Глава 8

Линейная алгебра

> Мы нередко слышим, что математика состоит в основном из "доказуемых теорем". Является ли работа писателя в основном "написанием предложений"?
>
> — Джан-Карло Рота (Gian-Carlo Rota)

Элемент данных вашего проекта в науке о данных включает в себя часть всей релевантной информации, которую вы можете найти, в одной или нескольких матрицах данных, в идеале как можно большего размера. Строки каждой матрицы представляют элементы или примеры, в то время как столбцы представляют различные элементы или атрибуты.

Линейная алгебра — это математика матриц: свойства расположений чисел и операции, которые можно с ними выполнять. Это делает ее языком науки о данных. Многие алгоритмы машинного обучения лучше всего понять через линейную алгебру. Алгоритмы для таких задач, как линейная регрессия, могут быть сведены к одной формуле за счет умножения правильной цепочки матричных произведений, чтобы получить желаемые результаты. Такие алгоритмы могут быть одновременно и простыми, и пугающе сложными, и тривиальными в реализации, и неординарными с точки зрения эффективности и надежности.

Возможно, вы когда-то проходили курс по линейной алгебре, но, вероятно, многое забыли. Здесь я рассмотрю б*о*льшую часть того, что вам нужно знать: основные операции над матрицами, чем они полезны, и как выработать интуицию о том, что они делают.

8.1. Сила линейной алгебры

Почему линейная алгебра столь сильна? Она регулирует работу матриц, а матрицы везде. Матричные представления важных объектов включают следующее.

- *Данные*. Наиболее полезное представление числовых наборов данных в виде матриц размером $n \times m$. Строки количеством n представляют объекты, элементы или экземпляры, в то время как каждый из m столбцов представляет отдельные элементы или измерения.

- *Наборы геометрических точек*. Матрица $n \times m$ может представлять собой облако точек в пространстве. Каждая из n строк представляет геометрическую точку, а столбцы m определяют размеры. Некоторые матричные операции имеют различные геометрические интерпретации, что позволяет нам обобщать двумерную геометрию, которую мы можем фактически визуализировать в многомерные пространства.
- *Системы уравнений*. линейное уравнение определяется суммой переменных, взвешенных по постоянным коэффициентам, например:

$$y = c_0 + c_1 x_1 + c_2 x_2 + \ldots c_{m-1} x_{m-1} .$$

  Система из n линейных уравнений может быть представлена в виде матрицы $n \times m$, где каждая строка представляет уравнение, и каждый из m столбцов связан с коэффициентами конкретной переменной (или постоянной "переменной" 1 в случае c_0). Зачастую для каждого уравнения необходимо представлять значение y. Обычно это делается с использованием отдельного массива $n \times 1$ или вектора значений решений.
- *Графики и сети*. Графы состоят из вершин и ребер, где ребра определены как упорядоченные пары вершин, например (i, j). Граф с n вершинами и m ребрами может быть представлен в виде матрицы M размером $n \times n$, где $M[i, j]$ обозначает количество (или вес) ребер от вершины i до вершины j. Существуют удивительные связи между комбинаторными свойствами и линейной алгеброй, такие как связь между путями в графах и матричным умножением, а также то, как кластеры вершин соотносятся с собственными значениями/векторами соответствующих матриц.
- *Перестановка операций*. Матрицы могут *делать* нечто. Тщательно разработанные матрицы могут выполнять такие геометрические операции над наборами точек, как перемещение, вращение и масштабирование. Умножение матрицы данных на соответствующую *матрицу перестановок* (permutation matrix) приведет к переупорядочению ее строк и столбцов. Движения могут быть определены *векторами* (vector), матрицами $n \times 1$, достаточно мощными для кодирования таких операций, как перевод и перестановка.

Повсеместное распространение матриц означает, что для манипулирования ими была разработана обширная инфраструктура инструментов. В частности, наличие высокопроизводительных библиотек линейной алгебры для вашего любимого языка программирования означает, что вам никогда не придется реализовать какой-либо базовый алгоритм самостоятельно. Лучшие реализации библиотек оптимизируют также такие вещи, как точность вычислений, ошибки кеширования и использование нескольких ядер, вплоть до уровня ассемблера. Наша цель — сформулировать задачу с использованием линейной алгебры, а алгоритмы оставить этим библиотекам.

8.1.1. Интерпретация линейных алгебраических формул

Краткие формулы, написанные как произведения матриц, могут позволять делать удивительные вещи, включая линейную регрессию, сжатие матриц и геометрические преобразования. Алгебраическая замена в сочетании с богатым набором тождеств дает элегантные механические способы манипулирования такими формулами.

