**Модельное обучение**

Другой способ обобщения на основе набора примеров - построить модель из этих примеров, а затем использовать эту модель для создания предсказания. Это называется модельное обучение ( Рисунок 1-16 ).

Рисунок 1-16. Модельное обучение

Например, предположим, что вы хотите знать, делают ли люди счастливыми людей, поэтому вы загружаете Индекс лучшей жизни данные из Сайт ОЭСР а также статистику ВВП на душу населения от Сайт МВФ . Затем вы присоединяетесь к таблицам и сортируете по ВВП на душу населения. Таблица 1-1 показывает отрывок из того, что вы получаете.

Таблица 1-1. Делают ли деньги людей счастливее?

Построим график данных для нескольких случайных стран ( Рисунок 1-17 ).

Рисунок 1-17. Вы видите здесь тенденцию?

Кажется, здесь действительно есть тенденция! Хотя данные шумный ( т. е. частично случайным образом), похоже, что удовлетворенность жизнью растет более или менее линейно по мере увеличения ВВП страны на душу населения. Итак, вы решили смоделировать удовлетворенность жизнью как линейную функцию ВВП на душу населения. Этот шаг называется выбор модели: вы выбрали линейная модель удовлетворенности жизнью всего с одним атрибутом - ВВП на душу населения (Уравнение 1-1 ).

Уравнение 1-1. Простая линейная модель

Эта модель имеет два параметры модели, θ 0 а также θ 1.5 Настраивая эти параметры, вы можете заставить вашу модель представлять любую линейную функцию, как показано на Рисунок 1-18.

Рисунок 1-18. Несколько возможных линейных моделей

Прежде чем вы сможете использовать свою модель, вам необходимо определить значения параметров θ 0 а также θ 1. Как узнать, какие значения позволят вашей модели работать лучше всего? Чтобы ответить на это вопрос, вам нужно указать меру производительности. Вы можете определить вспомогательная функция ( или фитнес-функция) это измеряет, как хороший ваша модель, или вы можете определить функция стоимости это измеряет, как плохой это. Для задач линейной регрессии люди обычно используют функцию стоимости, которая измеряет расстояние между предсказаниями линейной модели и примерами обучения; цель - минимизировать это расстояние.

Здесь на помощь приходит алгоритм линейной регрессии: вы скармливаете ему свои обучающие примеры, и он находит параметры, которые делают линейную модель наиболее подходящей для ваших данных. Это называется обучение модель. В нашем случае алгоритм находит, что оптимальная значения параметров θ 0 = 4.85 и θ 1 = 4,91 × 10 –5.

Теперь модель максимально соответствует обучающим данным (для линейной модели), как вы можете видеть на Рисунок 1-19 .

Рисунок 1-19. Линейная модель, которая лучше всего соответствует обучающим данным

Наконец-то вы готовы запустить модель, чтобы делать прогнозы. Например, вы хотите знать, насколько счастливы киприоты, но данные ОЭСР не дают ответа. К счастью, вы можете использовать свою модель, чтобы сделать хороший прогноз: вы просматриваете ВВП Кипра на душу населения, находите 22 587 долларов США, а затем применяете свою модель и обнаруживаете, что удовлетворенность жизнью, вероятно, будет где-то около 4,85 + 22 587 × 4,91 × 10- 5 = 5.96.

Чтобы подогреть аппетит, Пример 1-1 показывает код Python, который загружает данные, подготавливает их, 6 создает диаграмму рассеяния для визуализации, а затем обучает линейную модель и делает прогноз. 7

Пример 1-1. Обучение и запуск линейной модели с помощью Scikit-Learn

6. prepare\_country\_stats () определение функции здесь не показано (см. блокнот Jupyter в этой главе, если вам нужны все кровавые подробности). Это просто скучный код Pandas, который объединяет данные об удовлетворенности жизнью от ОЭСР с данными о ВВП на душу населения от МВФ.

7. Ничего страшного, если вы еще не понимаете весь код; мы представим Scikit-Learn в следующих главах

Если бы вместо этого вы использовали алгоритм обучения на основе экземпляров, вы бы обнаружили, что Словения имеет самый близкий ВВП на душу населения к Кипру (20 732 доллара США), а поскольку данные ОЭСР говорят нам, что удовлетворенность словенцами жизнью составляет 5,7, у вас будет предсказал удовлетворенность жизнью на Кипре на уровне 5,7. Если вы немного уменьшите масштаб и посмотрите на две ближайши страны, вы найдете Португалию и Испанию с уровнем удовлетворенности жизнью 5,1 и 6,5 соответственно. Усредняя эти три значения, вы получите 5,77, что довольно близко к прогнозу, основанному на вашей модели. Этот простой алгоритм называется k-Ближайшие соседи регрессия (в этом примере k = 3).

