные группы, примеры в пределах которых схожи между собой, но

отличаются от примеров, относящихся к другим группам. Одним из таких алгоритмов является *кластеризация методом* k-*средних* (*k*-means clustering). В данном алгоритме мы задаем желаемое количество кластеров, *k*, и алгоритм относит каждый пример ровно к одному из этих *k* кластеров. Алгоритм оптимизирует группирование, минимизируя внутрикластерную вариацию (называемую *инерцией*) так, чтобы сумма внутрикластерных вариаций по всем *k* кластерам была как можно меньшей.

С целью ускорения процесса кластеризации метод *k*-средних случайным образом относит каждое наблюдение к одному из *k* кластеров, а затем переназначает наблюдения для минимизации евклидовых расстояний между каждым наблюдением и центральной точкой кластера, или *центроидом*. Как следствие, различные запуски алгоритма *k*-средних — каждый со своей начальной точкой — будут приводить к несколько различающимся отнесениям наблюдений к тем или иным кластерам. Из этой серии различных запусков мы выбираем тот, который характеризуется наилучшим разделением, дающим наименьшую общую сумму внутрикластерных вариаций по всем *k* кластерам[6].

Иерархическая кластеризация

Альтернативой обычной кластеризации служит так называемая *иерархическая кластеризация*, не требующая предварительного задания количества кластеров. В одном из вариантов иерархической кластеризации, известном как *агломеративная кластеризация*, применяется кластеризация на основе деревьев и создается так называемая *дендрограмма*. Последняя графически отображается в виде перевернутого дерева, в котором листья находятся внизу, а ствол дерева — вверху.

Самые нижние листья — это индивидуальные примеры, содержащиеся в наборе данных. Затем, по мере перемещения вверх по перевернутому дереву, иерархическая кластеризация объединяет листья на основании их взаимного сходства. В первую очередь объединяются наиболее схожие примеры (или группы примеров), в последнюю очередь — наименее схожие. В конечном счете описанный итеративный процесс приводит к тому, что все примеры оказываются связанными, образуя единый ствол дерева.

Такое графическое представление весьма информативно. Как только алгоритм иерархической кластеризации завершит свою работу, мы сможем проанализировать дендрограмму и определить, в каком месте мы хотим обрезать дерево. Чем ниже линия обреза, тем больше останется индивидуальных

[6] Существуют более быстрые варианты кластеризации методом *k*-средних, которые мы обсудим в последующих главах.

ветвей (т.е. кластеров). Если мы хотим уменьшить количество кластеров, то линия обреза должна располагаться высоко на дендрограмме, ближе к стволу, находящемуся на самом верху перевернутого дерева. Размещение линии обреза аналогично выбору количества k кластеров в алгоритме кластеризации методом k-средних[7].

DBSCAN

DBSCAN (density-based spatial clustering of applications with noise — основанная на плотности пространственная кластеризация для приложений с шумами) — еще более мощный алгоритм кластеризации. DBSCAN группирует вместе тесно расположенные примеры. Теснота расположения определяется как минимальное количество примеров, взаимные расстояния между которыми не превышают заданной величины. При этом мы задаем как требуемое минимальное количество таких примеров, так и определенное расстояние, ограничивающее пределы близости.

Если пример находится в пределах указанного расстояния от нескольких кластеров, то он будет группироваться с тем из них, для которого теснота расположения оказывается большей. Примеры, расположенные в областях малой плотности, помечаются как выбросы.

В отличие от метода k-средних мы не должны предварительно задавать количество кластеров. Кластеры могут иметь произвольную форму. Метод DBSCAN гораздо меньше подвержен искажениям, которые обычно создаются выбросами.

Извлечение признаков

Обучение без учителя позволяет обучаться новым представлениям оригинальных признаков данных — это называется *извлечением признаков* (feature extraction). Данный подход может применяться для эффективного снижения размерности данных путем создания редуцированного подмножества оригинальных признаков. А кроме того, это позволяет генерировать новые признаки с целью повышения производительности в задачах обучения с учителем.

[7] В случае иерархической кластеризации по умолчанию используются евклидовы расстояния, но можно задействовать и другие аналогичные метрики, например *расстояние, исчисляемое на основе корреляции*, которое мы рассмотрим в последующих главах.

Автокодировщики

Для генерирования новых представлений признаков можно использовать нерекуррентную нейронную сеть прямого распространения, в которой количество узлов во входном и выходном слоях совпадает. Эта нейронная сеть, так называемый *автокодировщик* (autoencoder), способна эффективно реконструировать исходные признаки, обучаясь новому представлению с помощью промежуточных скрытых слоев[8].