Однако мне очень трудно интерпретировать такие последовательности операций способами, которые я действительно понимаю. Например, возьмем "алгоритм", лежащий в основе линейной регрессии наименьших квадратов:

$$c = (A^T A)^{-1} A^T b,$$

где система $n \times m$ — это $Ax = b$, а w — это вектор коэффициентов наилучшей аппроксимирующей линии.

Одна из причин, по которой я нахожу линейную алгебру сложной, — это *номенклатура* (nomenclature). Есть много разных терминов и концепций, которые нужно понять, чтобы действительно понять то, что происходит. Но еще большая проблема заключается в том, что большинство доказательств по вполне веской причине являются алгебраическими. На мой взгляд, алгебраические доказательства обычно не переносят интуиции о том, почему все работает так, как оно работает. Алгебраические доказательства легче проверить пошагово механическим способом, чем за счет понимания идей, лежащих в основе аргумента.

В этом тексте я представлю только одно формальное доказательство. По замыслу и теорема, и доказательство неверны.

Теорема 1. $2 = 1$.
Доказательство.

$$\begin{aligned} a &= b, \\ a^2 &= ab, \\ a^2 - b^2 &= ab - b^2, \\ (a+b)(a-b) &= b(a-b), \\ a + b &= b, \\ 2b &= b, \\ 2 &= 1. \end{aligned}$$

Если вы никогда раньше не видели такого доказательства, вы можете найти его убедительным, хотя, я надеюсь, вы понимаете на концептуальном уровне, что $2 \neq 1$. Каждая строка следует из предыдущей в ходе прямой алгебраической замены. Проблема, как выясняется, возникает при сокращении $(a - b)$, поскольку на самом деле мы делим на нуль.

Каковы уроки этого доказательства? Доказательства — это идеи, а не просто алгебраические манипуляции. Отсутствие идеи означает отсутствие доказательств. Чтобы понять линейную алгебру, ваша цель должна состоять в том, чтобы сначала проверить простейший интересный случай (обычно два измерения), чтобы выработать интуицию, а затем попытаться представить, как она может обобщаться для более высоких измерений. Всегда есть особые случаи, за которыми нужно следить, например деление на нуль. В линейной алгебре эти случаи включают несоответствия размеров и сингулярные (т.е. необратимые) матрицы. Теория линейной алгебры работает, за исключением случаев, когда она не работает, и лучше думать с точки зрения общих случаев, а не патологических.

8.1.2. Геометрия и векторы

Существует удобная интерпретация "векторов", означающих матрицы $1 \times d$, как *векторов* (vectors) в геометрическом смысле, представляющие собой направленные лучи от начала координат через данную точку в d измерениях.

Нормализация каждого такого вектора v на единицу длины (за счет деления каждой координаты на расстояние от v до начала координат) помещает его в d-мерную сферу, как показано на рис. 8.1: круг для точек на плоскости, реальная сфера для $d = 3$ и некоторая невизуализируемая гиперсфера для $d \geq 4$.

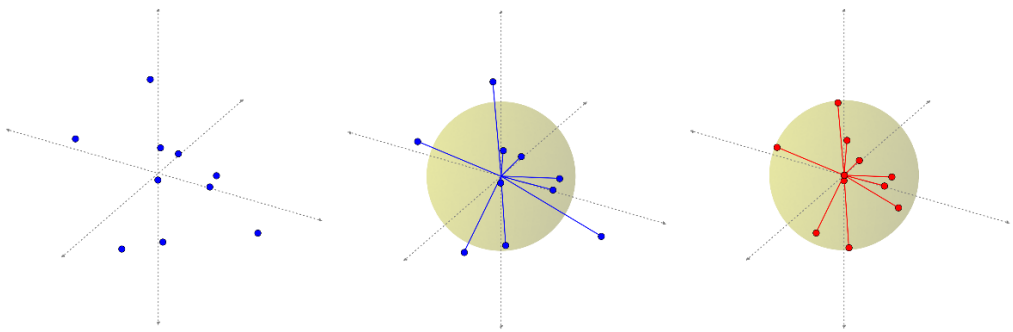

Рис. 8.1. Точки могут быть сведены до векторов на единичной сфере плюс их величины

Эта нормализация оказывается полезной вещью. Расстояния между точками становятся углами между векторами в целях сравнения. Две близлежащие точки будут иметь небольшой угол между ними относительно начала координат: малые расстояния означают небольшие углы. Игнорирование величин является формой масштабирования, делающей все точки непосредственно сопоставимыми.