Заменить модель линейной регрессии регрессией k-ближайших соседей в предыдущем коде так же просто, как заменить эти две строки:

Если все пойдет хорошо, ваша модель будет делать хорошие прогнозы. В противном случае вам может потребоваться использовать больше атрибутов (уровень занятости, здоровье, загрязнение воздуха и т. Д.), Получить больше или лучше качественных данных обучения или, возможно, выбрать более мощную модель (например, модель полиномиальной регрессии).

В итоге:

• Вы изучили данные.

• Вы выбрали модель.

• Вы обучили его на обучающих данных (т. Е. Алгоритм обучения искал значения параметров модели, которые

минимизируют функцию стоимости).

• Наконец, вы применили модель для прогнозирования новых случаев (это называется

вывод), надеясь, что эта модель будет хорошо обобщать.

Вот как выглядит типичный проект машинного обучения. В Глава 2 вы испытаете это на собственном опыте, выполнив сквозной проект.

Мы рассмотрели многое: теперь вы знаете, что такое машинное обучение, почему оно полезно, каковы некоторые из наиболее распространенных категорий систем машинного обучения и как выглядит типичный рабочий процесс проекта. Теперь давайте посмотрим, что может пойти не так в обучении и помешать вам делать точные прогнозы.

**Основные проблемы машинного обучения**

Короче говоря, поскольку ваша основная задача - выбрать алгоритм обучения и обучить его на некоторых данных, две вещи, которые могут пойти не так, - это «плохой алгоритм» и «плохие данные». Начнем с примеров неверных данных.

**Недостаточное количество обучающих данных**

Чтобы малыш узнал, что такое яблоко, все, что вам нужно, - это указать на яблоко и сказать «яблоко» (возможно, повторив эту процедуру несколько раз). Теперь ребенок может узнавать яблоки самых разных цветов и форм. Гений.

Машинного обучения еще нет; Для правильной работы большинства алгоритмов машинного обучения требуется много данных. Даже для очень простых задач обычно нужны тысячи примеров, а для сложных задач, таких как распознавание изображений или речи, могут потребоваться миллионы примеров (если вы не можете повторно использовать части существующей модели).

**Необоснованная эффективность данных**

В известная газета опубликованная в 2001 году исследователями Microsoft Микеле Банко и Эриком Бриллом было показано, что очень разные алгоритмы машинного обучения, в том числе довольно простые, почти одинаково хорошо справляются со сложной задачей устранения неоднозначности естественного языка. 8 как только им будет предоставлено достаточно данных (как вы можете видеть на Рисунок 1-20 ).

Рисунок 1-20. Важность данных по сравнению с алгоритмами 9

По словам авторов, «эти результаты предполагают, что мы, возможно, захотим пересмотреть компромисс между тратой времени и денег на разработку алгоритмов по сравнению с их расходами на разработку корпуса».

Идея о том, что данные имеют большее значение, чем алгоритмы для решения сложных проблем, получила дальнейшую популяризацию Питера Норвига и др. в статье под названием «Необоснованная эффективность данных» опубликовано в 2009 году. 10 Однако следует отметить, что малые и средние наборы данных все еще очень распространены, и не всегда легко или дешево получить дополнительные обучающие данные, поэтому пока не отказывайтесь от алгоритмов.

8 Например, зная, писать ли «кому», «два» или «слишком» в зависимости от контекста.

9 Рисунок воспроизведен с разрешения Банко и Брилл (2001), «Кривые обучения для разоблачения набора путаницы».

10 «Неоправданная эффективность данных», Питер Норвиг и др. (2009)

**Нерепрезентативные данные обучения**

Для хорошего обобщения крайне важно, чтобы ваши обучающие данные были репрезентативными для новых случаев, для которых вы хотите обобщить. Это верно независимо от того, используете ли вы обучение на основе экземпляров или обучение на основе моделей.

Например, набор стран, который мы использовали ранее для обучения линейной модели, не был полностью репрезентативным; несколько стран отсутствовали. Рисунок 1-21 показывает, как выглядят данные, когда вы добавляете отсутствующие страны.