Каждый скрытый слой автокодировщика обучается представлению оригинальных признаков, причем каждый последующий слой достраивает представление, которому обучились предыдущие слои. Благодаря этому автокодировщик обучается все более сложным представлениям на основе более простых.

На выходе автокодировщика мы получаем окончательное новое представление, которому он обучился. Затем это представление может быть использовано в качестве входа для модели обучения с учителем с целью уменьшения ошибки обобщения.

Извлечение признаков путем контролируемого обучения сетей прямого распространения

Если в нашем распоряжении имеются метки, то возможной альтернативой подхода, основанного на извлечении признаков, является использование нерекуррентной сети прямого распространения, в которой выходной слой пытается предсказать правильную метку. Как и в случае автокодировщиков, каждый скрытый слой обучается представлению оригинальных признаков.

Но когда генерируются новые представления, сеть работает непосредственно под управлением признаков. Для извлечения окончательного вновь изученного представления оригинальных признаков мы извлекаем предпоследний скрытый слой, непосредственно предшествующий выходному. Далее этот предпоследний слой можно использовать в качестве входа в любой модели обучения с учителем.

[8] Существует несколько разновидностей автокодировщиков, каждый из которых обучается отдельному типу представлений. В их число входят *обесшумливающие автокодировщики*, *разреженные автокодировщики* и *вариационные автокодировщики*. Все они будут подробно рассмотрены в последующих главах.

Глубокое обучение без учителя

Обучение без учителя находит целый ряд важных применений в технологии глубокого обучения; с некоторыми из них мы познакомимся в последующих главах. Соответствующая область исследований называется *глубоким обучением без учителя* (unsupervised deep learning).

Еще совсем недавно обучение глубоких сетей было практически неосуществимым из-за трудоемкости вычислений. В таких нейронных сетях скрытые слои обучаются внутренним представлениям для решения текущей задачи. Представления постепенно улучшаются за счет обновления весов различных узлов с использованием *градиента функции ошибки* на каждой итерации тренировки.

Подобное обновление весов требует интенсивных вычислений, в процессе которых могут возникать две основные трудности. Во-первых, градиент функции ошибки может уменьшиться до очень небольших значений, а поскольку обратное распространение ошибки основано на перемножении этих значений, веса сети будут обновляться очень медленно или вообще не обновляться, препятствуя ее обучению[9]. Это явление известно как *проблема затухающих градиентов*.

Суть другой проблемы, противоположной той, которая была только что описана, заключается в том, что градиент функции ошибки также может принимать очень большие значения. В таком случае веса сети могут обновляться с использованием огромных приращений, что делает процесс обновления крайне нестабильным. Это явление известно как *проблема взрывных градиентов*.

Предварительное обучение без учителя

Для преодоления описанных выше проблем, возникающих в процессе обучения очень глубоких, многослойных нейронных сетей, исследователи, занимающиеся машинным обучением, тренируют нейронные сети на протяжении нескольких последовательных этапов, каждый из которых включает мелкую нейронную сеть. Выход одной мелкой сети используется в качестве входа следующей нейронной сети. В типичных случаях первая мелкая сеть в этом конвейере включает неконтролируемую, т.е. обучаемую без учителя, сеть, тогда как более поздние сети обучаются с учителем.

[9] *Обратное распространение ошибки* (backpropagation) — это алгоритм градиентного спуска, применяемый в нейронных сетях для обновления весов. При этом сначала вычисляются веса последнего слоя, которые затем используются для обновления весов предыдущих слоев. Процесс продолжается до тех пор, пока не будут обновлены веса первого слоя.

Неконтролируемая часть сети известна как *жадное послойное предварительное обучение без учителя* (greedy layer-wise unsupervised pretraining). В 2006 году Джеффри Хинтон продемонстрировал успешное применение предварительного обучения без учителя для инициализации обучения конвейера глубокой нейронной сети, тем самым дав старт нынешней революции в области глубокого обучения. Предварительное обучение без учителя позволяет ИИ захватывать улучшенное представление оригинальных входных данных, преимуществами которого обучаемая с учителем часть сети может воспользоваться для решения текущей задачи.

