Вводный семинар

Евгений Соколов

5 сентября 2015 г.

1 Введение в анализ данных

Анализ данных, или машинное обучение — это наука, изучающая способы извлечения закономерностей из ограниченного количества примеров. На самом деле, анализ данных и машинное обучение немного отличаются, и существует большое количество мнений о том, чем именно. Одно из них заключается в том, что машинное обучение посвящено строгому изучению методов извлечения закономерностей из данных, а анализ данных — скорее название ремесла, направленного на решение прикладных задач. Мы не будем заострять внимание на этих тонкостях, а перейдем к конкретному примеру.

Представим, что нам принадлежит большая сеть ресторанов, и мы хотим открыть в неком городе новое заведение [1]. Мы нашли несколько точек в городе, где есть возможность приобрести помещение и организовать там ресторан. Нам важно, чтобы через определенное время он стал приносить прибыль — точнее, хочется открыть его в той точке, в которой прибыль окажется максимальной. Поставим задачу: для каждого возможного размещения ресторана предсказать прибыль, которую он принесет в течение первого года. Множество всех возможных точек размещения называется *пространством объектов* и обозначается через $\mathbb{X}$. Величина, которую мы хотим определять (т.е. прибыль ресторана), называется *целевой переменной*, а множество ее значений — пространством ответов $\mathbb{Y}$. В нашем случае пространство ответов является множеством вещественных чисел: $\mathbb{Y} = \mathbb{R}$.

Мы не являемся специалистами в экономике, поэтому не можем сделать такие прогнозы на основе своих экспертных знаний. У нас есть лишь примеры — поскольку мы владеем целой сетью ресторанов, то имеем данные по достаточно большому числу ранее открытых ресторанов и по их прибыли в течение первого года. Каждый такой пример называется *обучающим*, а вся их совокупность — *обучающей выборкой*, которая обозначается как $X^\ell = \{x_1, \dots, x_\ell\}$, где $x_1, \dots, x_\ell$ — обучающие объекты, а ℓ — их количество. Особенность обучающих объектов состоит в том, что для них известны ответы $y_1, \dots, y_\ell$.

Отметим, что объекты — это некие абстрактные сущности (точки размещения ресторанов), которыми компьютеры не умеют оперировать напрямую. Для дальнейшего анализа нам понадобится описать объекты с помощью некоторого набора характеристик, которые называются *признаками* (или факторами). Вектор всех признаков

[1] Задача по мотивам конкурса Restaurant Revenue Prediction: https://www.kaggle.com/c/restaurant-revenue-prediction

объекта x называется *признаковым описанием* этого объекта. Далее мы будем отождествлять объект и его признаковое описание. Признаки могут быть очень разными: бинарными, вещественными, категориальными (принимают значения из неупорядоченного множества), ординальными (принимают значения из упорядоченного множества), множествозначными (set-valued, значения являются подмножествами некоторого универсального множества). Признаки могут иметь сложную внутреннюю структуру: так, в качестве признака человека в задаче предсказания годового дохода может служить его фотография. Разумеется, фотографию можно представить и как некоторое количество бинарных или вещественных признаков, каждый из которых кодирует соответствующие пиксель изображения. Однако, работа с изображением как с одной сложной структурой позволяет вычислять по нему различные фильтры, накладывать требование инвариантности ответа к сдвигам и т.д. — именно на этом основано активно развивающееся сейчас *глубокое обучение* (deep learning).

В нашей задаче полезными могут оказаться признаки, связанные с демографией (средний возраст жителей ближайших кварталов, динамика изменения количества жителей) или недвижимостью (например, средняя стоимость квадратного метра в окрестности, количество школ, магазинов, заправок, торговых центров, банков поблизости). Разработка признаков (feature engineering) для любой задачи является одним из самых сложных и самых важных этапов анализа данных.