Рисунок 1-21. Более репрезентативная обучающая выборка

Если вы обучите линейную модель на этих данных, вы получите сплошную линию, в то время как старая модель будет представлена пунктирной линией. Как видите, добавление нескольких отсутствующих стран не только существенно меняет модель, но и дает понять, что такая простая линейная модель, вероятно, никогда не будет работать хорошо. Кажется, что очень богатые страны не более счастливы, чем умеренно богатые страны (на самом деле они кажутся несчастнее), и наоборот, некоторые бедные страны кажутся более счастливыми, чем многие богатые страны.

Используя нерепрезентативный обучающий набор, мы обучили модель, которая вряд ли сможет делать точные прогнозы, особенно для очень бедных и очень богатых стран.

Крайне важно использовать обучающий набор, который отражает те случаи, на которые вы хотите обобщить. Часто это сложнее, чем кажется: если образец слишком мал, у вас будет шум выборки ( т.е. непредставительные данные в результате случайности), но даже очень большие выборки могут быть нерепрезентативными, если метод выборки ошибочен. Это называется систематическая ошибка выборки.

**Известный пример смещения выборки**

Возможно, самый известный пример предвзятости выборки произошел во время президентских выборов в США в 1936 году, когда Лэндон был противопоставлен Рузвельту: Литературный дайджест провели очень крупный опрос, отправив почту примерно 10 миллионам человек. Он получил 2,4 миллиона ответов и с большой уверенностью предсказал, что Лэндон получит 57% голосов.

Вместо этого победил Рузвельт, набравший 62% голосов. Недостаток был в Литературный дайджест метод выборки:

• Во-первых, чтобы получить адреса для отправки опросов, Литературный дайджест пользовались телефонными справочниками, списками подписчиков журналов, списками членов клубов и т. д. Во всех этих списках предпочтение отдается более состоятельным людям, которые с большей вероятностью проголосуют за республиканцев (отсюда и Лэндон).

• Во-вторых, ответило менее 25% людей, принявших участие в опросе. Опять же, это вносит предвзятость выборки, исключая людей, которые не очень заботятся о политике, людей, которым не нравится Литературный дайджест, и другие ключевые группы. Это особый тип смещения выборки, называемый систематическая ошибка отсутствия ответов.

Вот еще один пример: предположим, вы хотите создать систему для распознавания видео в стиле фанк. Один из способов создать свой обучающий набор - поискать «фанк-музыку» на YouTube и использовать полученные видео. Но это предполагает, что поисковая система YouTube возвращает набор видео, которые представляют все музыкальные клипы фанк на YouTube. На самом деле результаты поиска, скорее всего, будут смещены в сторону популярных исполнителей (а если вы живете в Бразилии, вы получите много видео «фанк-кариока», которые не похожи на Джеймса Брауна). С другой стороны, как еще можно получить большой тренировочный набор?

**Данные низкого качества**

Очевидно, что если ваши данные обучения полны ошибок, выбросов и шума (например, из-за некачественных измерений), системе будет труднее обнаруживать лежащие в основе закономерности, поэтому ваша система с меньшей вероятностью будет работать хорошо. Часто стоит потратить время на очистку данных о тренировках. По правде говоря, большинство специалистов по данным тратят на это значительную часть своего времени. Например:

• Если некоторые экземпляры явно выбрасываются, их можно просто отбросить или попытаться исправить ошибки вручную.

• Если в некоторых экземплярах отсутствуют некоторые функции (например, 5% ваших клиентов не указали свой возраст), вы должны решить, хотите ли вы игнорировать этот атрибут в целом, игнорировать эти экземпляры, заполнить отсутствующие значения (например, со средним возрастом), или обучите одну модель с функцией, а другую - без нее, и так далее.

**Нерелевантные функции**

Как говорится: мусор на входе, мусор на выходе. Ваша система будет способна к обучению, только если обучающие данные содержат достаточно релевантных функций и не слишком много нерелевантных. Важнейшая часть успеха проекта машинного обучения - это хороший набор функций для обучения. Этот процесс, называемый разработка функций, включает:

• Выбор функции: выбор наиболее полезных функций для обучения среди существующих функций.

• Извлечение признаков: объединение существующих функций для создания более полезных (как мы видели ранее, могут помочь алгоритмы уменьшения размерности).

• Создание новых функций путем сбора новых данных.

Теперь, когда мы рассмотрели множество примеров плохих данных, давайте рассмотрим пару примеров плохих алгоритмов.

**Переоснащение обучающих данных**

Допустим, вы приехали в чужую страну, и водитель такси ограбил вас. У вас может возникнуть соблазн сказать, что все таксисты в этой стране - воры. Чрезмерное обобщение - это то, что мы, люди, делаем слишком часто, и, к сожалению, машины могут попасть в ту же ловушку, если мы не будем осторожны. В машинном обучении это называется переоснащение: это означает, что модель хорошо работает на обучающих данных, но плохо обобщается.