Этот подход называют "жадным", поскольку каждая часть нейронной сети обучается независимо от других ее частей, а не совместно с ними. Для большинства современных нейронных сетей предварительное обучение обычно не требуется. Вместо этого все слои обучаются совместно посредством обратного распространения ошибки. Благодаря прогрессу в области компьютерных технологий справляться с проблемами затухающих и взрывных градиентов стало намного проще.

Предварительное обучение без учителя упростило решение не только задач обучения с учителем, но и задач *переносимого обучения* (transfer learning), в котором алгоритмы машинного обучения используются для сохранения знаний, полученных в процессе решения одной задачи, чтобы ускорить решение другой родственной задачи, причем на основе значительно меньшего объема данных.

Ограниченные машины Больцмана

Одним из прикладных примеров предварительного обучения без учителя может служить *ограниченная машина Больцмана* (restricted Boltzmann machine — RBM), представляющая собой мелкую двухслойную нейронную сеть. Ее первым слоем является входной слой, а вторым — скрытый. Каждый узел связан с каждым узлом другого слоя, но узлы в пределах одного и того же слоя не связаны между собой — это и есть то ограничение, которое фигурирует в названии данного метода.

RBM способна решать такие задачи обучения без учителя, как снижение размерности и извлечение признаков, а кроме того, подходит для использования в качестве составной части систем обучения с учителем, обеспечивающей предварительное обучение без учителя. RBM аналогичны автокодировщикам, но отличаются от них в ряде важных аспектов. Например, если в автокодировщиках предусмотрен выходной слой, то в RBM он отсутствует. К обсуждению этого и других отличий мы вернемся в последующих главах.

Глубокие сети доверия

RBM могут связываться между собой, образуя многозвенную нейронную сеть — так называемую *глубокую сеть доверия* (deep belief network — DBN). Скрытый слой каждой RBM используется в качестве входа для следующей RBM. Другими словами, каждая RBM генерирует представление данных, от которого затем отталкивается очередная RBM. Связывая между собой последовательные этапы обучения представлениям такого типа, глубокая сеть доверия способна обучаться более сложным представлениям, которые часто используются в качестве *детекторов признаков*[10].

Генеративно-состязательные сети

Одним из главных достижений в области глубокого обучения без учителя стали *генеративно-состязательные сети* (generative adversarial network — GAN), разработанные Яном Гудфеллоу с коллегами из Монреальского университета в 2014 году. Генеративно-состязательные сети имеют многочисленные применения; например, их можно использовать для создания высокореалистичных синтетических данных, таких как изображения или речь, либо для обнаружения аномалий.

GAN состоит из двух нейронных сетей. Одна из них, называемая *генератором*, генерирует данные на основе модельного распределения, которое она создает, используя выборки из реально полученных данных. Другая сеть, называемая *дискриминатором*, пытается отличить данные, созданные генератором, от данных, подчиняющихся истинному распределению.

Если воспользоваться простой аналогией и отвести генератору роль фальшивомонетчика, то дискриминатор можно уподобить криминалисту, пытающемуся отличить подделку. Эти две сети вовлечены в *игру с нулевой суммой* (zero-sum game). Генератор пытается ввести дискриминатор в заблуждение, чтобы тот думал, будто синтетические данные поступают из источника с истинным распределением данных, тогда как дискриминатор пытается выявить, что данные в действительности синтетические.

Генеративно-состязательные сети — это алгоритмы обучения без учителя, поскольку генератор способен изучить базовую структуру истинного распределения данных, даже если они не снабжены метками. Генеративно-состяза-

[10] Детекторы признаков обучаются подходящим представлениям исходных данных, способствующим выделению различающихся элементов. Например, в случае изображений детекторы признаков облегчают выделение таких элементов, как нос, глаза, рот и т.п.

тельные сети достигают этого в процессе тренировки, эффективно выявляя структуру с использованием умеренного количества параметров.

Этот процесс аналогичен процессу обучения представлениям посредством глубокого обучения. Каждый скрытый слой нейронной сети генератора захватывает представление базовых данных. Первоначальные представления очень простые, но каждый последующий слой усложняет представление, полученное в предыдущем слое.

Совместно используя все эти слои, генератор обучается базовой структуре данных и, опираясь на приобретенный опыт, пытается создать синтетические данные, которые были бы почти идентичны настоящим. Если генератору удается уловить суть истинного распределения данных, то синтетические данные будут казаться реальными.

Обработка последовательных данных с помощью обучения без учителя

Обучение без учителя позволяет также обрабатывать последовательные данные, например временны*е* ряды. Один из таких подходов предполагает обучение скрытым состояниям *марковской модели*. В *простой марковской модели* состояния полностью наблюдаемы и изменяются стохастически (другими словами — случайным образом). Будущие состояния зависят от текущего состояния и не зависят от предыдущих.