Описанная задача является примером задачи *обучения с учителем (supervised learning)*, а более конкретно задачей *регрессии* — именно так называются задачи с вещественной целевой переменной. Перечислим другие виды задач обучения с учителем:

1. $\mathbb{Y} = \{0, 1\}$ — бинарная классификация. Например, мы можем предсказывать, кликнет ли пользователь по рекламному объявлению, вернет ли клиент кредит в установленный срок, сдаст ли студент сессию, случится ли определенное заболевание с пациентом (на основе его генома).

2. $\mathbb{Y} = \{1, \ldots, M\}$ — многоклассовая (multi-class) классификация. Примером может служить определение предметной области для научной статьи (математика, биология, психология и т.д.).

3. $\mathbb{Y} = \{0, 1\}^M$ — многоклассовая классификация с пересекающимися классами (multi-label classification). Примером может служить задача медицинской диагностики, где для пациента нужно определить набор заболеваний, которыми он страдает.

4. Ранжирование — задача, в которой требуется восстановить порядок на некотором множестве объектов. Основным примером является задача ранжирования поисковой выдачи, где для любого запроса нужно отсортировать все возможные документы по релевантности этому запросу.

5. Частичное обучение (semi-supervised learning) — задача, в которой для одной части объектов обучающей выборки известны и признаки, и ответы, а для другой только ответы. Такие ситуации возникают, например, в медицинских задачах, где получение ответа является крайне сложным (например, требует проведения дорогостоящего анализа).

Существует также обучение без учителя — класс задач, где ответы неизвестны или вообще не существуют, и требуется найти некоторые закономерности в данных лишь на основе признаковых описаний:

1. Кластеризация — задача разделения объектов на группы, обладающие некоторыми свойствами. Примером может служить кластеризация документов из электронной библиотеки или кластеризация абонентов мобильного оператора.

2. Оценивание плотности — задача приближения распределения объектов. Примером может служить задача обнаружения аномалий, в которой на этапе обучения известны лишь примеры «правильного» поведения оборудования (или, скажем, игроков на бирже), а в дальнейшем требуется обнаруживать случаи некорректной работы (соответственно, незаконного поведения игроков). В таких задачах сначала оценивается распределение «правильных» объектов, а затем аномальными объявляются все объекты, которых в рамках этого распределения получают слишком низкую вероятность.

3. Визуализация — задача изображения многомерных объектов в двумерном или трехмерном пространстве таким образом, что сохранялось как можно больше зависимостей и отношений между ними.

4. Понижение размерности — задача генерации таких новых признаков, что их меньше, чем исходных, но при этом с их помощью задача решается не хуже (или с небольшими потерями качества, или лучше — зависит от постановки). К этой же категории относится задача построения латентных моделей, где требуется описать процесс генерации данных с помощью некоторого (как правило, небольшого) набора скрытых переменных. Примерами являются задачи тематического моделирования и построения рекомендаций, которым будет посвящена часть курса.

Вернемся к нашей задаче по предсказанию прибыли ресторана. Предположим, что мы собрали обучающую выборку и изобрели некоторое количество признаков. Результатом будет матрица «объекты-признаки» $X \in \mathbb{R}^{\ell \times d}$ (ℓ — число объектов, d — число признаков), в которой каждая строка содержит признаковое описание одного из обучающих объектов. Таким образом, строки в этой матрице соответствуют объектам, а столбцы — признакам.

Нашей задачей является построение функции $a : \mathbb{X} \to \mathbb{Y}$, которая для любого объекта будет предсказывать ответ. Такая функция называется *алгоритмом* или *моделью* (hypothesis). Понятно, что нам подойдет далеко не каждый алгоритм — например, вряд ли мы извлечем какую-то выгоду из алгоритма $a(x) = 0$, который предсказывает нулевую прибыль для любого ресторана независимо от его признаков. Чтобы формализовать соответствие алгоритма нашим ожиданиям, нужно ввести *функционал качества* [2], измеряющий качество работы алгоритма. Крайне популярным функционалом в задаче регрессии является среднеквадратичная ошибка (mean squared

[2] Достаточно часто употребляется термин «метрика качества», но он является не очень удачным, поскольку вызывает лишние ассоциации с метриками, обсуждаемыми в алгебре. Тем не менее, мы иногда будем использовать этот термин для разнообразия.

error, MSE):

$$Q(a, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}\left(a(x_i) - y_i\right)^2.$$

Чем более маленькое значение этого функционала дает алгоритм, тем он лучше. Данный функционал является очень удобным благодаря дифференцируемости и простоте, но, как мы увидим дальше в курсе, обладает и большим количеством недостатков.