Рисунок 1-22 показывает пример модели высокой полиномиальной удовлетворенности жизнью, которая сильно превышает данные обучения. Несмотря на то, что она работает с обучающими данными намного лучше, чем простая линейная модель, вы действительно доверяете ее предсказаниям?

Рисунок 1-22. Переоснащение обучающих данных

Сложные модели, такие как глубокие нейронные сети, могут обнаруживать тонкие закономерности в данных, но если обучающий набор зашумлен или слишком мал (что приводит к шуму выборки), то модель, вероятно, сама обнаружит закономерности в шуме. Очевидно, что эти шаблоны не будут распространяться на новые случаи. Например, предположим, что вы добавляете в свою модель удовлетворенности жизнью множество других атрибутов, в том числе неинформативных, таких как название страны. В этом случае сложная модель может обнаруживать закономерности, такие как тот факт, что все страны в обучающих данных с ш от их имени удовлетворенность жизнью выше 7: Новая Зеландия (7,3), Норвегия (7,4), Швеция (7,2) и Швейцария (7,5). Как уверенно Вы считаете, что правило W-удовлетворенности распространяется на Руанду или Зимбабве? Очевидно, что этот шаблон возник в обучающих данных по чистой случайности, но модель не может сказать, является ли шаблон реальным или просто результатом шума в данных.

Переобучение происходит, когда модель слишком сложна по сравнению с количеством и шумностью обучающих данных. Возможные решения:

• Упростить модель, выбрав модель с меньшим количеством параметров (например, линейную модель, а не полиномиальную модель высокой степени), уменьшив количество атрибутов в обучающих данных или ограничив модель

• Чтобы собрать больше данных о тренировках

• Чтобы уменьшить шум в обучающих данных (например, исправить ошибки данных и удалить выбросы).

Ограничение модели для упрощения и снижения риска переобучения называется регуляризация. Например, линейная модель, которую мы определили ранее, имеет два параметра: θ 0 а также θ 1. Это дает алгоритму обучения два степени свободы чтобы адаптировать модель к обучающим данным: она может настраивать как высоту ( θ 0) и наклон ( θ 1) линии. Если мы заставили θ 1 = 0, алгоритм будет иметь только одну степень свободы, и ему будет намного сложнее правильно подобрать данные: все, что он может сделать, это переместить строку вверх или вниз, чтобы максимально приблизиться к обучающим экземплярам, чтобы в итоге получилось вокруг среднего. Действительно, очень простая модель! Если мы позволим алгоритму модифицировать θ 1 но мы заставляем его держать его маленьким, тогда алгоритм обучения будет эффективно иметь некоторые где между одной и двумя степенями свободы. Это позволит создать более простую модель, чем с двумя степенями свободы, но более сложную, чем с одной. Вы хотите найти правильный баланс между точной подгонкой обучающих данных и сохранением достаточно простой модели, чтобы обеспечить ее хорошее обобщение.

Рисунок 1-23 показаны три модели: пунктирная линия представляет исходную модель, которая была обучена с отсутствием нескольких стран, пунктирная линия - наша вторая модель, обученная со всеми странами, а сплошная линия – линейная модель, обученная с теми же данными, что и первая модель, но с ограничением регуляризации. Вы можете видеть, что регуляризация вынудила модель иметь меньший наклон, что немного меньше соответствует обучающим данным, на которых была обучена модель, но на самом деле позволяет лучше обобщать ее для новых примеров.

Рисунок 1-23. Регуляризация снижает риск переобучения

Объем регуляризации, применяемой во время обучения, можно контролировать с помощью гиперпараметр. Гиперпараметр - это параметр алгоритма обучения (а не модели). Таким образом, на него не влияет сам алгоритм обучения; он должен быть установлен до тренировки и остается постоянным во время тренировки. Если вы установите гиперпараметр регуляризации на очень большое значение, вы получите почти плоскую модель (наклон, близкий к нулю); алгоритм обучения почти наверняка не переоценивает обучающие данные, но вряд ли найдет хорошее решение. Настройка гиперпараметров - важная часть построения системы машинного обучения (подробный пример вы увидите в следующей главе).

**Недостаток тренировочных данных**

Как нетрудно догадаться, недостаточное оснащение является противоположностью переобучения: это происходит, когда ваша модель слишком проста для изучения базовой структуры данных. Например, линейная модель удовлетворенности жизнью не соответствует требованиям; реальность просто сложнее модели, поэтому ее прогнозы обязательно будут неточными даже на обучающих примерах.