Скалярное произведение (dot product) является весьма полезной операцией, сводящей векторы к скалярным величинам. Скалярное произведение двух векторов A и B длиной n определяется следующим образом:

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

Мы можем использовать операцию скалярного произведения для вычисления угла $\theta = \angle AOB$ между векторами A и B, где O — это начало координат:

$$\cos(\Theta) = \frac{A \cdot B}{||A||\,||B||}$$

Попробуем разобрать эту формулу. Символ $||V||$ означает "длина вектора V". Для единичных векторов она по определению равна 1. В общем, это величина, на которую мы должны разделить V, чтобы сделать его единичным вектором.

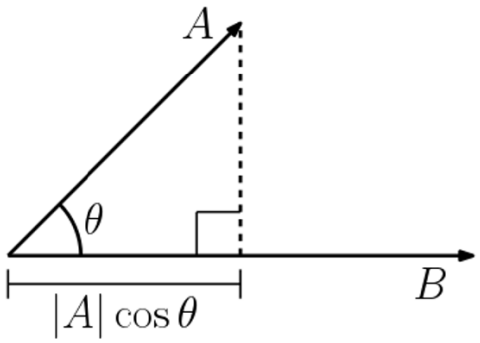

Рис. 8.2. Скалярное произведение двух векторов определяет косинус угла между ними

Но какова связь между скалярным произведением и углом? Рассмотрим простейший случай угла, определенного между двумя лучами: A, направленного вдоль нулевого градуса, и $B = (x, y)$. Таким образом, единичный луч равен $A = (1, 0)$. В этом случае скалярное произведение равно $1 \cdot x + 0 \cdot y = x$, что и должно быть $\cos(\theta)$, если B — это единичный вектор. Мы можем принять на веру, что это обобщение для B подходит и для более высоких измерений.

Таким образом, меньший угол означает более близкие точки на сфере. Но есть другая связь между вещами, которые мы знаем. Напомним особые случаи функции косинуса, приведенные здесь в радианах:

$$\cos(0) = 1,\ \cos(\pi/2) = 0,\ \cos(\pi) = -1.$$

Значения функции косинуса находятся в диапазоне [–1, 1], точно в том же диапазоне, что и коэффициент корреляции. Кроме того, интерпретация

одинакова: два одинаковых вектора идеально коррелируют, а антиподальные точки коррелируют отрицательно. Ортогональные точки/векторы (случай $\Theta = \pi/2$) имеют меньше всего общего друг с другом.

Функция косинуса — это в точности корреляция двух переменных с нулевым средним. Для единичных векторов $\|A\| = \|B\| = 1$, поэтому угол между A и B полностью определяется скалярным произведением.

На заметку. Скалярное произведение двух векторов измеряет сходство точно так же, как коэффициент корреляции Пирсона.

8.2. Визуализация матричных операций

Я предполагаю, что вы уже имели опыт работы с основными матричными операциями, такими как транспонирование, умножение и инверсия. Этот раздел предназначен для напоминания, а не начального обучения.

Рис. 8.3. Пример матричного изображения: Линкольн (слева) и его мемориал (справа). Центральное изображение представляет собой линейную комбинацию того, что слева, и того, что справа, для $\alpha = 0{,}5$

Но чтобы выработать лучшую интуицию, я буду представлять матрицы в виде изображений, а не чисел, чтобы мы могли видеть происходящее, когда оперируем ими. На рис. 8.3 показаны наши основные матричные изображения: президент Авраам Линкольн (слева) и здание его мемориала (справа). Первое представляет собой человеческое лицо, а второе содержит ярко выраженные ряды и колонки.

Имейте в виду, что мы будем постепенно масштабировать матрицу между каждыми операциями, поэтому абсолютный цвет не имеет значения. Интересные шаблоны кроются в различиях между светлым и темным, т.е. самыми маленькими и самыми большими числами в текущей матрице. Обратите также внимание, что исходный элемент матрицы M [1, 1] представляет левый верхний угол изображения.