В *скрытой марковской модели* состояния являются лишь частично наблюдаемыми, но, подобно простым марковским моделям, выходы этих частично наблюдаемых состояний являются полностью наблюдаемыми. Поскольку имеющихся наблюдений недостаточно для полного определения состояния, приходится привлекать обучение без учителя, обеспечивающее более полное обнаружение всех скрытых состояний.

Алгоритмы скрытой марковской модели включают обучение вероятному следующему состоянию при условии, что известна последовательность ранее встречавшихся частично наблюдаемых состояний и полностью наблюдаемых выходов. Эти алгоритмы широко применяются для решения задач, связанных с обработкой речи, текста и временны*х* рядов.

Обучение с подкреплением с использованием обучения без учителя

Обучение с подкреплением (reinforcement learning) — третья из основных методологий машинного обучения, в соответствии с которой *агент* определяет свое оптимальное поведение (*действия*) в условиях *окружения* на основе обратной связи (получаемого *вознаграждения*). Эта обратная связь называется *сигналом подкрепления*. Целью агента является максимизация накапливаемого вознаграждения.

Несмотря на то что обучение с подкреплением было на слуху еще с 1950-х годов, в заголовках новостей оно стало фигурировать лишь в последние годы. В 2013 году компания DeepMind (в настоящее время куплена компанией Google) применила обучение с подкреплением для достижения результатов, превосходящих возможности человека, во многих играх на платформе Atari. Разработанной компанией DeepMind системе удалось добиться этого, используя в качестве входа лишь данные сенсорных датчиков без предварительного знания правил игры.

В 2016 году компания DeepMind вновь стала объектом внимания сообщества машинного обучения, на этот раз благодаря программе AlphaGo, победившей Ли Седоля, одного из лучших игроков в го. Эти успехи укрепили позиции обучения с подкреплением как одного из главных направлений в разработке ИИ.

Сегодня исследователи, работающие в области машинного обучения, применяют обучение с подкреплением для решения многих задач, включая следующие.

- Биржевая торговля, в ходе которой агент покупает и продает акции (действия), получая взамен прибыль или убытки (вознаграждение).
- Видео- и настольные игры, в которых агент принимает решения относительно ходов в игре (действия), в конечном счете приводящие к выигрышу или проигрышу (вознаграждение).
- Беспилотные автомобили, в которых агент управляет транспортным средством (действия) и либо нормально следует по своему маршруту, либо совершает дорожно-транспортное происшествие (вознаграждение).
- Управление роботом, когда агент передвигается среди объектов окружения (действия) и либо достигает конечного пункта, либо ему это не удается (вознаграждение).

В простейших случаях мы имеем дело с конечной задачей, т.е. с задачей, в которой имеется конечное число состояний окружения, конечное число действий, доступных при том или ином состоянии окружения, и конечное число вознаграждений. Действия, совершаемые агентом при заданном текущем состоянии окружения, определяют следующее действие, а целью агента является максимизации долгосрочного вознаграждения. Это семейство задач известно как *марковские процессы принятия решений с конечным числом состояний*.

Однако на практике не все так просто. Вознаграждение неизвестно и по своему характеру является динамической, а не статической величиной. Нашим помощником в установлении этой неизвестной функции вознаграждения и ее наилучшей аппроксимации может выступать обучение без учителя. Используя приближенную функцию вознаграждения, мы можем применить решения на основе обучения с подкреплением для увеличения накапливаемой величины вознаграждения.

Обучение с частичным привлечением учителя

Несмотря на то что обучение с учителем и обучение без учителя — две различные методологии машинного обучения, оба алгоритма могут совместно применяться в качестве звеньев конвейера машинного обучения[11]. В типичных случаях смесь алгоритмов обучения с учителем и без учителя используется в тех случаях, когда мы хотим в полной мере извлечь выгоду из немногих имеющихся меток или найти новые, еще неизвестные закономерности, исходя из неразмеченных данных, в дополнение к тем, которые были получены на основе размеченных данных. Задачи этого типа решаются путем использования гибридного варианта обучения с учителем и без учителя, получившего название *обучение с частичным привлечением учителя* (semisupervised learning). В последующих главах мы вернемся к более подробному обсуждению этой темы.

[11] Термин *конвейер* (pipeline) обозначает систему приложений машинного обучения, применяемых последовательно для достижения общей цели.