Основные варианты решения этой проблемы:

• Выбор более мощной модели с большим количеством параметров

• Добавление лучших функций в алгоритм обучения (разработка функций)

• Уменьшение ограничений на модель (например, уменьшение гиперпараметра регуляризации)

**Отступая**

К настоящему времени вы уже много знаете о машинном обучении. Однако мы рассмотрели так много концепций, что вы можете почувствовать себя немного потерянным, поэтому давайте вернемся назад и посмотрим на общую картину:

• Машинное обучение помогает машинам лучше справляться с некоторыми задачами, обучаясь на основе данных, вместо того, чтобы явно кодировать правила.

• Существует много разных типов систем машинного обучения: контролируемые или неконтролируемые, пакетные или онлайновые, экземпляры или модели и т. д.

• В проекте машинного обучения вы собираете данные в обучающий набор и передаете его в обучающий алгоритм. Если алгоритм основан на модели, он настраивает некоторые параметры, чтобы подогнать модель под обучающий набор (т. е. Чтобы делать хорошие прогнозы на самом обучающем наборе), а затем, надеюсь, он сможет делать хорошие прогнозы и для новых случаев. Если алгоритм основан на экземплярах, он просто изучает примеры наизусть и обобщает их на новые экземпляры, сравнивая их с изученными экземплярами с использованием меры сходства.

• Система не будет работать хорошо, если ваша обучающая выборка слишком мала, или если данные не репрезентативны, зашумлены или загрязнены нерелевантными функциями (мусор на входе, мусор на выходе). И, наконец, ваша модель не должна быть ни слишком простой (в этом случае она не подходит), ни слишком сложной (в этом случае она может быть избыточной).

Остается затронуть еще одну важную тему: обучив модель, вы не хотите просто «надеяться» на ее обобщение на новые случаи. Вы хотите его оценить и при необходимости настроить. Посмотрим как.

**Тестирование и проверка.**

Единственный способ узнать, насколько хорошо модель будет обобщена на новые случаи, - это попробовать ее на новых случаях. Один из способов сделать это - запустить вашу модель в производство и контролировать, насколько хорошо она работает. Это работает хорошо, но если ваша модель ужасно плохая, пользователи будут жаловаться - не лучшая идея.

Лучше разделить данные на два набора: Обучающий набор и набор тестов. Как следует из этих названий, вы обучаете свою модель, используя обучающий набор, и тестируете ее, используя тестовый набор. Частота ошибок в новых случаях называется ошибка обобщения ( или же ошибка вне выборки), и, оценив вашу модель на тестовом наборе, вы получите оценку этой ошибки. Это значение говорит вам, насколько хорошо ваша модель будет работать на экземплярах, которых она никогда раньше не видела.

Если ошибка обучения мала (т. е. Ваша модель допускает мало ошибок на обучающем наборе), но ошибка обобщения высока, это означает, что ваша модель перенасыщается обучающими данными.

Обычно 80% данных используются для обучения и продержаться 20% на тестирование. Однако это зависит от размера набора данных: если он содержит 10 миллионов экземпляров, то удержание 1% означает, что ваш тестовый набор будет содержать 100 000 экземпляров: этого, вероятно, более чем достаточно, чтобы получить хорошую оценку ошибки обобщения.

**Настройка гиперпараметров и выбор модели.**

Итак, оценить модель достаточно просто: просто используйте набор тестов. Теперь предположим, что вы колеблетесь между двумя моделями (скажем, линейной моделью и полиномиальной моделью): как вы можете решить? Один из вариантов - обучить обоих и сравнить, насколько хорошо они обобщают, используя набор тестов.

Теперь предположим, что линейная модель обобщает лучше, но вы хотите применить некоторую регуляризацию, чтобы избежать переобучения. Возникает вопрос: как выбрать значение гиперпараметра регуляризации? Один из вариантов - обучить 100 различных моделей, используя 100 различных значений этого гиперпараметра. Предположим, вы нашли лучшее значение гиперпараметра, которое дает модель с наименьшей ошибкой обобщения, скажем, всего 5%.

Итак, вы запускаете эту модель в производство, но, к сожалению, она работает не так хорошо, как ожидалось, и дает 15% ошибок. Что только что произошло?

Проблема в том, что вы несколько раз измерили ошибку обобщения на тестовом наборе и адаптировали модель и гиперпараметры для получения наилучшей модели. для этого конкретного набора. Это означает, что модель вряд ли будет работать на новых данных.