8.2.1. Сложение матриц

Сложение матриц (matrix addition) является простой операцией: для матриц A и B размером $n \times m$ каждая означает, что для $C = A + B$:

$$C_{ij} = A_{ij} + B_{ij}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

Скалярное умножение (scalar multiplication) позволяет изменить вес каждого элемента в матрице одновременно, возможно, чтобы нормализовать их. Для любой матрицы A и числа c, $A' = c \cdot A$ означает, что

$$A'_{ij} = cA_{ij}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

Сочетание сложения матриц со скалярным умножением позволяет выполнять *линейные комбинации* (linear combination) матриц. Формула $\alpha \cdot A + (1 - \alpha) \cdot B$ имеет плавное затухание между A (для $\alpha = 1$) и B (для $\alpha = 0$), как показано на рис. 8.3. Это и обеспечивает способ преобразования изображений из A в B.

Транспонирование (transpose) матрицы M меняет местами строки со столбцами, превращая матрицу $a \times b$ в матрицу $b \times a$, где M^T — это

$$M^T_{ij} = M_{ji}, \text{ для всех } 1 \le i \le n \text{ и } 1 \le j \le m.$$

В результате транспонирования квадратной матрицы получается квадратная матрица, поэтому M и M^T можно безопасно складывать и умножать. В более общем смысле: транспонирование — это операция, которая используется для изменения ориентации матрицы, чтобы ее *можно* было складывать или умножать на другую.

Транспонирование матрицы как бы "поворачивает" ее на 180 градусов[1], поэтому $(A^T)^T = A$. В случае квадратных матриц добавление матрицы к ее транспозиции (т.е. к ее транспонированной версии) симметрично, как показано на рис. 8.4 (справа). Причина ясна: $C = A + A^T$ подразумевает, что

$$C_{ij} = A_{ij} + A_{ji} = C_{ji}.$$

Рис. 8.4. Линкольн (слева) и его транспозиция (в центре). Сумма матрицы и ее транспозиции симметрична вдоль главной диагонали (справа)

[1] В данном случае автор говорит образно. На самом деле в результате транспонирования матрицы она зеркально поворачивается на 90 градусов против часовой стрелки, поскольку строки становятся столбцами, а столбцы — строками с теми же номерами.

8.2.2. Умножение матриц

Матричное умножение представляет собой совокупную версию *скалярного произведения* (dot product), или *внутреннего произведения* (inner product), векторов. Напомним, что для двух n-элементных векторов, X и Y, скалярное произведение $X \cdot Y$ определено как:

$$X \cdot Y = \sum_{i=1}^{n} X_i Y_i.$$

Скалярные произведения измеряют, насколько "синхронизированы" эти два вектора. Мы уже видели скалярное произведение при вычислении косинусного расстояния и коэффициента корреляции. Это операция, которая сводит пару векторов к одному числу.

Матричное произведение XY^T этих двух векторов создает матрицу 1×1, содержащую скалярное произведение $X \cdot Y$. Для матриц общего вида произведение $C = AB$ определяется как:

$$C_{ij} = \sum_{i=1}^{k} A_{ik} B_{kj}.$$

Чтобы это работало, A и B должны иметь одинаковые внутренние размеры, подразумевая, что если A равно $n \times k$, то B должно иметь размеры $k \times m$. Каждый элемент произведения матрицы C размером $n \times m$ является скалярным произведением i-й строки A с j-м столбцом B.

Наиболее важные свойства умножения матриц таковы.

- *Они не коммутативны*. *Коммутативность* (commutativity) — это напоминание о том, что порядок не имеет значения $xy = yx$. Хотя мы принимаем коммутативность как должное при умножении целых чисел, порядок *имеет* значение при умножении матриц. Для любой пары неквадратных матриц A и B, как минимум один из AB или BA имеет совместимые размеры. Но умножение даже квадратной матрицы не коммутативно, как показывают скаляры ниже:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

  и ковариационные матрицы на рис. 8.5.

- *Матричное умножение ассоциативно*. *Ассоциативность* (associativity) дает нам право применять скобки по своему усмотрению, выполняя операции в относительном порядке, который мы выбираем. При вычислении произведения ABC у нас есть выбор из двух вариантов: $(AB)C$ или $A(BC)$.