В большинстве случаев функционалы представляют собой сумму ошибок на отдельных объектах обучающей выборки. Функция, измеряющая ошибку одного предсказания, называется *функцией потерь* $L : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_+$ (в нашем случае $L(y, z) = = (y - z)^2$.

Заметим, что именно функционал качества будет определять во всех дальнейших рассуждениях, какой алгоритм является лучшим. Если метрика выбрана неудачна и не соответствует бизнес-требованиям или особенностям данных, то все дальнейшие действия обречены на провал. Именно поэтому выбор базовой метрики является крайне важным этапом в решении любой задачи анализа данных. Она не обязательно должна обладать хорошими математическими свойствами (непрерывности, выпуклость, дифференцируемость и т.д.), но обязана отражать все важные требования к решению задачи.

Как только функционал качества зафиксирован, можно приступать к построению алгоритма $a(x)$. Как правило, для этого фиксируют некоторое *семейство алгоритмов* $\mathcal{A}$, и пытаются выбрать из него алгоритм, наилучший с точки зрения функционала [3]. В машинном обучении было изобретено большое количество семейств алгоритмов, и, наверное, самым простым и самым тщательно изученным среди них является семейство *линейных моделей*, которые дают предсказание, равное линейной комбинации признаков:

$$\mathcal{A} = \{a(x) = w_0 + w_1 x^1 + \cdots + w_d x^d \,|\, w_0, w_1, \ldots, w_d \in \mathbb{R}\},$$

где через x^i обозначается значение i-го признака у объекта x. Лучшая из таких моделей будет выбираться путем минимизации MSE-функционала:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\left(w_0 + \sum_{j=1}^{d} w_j x_i^j - y_i\right)^2 \to \min_{w_0, w_1, \ldots, w_d}.$$

Процесс поиска оптимального алгоритма называется *обучением*.

Зачастую возникает потребность в *предобработке данных* до начала построения модели. Здесь может идти речь о некотором ряде манипуляций:

- Некоторые модели хорошо работают только при выполнении определенных требований. Так, для линейных моделей крайне важно, чтобы признаки были *нормированными*, то есть измерялись в одной шкале. Примером способа нормировки данных является вычитание среднего и деление на дисперсию каждого столбца в матрице «объекты-признаки».

[3] «Пытаются выбрать» — потому что функционал может оказать слишком сложным, не позволяющим точный поиск глобального минимума. В этом случае часто ограничиваются поиском локального экстремума.

- Бывает, что в выборку попадают *выбросы* — объекты, которые не являются корректными примерами из-за неправильно посчитанных признаков, ошибки сбора данных или чего-то еще. Их наличие может сильно испортить модель.

- Некоторые признаки могут оказаться *шумовыми*, то есть не имеющими никакого отношения к целевой переменной и к решаемой задаче. Примером, скорее всего, может служить признак «фаза луны в день первого экзамена» в задаче предсказания успешности прохождения сессии студентом.

Как показывает практика, простейшая предобработка данных может радикально улучшить качество итоговой модели.

Во время обучения модели очень важно следить за тем, чтобы не произошло *переобучения* (overfitting). Разберем это явление на примере. Допустим, что мы выбрали очень богатое семейство алгоритмов, состоящее из всех возможных функций: $\mathcal{A} = \{a : \mathbb{X} \to \mathbb{Y}\}$. В этом семействе всегда будет алгоритм, не допускающий ни одной ошибки на обучающей выборке, который просто запоминает ее:

$$a(x) = \begin{cases} y_i, & \text{если } x = x_i, \\ 0, & \text{если } x \in X^\ell. \end{cases}$$