Успешные примеры обучения без учителя

За последние десять лет большинство наиболее успешных коммерческих применений машинного обучения было связано с использованием обучения с учителем, однако в настоящее время ситуация начала меняться. Все большее распространение получает обучение без учителя. В одних случаях обучение без учителя служит всего лишь средством, позволяющим улучшить приложения на основе обучения с учителем. В других случаях обучение без учителя уже само по себе выступает в качестве основы для построения коммерческих приложений. Ниже кратко описаны две наибольшие области применения обучения без учителя по состоянию на сегодняшний день.

Обнаружение аномалий

Снижение размерности позволяет редуцировать исходное многомерное пространство признаков в преобразованное пространство более низкой размерности, в котором мы находим области с высокой плотностью точек. Эти области образуют нормальное пространство. Точки, расположенные на гораздо бо́льших расстояниях, называются *выбросами*, или *аномалиями*, и заслуживают более пристального рассмотрения.

Как правило, системы обнаружения аномалий служат для выявления попыток мошенничества, связанных с использованием банковских карт, средств коммуникации и страховых полисов. Кроме того, обнаружение аномалий применяется для идентификации редких опасных событий, таких как взлом устройств, подключенных к Интернету, сбои в работе критического оборудования, например самолетов и поездов, и появление брешей в системах кибербезопасности из-за действий вредоносных программ.

Обнаружение аномалий можно использовать для распознавания спама, что уже обсуждалось нами ранее. К числу других применений относится обнаружение фактов финансирования терроризма, отмывания денег, торговли людьми, наркотиками и оружием, идентификация рискованных финансовых операций и диагностирование онкологических заболеваний.

Чтобы облегчить выявление аномалий, можно задействовать алгоритм кластеризации для группирования сходных аномалий с последующим назначением меток этим кластерам вручную на основании типов представляемого ими поведения. В состав такой системы можно включить интеллектуальный агент на основе обучения без учителя, способный обнаруживать аномалии, кластеризовать их в соответствующие группы и, используя метки,

присвоенные человеком, рекомендовать соответствующий порядок действий для бизнес-анализа.

В случае систем обнаружения аномалий мы можем воспользоваться описанным подходом на основе меток кластеров и, отталкиваясь от задачи, предназначенной для обучения без учителя, создать в конечном счете задачу для обучения с частичным привлечением учителя. Уже потом мы можем запустить алгоритмы обучения с учителем наряду с алгоритмами обучения без учителя. Чтобы приложения машинного обучения были успешными, системы обучения без учителя и с учителем должны применяться совместно, взаимно дополняя друг друга.

Система обучения с учителем находит известные закономерности с высокой степенью точности, тогда как система обучения без учителя обнаруживает новые закономерности, которые могут представлять для нас интерес. Выявив эти закономерности с помощью обучения без учителя, можно разметить данные вручную, переведя их из категории неразмеченных в категорию размеченных.

Сегментирование групп

С помощью кластеризации мы можем сегментировать группы на основании сходства их поведения в таких областях, как маркетинг, диагностика заболеваний, онлайн-покупки, прослушивание музыки, просмотр видеороликов, службы онлайн-знакомств, социальные сети и классификация документов. В каждом из этих случаев приходится иметь дело с огромными объемами данных, и эти данные размечены лишь частично.

Для уточнения уже известных закономерностей можно применять обучение с учителем. Однако зачастую мы хотим обнаруживать новые закономерности и группы, представляющие для нас интерес, и использование для этой цели обучения без учителя является вполне естественным вариантом выбора. Опять-таки, все дело в синергии. Залогом построения более надежных решений машинного обучения является совместное использование обучения с учителем и без учителя.

Резюме

В этой главе были рассмотрены следующие темы:

- различия между системами на основе правил и системами на основе машинного обучения;
- различия между обучением с учителем и без учителя;
- как обучение без учителя помогает справиться с проблемами, с которыми часто приходится сталкиваться в процессе тренировки моделей машинного обучения;
- распространенные алгоритмы обучения с учителем и без учителя, а также алгоритмы обучения с подкреплением и обучения с частичным привлечением учителя;
- две основные области применения обучения без учителя — обнаружение аномалий и сегментирование групп.

В главе 2 мы поговорим о том, как создавать приложения машинного обучения. Затем мы подробно рассмотрим методы снижения размерности и кластеризации, попутно создав систему обнаружения аномалий и систему сегментирования групп.