Общее решение этой проблемы называется проверка удержания: вы просто протягиваете часть обучающего набора, чтобы оценить несколько моделей-кандидатов и выбрать лучшую. Новый удерживаемый набор называется набор проверки ( или иногда набор для разработки, или же набор разработчиков). В частности, вы обучаете несколько моделей с различными гиперпараметрами на сокращенном обучающем наборе (т. Е. Полный обучающий набор за вычетом проверочного набора) и выбираете модель, которая лучше всего работает на проверочном наборе. После этого процесса проверки удержания вы обучаете лучшую модель на полном обучающем наборе (включая набор проверки), и это дает вам окончательную модель. Наконец, вы оцениваете эту окончательную модель на тестовом наборе, чтобы получить оценку ошибки обобщения.

Это решение обычно работает достаточно хорошо. Однако, если набор для проверки слишком мал, оценка модели будет неточной: вы можете по ошибке выбрать неоптимальную модель. И наоборот, если набор для проверки слишком велик, оставшийся обучающий набор будет намного меньше, чем полный обучающий набор. Почему это плохо? Что ж, поскольку окончательная модель будет обучаться на полном обучающем наборе, сравнивать модели-кандидаты, обученные на гораздо меньшем обучающем наборе, не идеально. Это все равно что выбрать самого быстрого спринтера для участия в марафоне. Один из способов решить эту проблему - повторить перекрестная проверка, используя множество небольших проверочных наборов. Каждая модель оценивается один раз для каждого набора проверки после обучения на остальных данных. Усредняя все оценки модели, мы получаем гораздо более точную оценку ее производительности. Однако есть недостаток: время обучения умножается на количество проверочных наборов.

Несоответствие данных.

В некоторых случаях легко получить большой объем данных для обучения, но он не является полностью репрезентативным для данных, которые будут использоваться в производственной среде. Например, предположим, что вы хотите создать мобильное приложение, чтобы фотографировать цветы и автоматически определять их вид. Вы можете легко загрузить миллионы изображений цветов в Интернете, но они не будут полностью соответствовать изображениям, которые фактически будут сделаны с помощью приложения на мобильном устройстве. Возможно, у вас есть только 10 000 репрезентативных снимков (т. Е. Фактически сделанных с помощью приложения). В этом случае самое важное правило, которое следует помнить, заключается в том, что набор для проверки и тест должны быть как можно более репрезентативными для данных, которые вы ожидаете использовать в производстве, поэтому они должны состоять исключительно из репрезентативных изображений: вы можете перетасовать их и поместить половина в проверочном наборе, и половина в тестовом наборе (убедитесь, что в обоих наборах нет дубликатов или почти дубликатов). После обучения вашей модели на веб-изображениях, если вы заметите, что производительность вашей модели на проверочном наборе неутешительна, вы не узнаете, связано ли это с тем, что ваша модель превосходит обучающий набор, или это просто из-за несоответствие между изображениями в Интернете и изображениями в мобильном приложении. Одно из решений - поместить часть обучающих изображений (из Интернета) в еще один набор, который Эндрю Нг называет или это просто из-за несоответствия между изображениями в Интернете и изображениями мобильного приложения. Одно из решений - поместить часть обучающих изображений (из Интернета) в еще один набор, который Эндрю Нг называет или это просто из-за несоответствия между изображениями в Интернете и изображениями мобильного приложения. Одно из решений - поместить часть обучающих изображений (из Интернета) в еще один набор, который Эндрю Нг называет набор train-dev. После обучения модели (на обучающей выборке нет в наборе train-dev), вы можете оценить его на наборе train-dev: если он работает хорошо, значит, модель не переоснащается обучающим набором поэтому, если он плохо работает на наборе проверки, проблема должна исходить из данных несоответствие. Вы можете попытаться решить эту проблему, предварительно обработав веб-изображения, чтобы они больше походили на изображения, которые будут сделаны мобильным приложением, а затем повторно обучив модель. И наоборот, если модель плохо работает на наборе train-dev, то модель должна превосходить обучающий набор, поэтому вам следует попытаться упростить или упорядочить модель, получить больше обучающих данных и очистить обучающие данные, как обсуждалось ранее.

**Теорема об отсутствии бесплатного обеда**

Модель - это упрощенная версия наблюдений. Упрощения предназначены для того, чтобы отбросить лишние детали, которые вряд ли можно будет обобщить на новые примеры. Однако, чтобы решить, какие данные удалить, а какие сохранить, вы должны предположения. Например, линейная модель предполагает, что данные принципиально линейны и что расстояние между экземплярами и прямой линией - это просто шум, который можно спокойно игнорировать.