Очевидно, что такой алгоритм нас не устраивает, поскольку для любого нового ресторана предскажет нулевую ошибку. Алгоритм оказался *переобученным* — он слишком сильно подогнался под обучающую выборку, не выявив никаких закономерностей в ней. Существует ряд методов, направленных на борьбу с переобучением, которые мы будем обсуждать на следующих занятиях. Впрочем, одну из идей мы можем обсудить уже сейчас. В нашем примере переобучение возникло из-за большой сложности семейства — алгоритмом могла оказаться любая функция. Очевидно, что если бы мы ограничили себя только линейными моделями, то итоговый алгоритм уже не смог бы запомнить всю выборку. Таким образом, можно бороться с переобучением путем *контроля сложности семейства алгоритмов* — чем меньше у нас данных для обучения, тем более простые семейства следует выбирать.

После того, как модель построена, нам нужно оценить, насколько хорошо она будет работать на новых данных. Для этого, например, можно в самом начала отложить часть обучающих объектов и не использовать их при построении модели. Тогда можно будет измерить качество готовой модели на этой *отложенной выборке*, получив тем самым оценку того, насколько она готова к работе на новых данных. Существуют и более сложный класс методов, называемый кросс-валидацией, о котором речь пойдет позже.

Итак, перечислим основные этапы решения задачи анализа данных:

1. Постановка задачи;

2. Выделение признаков;

3. Формирование выборки;

4. Выбор метрики качества;

5. Предобработка данных;

6. Построение модели;
7. Оценивание качества модели.

2 Напоминание основных фактов из математики

Далее в курсе нам понадобится ряд инструментов из других областей математики, о которых мы кратко напомним в данном разделе.

§2.1 Математический анализ

Производная и градиент. Одно из центральных понятий математического анализа — *производная*, которая определяется как предел отношения приращения функции к приращению аргумента в точке:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

Производная характеризует скорость роста функции в данной точке, а геометрически может быть найдена как угловой коэффициент касательной к графику функции в данной точке. Производная оказывается крайне полезной при поиске минимумов и максимумов функции. Согласно теореме Ферма, если x_0 — локальный экстремум функции $f(x)$, и она дифференцируема в этой точке, то производная в этой точке равна нулю:

$$f'(x_0) = 0.$$

Таким образом, если функция дифференцируема, то для поиска ее глобального минимума (при условии, что он существует) достаточно найти все корни уравнения $f'(x) = 0$, и выбрать из них тот, на котором функция достигает наименьшего значения.

Все это обобщается и на случай функций многих переменных — а алгоритмы являются таковыми, поскольку принимают на вход все признаки объекта. Рассмотрим функцию d переменных $f : \mathbb{R}^d \to \mathbb{R}$. Ее *частной производной* по переменной x^i в точке x_0 называется обычная производная по ней при фиксированных остальных переменных:

$$\frac{\partial}{\partial x_i} f(x_0) = \lim_{\Delta x^i \to 0} \frac{f(x_0^1, \ldots, x_0^{i-1}, x_0^i + \Delta x^i, x_0^{i+1}, \ldots, x_0^d) - f(x_0^1, \ldots, x_0^{i-1}, x_0^i, x_0^{i+1}, \ldots, x_0^d)}{\Delta x^i}.$$

Вектор частных производных функции называется *градиентом*:

$$\nabla f(x) = \left(\frac{\partial}{\partial x^1} f(x), \ldots, \frac{\partial}{\partial x^d} f(x) \right).$$

Можно показать, что вектор градиента в точке x указывает в сторону наискорейшего возрастания функции, а вектор антиградента (градиент с минусом) — в сторону наискорейшего убывания. Из-за этого свойства градиент широко используется в оптимизации: так, метод градиентного спуска основан на последовательном продвижении в пространстве в сторону антиградиента. Теорема Ферма тоже легко обобщается на случай функций многих переменных: необходимым условием экстремума в точке x_0 является равенство градиента нулю:

$$\nabla f(x_0) = 0.$$

Ряды. Можно ли вычислить значение произвольной функции, используя только операции сложения и умножения? Если она имеет k производных в точке x_0, то это можно сделать с помощью разложения в ряд Тейлора:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + o(|x - x_0|^k).$$

Аналогичным, но более громоздким способом можно разложить и функцию многих переменных. Разложение в ряд Тейлора дает некоторое представление о том, почему имеют смысл полиномиальные модели, которые вычисляют предсказание как некоторый полином от признаков: если степень полинома достаточно высока, то он, как и ряд Тейлора, сможет достаточно хорошо аппроксимировать неизвестную зависимость.

Ряд Тейлора делает разложение по степеням, но возможны и другие базисы. Так, в рядах Фурье разложение делается по тригонометрическим функциям с разной частотой:

$$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \sin nx + b_n \cos nx)\,.$$

Коэффициенты ряда Фурье часто оказываются хорошими признаками в задачах, связанных с анализом сигналов и временных рядов.

§2.2 Линейная алгебра

Матрица — это, по сути, таблица размера $m \times n$, заполненная числами. Матрицы естественным образом возникают во многих ситуациях. Так, линейную систему уравнений можно компактно записать, используя операцию умножения матрицы на вектор:

$$Ax = y.$$

При этом многие теоремы, связанные со свойствами и решениями таких систем, также легко формулируются в матричных терминах.

Матрица $A \in \mathbb{R}^{m \times n}$ задает линейную функцию, переводящую векторы из пространства $\mathbb{R}^n$ в пространство $\mathbb{R}^m$. Матричное умножение при этом определено так, чтобы произведение матриц A и B давало матрицу C, определяющую композицию соответствующих линейных функций. Если матрица квадратная, то она задает линейное преобразование, которое, как можно показать, всегда является композицией некоторого числа поворотов, отражений, проекций и масштабирований.

Одной из важных характеристик любой матрицы A является ее *ранг* $\operatorname{rk} A$, который определяется как число линейно независимых строк или столбцов в ней. Ранг является мерой количества информации, содержащейся в матрице. Так, если ранг матрицы размера $n \times n$ равен единице, то она может быть закодирована с помощью всего лишь $2n - 1$ чисел — достаточно сохранить первую ее строку, а для каждой из остальных строк запомнить коэффициент, домножением на который ее можно получить из первой строки.

Структура любой квадратной матрицы может быть описана с помощью ее *собственных векторов* и *собственных значений*. Вектор x является собственным для матрицы A, если он только лишь масштабируется под действием A:

$$Ax = \lambda x.$$

Число λ называется собственным значением.

Существует большое количество матричных разложений, которые направлены на представление матрицы в виде произведения других, более простых матриц. Например, LU-разложение заключается в представлении матрицы A в виде произведение верхнетреугольной и нижнетреугольной матриц. Благодаря этому решение линейной системы с произвольной матрицей можно заменить на решение двух систем с треугольными матрицами, что является существенно более простой задачей. Еще один пример — собственное разложение квадратной матрицы A:

$$A = Q\Lambda Q^{-1},$$

где столбцы матрицы Q являются собственными векторами матрицы A, а Λ — диагональная матрица с собственными значениями на диагонали.

В машинном обучении играет крайне важную роль *сингулярное разложение*(singular value decomposition, SVD), которое представляет произвольную матрицу $A \in \mathbb{R}^{m\times n}$ в виде произведения трех матриц:

$$A = UDV^T,$$

где $U \in \mathbb{R}^{m\times m}$ и $V \in \mathbb{R}^{n\times n}$ — ортогональные матрицы, $D \in \mathbb{R}^{m\times n}$ — диагональная матрица с неотрицательными элементами. При этом столбцы матриц U и V являются собственными векторами матриц AA^T и A^TA соответственно, а элементы на диагонали D —квадратные корни из собственных чисел этих двух матриц (наборы собственных чисел у этих матриц совпадают), которые называются сингулярными числами.