В знаменитой газете 1996 года , 11 Дэвид Вулперт продемонстрировал, что если вы не делаете абсолютно никаких предположений относительно данных, то нет причин предпочитать одну модель любой другой. Это называется нет бесплатного обеда (NFL) теорема. Для некоторых наборов данных лучше всего модель - это линейная модель, а для других наборов данных - нейронная сеть. Нет модели, которая априори гарантированно работает лучше (отсюда и название теоремы). Единственный способ узнать наверняка, какая модель лучше, - это оценить их все. Поскольку это невозможно, на практике вы делаете некоторые разумные предположения о данных и оцениваете только несколько разумных моделей. Например, для простых задач вы можете оценивать линейные модели с различными уровнями регуляризации, а для сложных задач вы можете оценивать различные нейронные сети.

**ГЛАВА 2 Проект сквозного машинного обучения.**

С электронными книгами раннего выпуска вы получаете книги в их самой ранней форме - необработанное и неотредактированное содержание автора по мере того, как он или она пишет, - так что вы можете воспользоваться этими технологиями задолго до официального выпуска этих книг. Следующая глава будет второй в финальном выпуске книги.

В этой главе вы от конца до конца пройдете примерный проект, притворившись недавно нанятым специалистом по обработке данных в компанию, занимающуюся недвижимостью. Вот основные шаги, которые вам предстоит пройти:

1. Взгляните на картину в целом.

2. Получите данные.

3. Откройте для себя и визуализируйте данные, чтобы получить представление.

4. Подготовьте данные для алгоритмов машинного обучения.

5. Выберите модель и обучите ее.

6. Выполните точную настройку вашей модели.

7. Представьте свое решение.

8. Запускайте, отслеживайте и обслуживайте вашу систему.

**Работа с реальными данными**

Когда вы изучаете машинное обучение, лучше всего поэкспериментировать с реальными данными, а не только с искусственными наборами данных. К счастью, есть тысячи открытых наборов данных на выбор, охватывающих все виды доменов. Вот несколько мест, где можно получить данные:

• Популярные репозитории открытых данных:

- Репозиторий машинного обучения UC Irvine

- Наборы данных Kaggle

- Наборы данных Amazon AWS

• Мета-порталы (в них перечислены репозитории открытых данных):

- http://dataportals.org/

- http://opendatamonitor.eu/

- http://quandl.com/

• Другие страницы со списком многих популярных репозиториев открытых данных:

- Список наборов данных машинного обучения в Википедии

- Quora.com вопрос

- Субреддит наборов данных.

В этой главе мы выбрали набор данных о ценах на жилье в Калифорнии из репозитория StatLib(Рисунок 2-1). Этот набор данных был основан на данных Калифорнийской переписи 1990 года. Это не совсем недавно (в то время вы все еще могли позволить себе хороший дом в районе залива), но у него есть много качеств для обучения, поэтому мы будем притворяться, что это свежие данные. Мы также добавили категориальный атрибут и удалили несколько функций в учебных целях.

Рисунок 2-1. Цены на жилье в Калифорнии.

**Взгляните на картину в целом.**

Добро пожаловать в Корпорацию по машинному обучению! Первое, что вас просят выполнить, - это построить модель цен на жилье в Калифорнии, используя данные калифорнийской переписи. Эти данные содержат такие показатели, как численность населения, средний доход, медианная цена жилья и т. д. Для каждой группы кварталов в Калифорнии. Блочные группы - это наименьшая географическая единица, для которой Бюро переписи США публикует выборочные данные (блочная группа обычно насчитывает от 600 до 3000 человек). Мы будем для краткости называть их просто «округами».

Ваша модель должна учиться на этих данных и быть способной предсказать среднюю стоимость жилья в любом районе с учетом всех остальных показателей.

Поскольку вы хорошо организованный специалист по данным, первое, что вы делаете, - это вытаскиваете контрольный список для проекта машинного обучения. Вы можете начать с того, что в ??? ; он должен работать достаточно хорошо для большинства проектов машинного обучения, но не забудьте адаптировать его к вашим потребностям. В этой главе мы пройдемся по многим пунктам контрольного списка, но мы также пропустим некоторые, потому что они не требуют пояснений или потому, что они будут обсуждаться в следующих главах.

**Сформулируйте проблему.**

Первый вопрос, который следует задать своему боссу, - какова именно цель бизнеса; создание модели, вероятно, не является конечной целью. Как компания ожидает от использования и выгоды из этой модели? Это важно, потому что это определит, как вы сформулируете проблему, какие алгоритмы вы выберете, какой показатель производительности вы будете использовать для оценки своей модели и сколько усилий вы должны потратить на ее настройку.