Оставим в матрицах U, D и V только те столбцы, которые соответствуют k наибольшим сингулярным числам, получив матрицы U_1, D_1 и V_1. Перемножив их, получим новую матрицу $B \in \mathbb{R}^{m\times n}$:

$$B = U_1D_1V_1^T,$$

которая будет иметь ранг k. Важность сингулярного разложения заключается в том, что матрица B по норме Фробениуса будет являться наиболее близкой к A среди всех матриц ранга k:

$$B = \underset{\mathrm{rk}\,B=k}{\arg\min} \sum_{i=1}^{m}\sum_{j=1}^{n} (B_{ij} - A_{ij})^2 .$$

Таким образом, сингулярное разложение позволяет оптимальным образом понизить ранг любой матрицы.

§2.3 Теория вероятностей

Теория вероятностей изучает *случайные величины*. Если говорить очень неформально, то случайная величина ξ — это величина, которая принимает одно из возможных значений в результате эксперимента. Случайная величина описывается своим распределением $P_\xi(x)$, которое описывает возможные значения и их вероятности. Две случайные величины ξ и η могут быть *независимыми* — это означает, что их значения никак не зависят друг от друга, или, более формально, что их совместное распределение распадается на произведение двух маргинальных распределений: $P_{\xi,\eta}(x,y) = P_\xi(x)P_\eta(y)$.

Основные виды распределений, которые рассматриваются в теории вероятностей — дискретные и непрерывные. Дискретное распределение задается m точками $x_1, \ldots, x_m$ и их вероятностями $p_1, \ldots, p_m$, которые суммируются в единицу. Простейший пример дискретного распределения — распределение Бернулли. Случайная величина с таким распределением принимает значения 1 и 0 с вероятностями p и $1-p$. Такое распределение, например, имеет результат подбрасывания идеальной монеты.

Сумма n независимых случайных величин с распределением Бернулли имеет биномиальное распределение, которое принимает целые значения от 0 до n с вероятностями

$$P(\xi = k) = C_n^k p^k (1-p)^{n-k}.$$

Данное распределение описывает число успехов в n независимых испытаниях и часто возникает, например, в A/B тестировании.

Другим примером дискретного распределения является распределение Пуассона, которое моделирует число событий, произошедших за фиксированный временной интервал при их известной средней частоте:

$$P(\xi = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, \ldots,$$

где $\lambda > 0$ — параметр, задающий среднюю частоту событий. Данное распределение может использоваться, например, для описания числа страниц, посещенных пользователем на сайте интернет-магазина, или для любой другой величины, имеющей смысл количества некоторых событий.

Непрерывная случайная величина задается *плотностью* $p(x)$, через которую определяется вероятность попадания величины в любую область A:

$$P(\xi \in A) = \int_A p(x)dx.$$

Примером является нормальное распределение с плотностью

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)^2.$$

Величины μ и σ являются параметрами распределения и задают его матожидание и дисперсию.

Другой пример — экспоненциальное распределение, которое моделирует время между последовательными событиями в пуассоновском процессе:

$$p(x) = \lambda e^{-\lambda x}, \quad x \geqslant 0,$$

где $\lambda \geqslant 0$ — параметр.

Матожидание $\mathbb{E}\xi$ случайной величины — это среднее значение, которое она примет в бесконечном числе экспериментов. Для дискретных случайных величин оно определяется через взвешенную сумму

$$\mathbb{E}\xi = \sum_{i=1}^{m} p_i x_i,$$

а для непрерывных — через интеграл

$$\mathbb{E}\xi = \int_{\mathbb{R}} xp(x)dx.$$

Дисперсия $\mathbb{D}\xi$ случайной величины описывает ее разброс и вычисляется как

$$\mathbb{D}\xi = \mathbb{E}(\xi - \mathbb{E}\xi)^2.$$

Для двух случайных величин ξ и η можно измерить коэффициент корреляции ρ, который определяет меру их линейной зависимости:

$$\rho(\xi, \eta) = \frac{\mathbb{E}(\xi - \mathbb{E}\xi)(\eta - \mathbb{E}\eta)}{\sqrt{\mathbb{D}\xi\mathbb{D}\eta}}.$$

Если эти случайные величины связаны соотношением $\xi = \eta + \text{const}$, то их коэффициент корреляции будет равен единице. Если они независимы, то коэффициент корреляции будет равен нулю.