Ваш начальник отвечает, что результаты вашей модели (прогноз средней цены жилья в округе) будут переданы в другую систему машинного обучения (см. Рисунок 2-2 ), наряду со многими другими сигналы. Последующая система определит, стоит ли инвестировать в данную область или нет. Правильное решение этого вопроса имеет решающее значение, поскольку оно напрямую влияет на доход.

Рисунок 2-2. AMachine Learning для инвестиций в недвижимость.

**Pipelines**

Последовательность обработки данных составныx частей называются данными конвейера. Конвейеры очень распространены в системах машинного обучения, так как есть много данных, которыми нужно манипулировать, и множество преобразований данных, которые необходимо применить. Компоненты обычно работают асинхронно. Каждый компонент извлекает большой объем данных, обрабатывает его и выводит результат в другое хранилище данных, а затем через некоторое время следующий компонент в конвейере извлекает эти данные и выдает свой собственный вывод и так далее. Каждый компонент достаточно самодостаточен: интерфейс между компонентами - это просто хранилище данных. Это делает систему довольно простой для понимания (с помощью графа потока данных), и разные команды могут сосредоточиться на разных компонентах. Более того, если компонент выходит из строя, нижестоящие компоненты часто могут продолжать нормально работать (по крайней мере, какое-то время), просто используя последний вывод сломанного компонента. Это делает архитектуру достаточно надежной.

С другой стороны, неисправный компонент может какое-то время оставаться незамеченным, если не будет реализован надлежащий мониторинг. Данные устаревают, и общая производительность системы падает.

Следующий вопрос, который нужно задать, - как выглядит текущее решение (если есть). Это часто дает вам эталонную производительность, а также дает представление о том, как решить проблему. Ваш босс отвечает, что районные цены на жилье в настоящее время оцениваются экспертами вручную: группа собирает самую свежую информацию о районе, а когда они не могут получить среднюю стоимость жилья, они оценивают ее по сложным правилам.

Это дорого и требует много времени, а их оценки невелики; в тех случаях, когда им удается узнать фактическую медианную цену жилья, они часто понимают, что их оценки отклонились более чем на 20%. Вот почему компания считает, что было бы полезно обучить модель для прогнозирования средней цены жилья в районе с учетом других данных об этом районе. Данные переписи выглядят как отличный набор данных для использования с этой целью, поскольку он включает медианные цены на жилье в тысячах районов, а также другие данные.

Хорошо, со всей этой информацией теперь вы готовы приступить к проектированию своей системы. Во-первых, вам нужно сформулировать проблему: контролируемое, неконтролируемое или подкрепляющее обучение? Это задача классификации, задача регрессии или что-то еще? Что следует использовать: пакетное обучение или методы онлайн-обучения? Прежде чем продолжить чтение, сделайте паузу и попытайтесь ответить на эти вопросы для себя.

Вы нашли ответы? Посмотрим: очевидно, что это типичное контролируемое учебное задание, поскольку вам дают помеченный обучающие примеры (каждый пример имеет ожидаемый результат, т. е. среднюю стоимость жилья в округе). Более того, это также типичная задача регрессии, поскольку вас просят предсказать значение. В частности, это множественная регрессия проблема, поскольку система будет использовать несколько функций для прогнозирования (она будет использовать население района, средний доход и т. д.). Это также одномерная регрессия проблема, поскольку мы пытаемся предсказать только одно значение для каждого района. Если бы мы пытались предсказать несколько значений для каждого района, это было бы многомерная регрессия проблема. Наконец, нет непрерывного потока данных, поступающих в систему, нет особой необходимости приспосабливаться к быстрому изменению данных, а данные достаточно малы, чтобы поместиться в памяти, поэтому простое пакетное обучение должно подойти.

Если объем данных был огромным, вы могли либо разделить свою работу по пакетному обучению между несколькими серверами (используя MapReduce техника), или вместо этого вы можете использовать технику онлайн-обучения.

**Выберите показатель эффективности.**

Следующим шагом будет выбор показателя эффективности. Типичным показателем производительности для регрессионных моделей является среднеквадратическая ошибка (RMSE). Он дает представление о том, сколько ошибок система обычно допускает в своих прогнозах, с большим весом для больших ошибок. Уравнение 2-1 показывает математическую формулу для вычисления RMSE.

Уравнение 2-1. Среднеквадратичная ошибка (RMSE) .