Более длинные цепочки матриц дают еще большую свободу, причем количество возможных скобок растет экспоненциально при увеличении длины цепочки. Все они будут возвращать один и тот же ответ, как показано здесь:

$$\left(\begin{bmatrix}1 & 2\\ 4 & 4\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\right)\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix}=\begin{bmatrix}1 & 4\\ 3 & 8\end{bmatrix}\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix}=\begin{bmatrix}7 & 2\\ 17 & 6\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\left(\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}3 & 2\\ 1 & 0\end{bmatrix}\right)=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\begin{bmatrix}3 & 2\\ 2 & 0\end{bmatrix}=\begin{bmatrix}7 & 2\\ 17 & 6\end{bmatrix}$$

Есть две основные причины, почему ассоциативность имеет для нас значение. В алгебраическом смысле это позволяет нам идентифицировать соседние пары матриц в цепочке и заменять их в соответствии с тождеством, если оно у нас есть. Но еще одной проблемой являются вычисления. Размер промежуточных матричных произведений легко может резко возрасти в середине. Предположим, мы стремимся вычислить $ABCD$, где A равно $1 \times n$, B и C равно $n \times n$, а D равно $n \times 1$. Произведение $(AB)(CD)$ состоит всего из $2n^2 + n$ операций, при условии обычного алгоритма умножения матриц со вложенными циклами. Напротив, $(A(BC))D$ потребует уже $n^3 + n^2 + n$ операций.

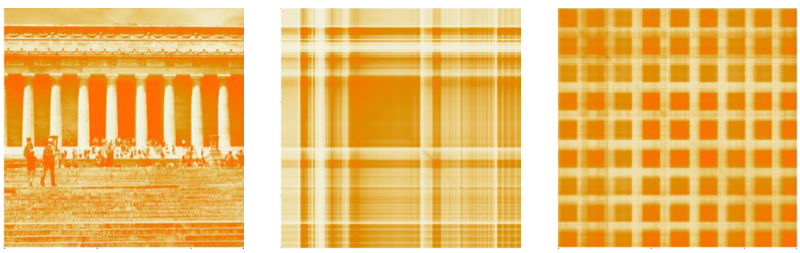

Рис. 8.5. Мемориал Линкольна M (слева) и его ковариационные матрицы. Большой блок в середине MM^T (в центре) является результатом сходства всех рядов от средней полосы M. Рисунок плотной сетки M^TM (справа) отражает регулярный шаблон колонн на здании мемориала

Алгоритм умножения матриц с вложенными циклами, которому вас учили в старших классах школы, тривиально прост в программировании, и он действительно рассматривается далее. Но не спешите программировать его. Гораздо более быстрые и численно устойчивые алгоритмы есть в высоко оптимизированных библиотеках линейной алгебры вашего любимого языка программирования. Формулировка алгоритмов в виде матричных произведений на основе больших массивов чисел вместо использования специальной логики

является нелогичным для большинства кибернетиков. Но эта стратегия может дать очень большие выигрыши на практике.

8.2.3. Применение матричного умножения

На первый взгляд, умножение матриц — довольно неуклюжая операция. Когда я впервые столкнулся с линейной алгеброй, я не мог понять, почему мы не могли просто умножить числа попарно, как сложение матрицы, и покончить с этим.

Причина, по которой мы занимаемся умножением матриц, заключается в том, что с ним можно многое сделать. Здесь мы рассмотрим эти случаи.

Ковариационные матрицы

Умножение матрицы A на ее транспозицию A^T является очень распространенной операцией. Почему? С одной стороны, мы *можем* их умножить: если A — это матрица $n \times d$, то A^T — это матрица $d \times n$. Таким образом, они всегда совместимы для умножения AA^T. Они одинаково совместимы для умножения и другим способом, т.е. A^TA.

Оба эти произведения имеют важные интерпретации. Предположим, что A — это матрица объектов $n \times d$, состоящая из n строк, представляющих элементы или точки, и d столбцов, представляющих наблюдаемые признаки этих элементов.

- $C = AA^T$ — это матрица размером $n \times n$ скалярных произведений, измеряющая "синхронность" среди точек. В частности, C_{ij} является мерой того, насколько элемент i похож на элемент j.
- $D = A^TA$ — это матрица размером $d \times d$ скалярных произведений, измеряющая "синхронность" среди столбцов или элементов. Здесь D_{ij} представляет сходство между признаком i и признаком j.

Эти матрицы достаточно часто используются, чтобы заработать собственное имя — *ковариационные матрицы* (covariance matrices). Этот термин нередко встречается в разговорах между специалистами по данным, так что вы должны знать о нем. Формула ковариации, которую мы дали при вычислении коэффициента корреляции, была такова

$$Cov(X,Y) = \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}).$$

Таким образом, наши две матрицы, строго говоря, являются ковариационными матрицами только в том случае, если строки или столбцы A имеют нулевое среднее значение. Но независимо от этого, величины матричного про-

изведения отражают степень, в которой значения конкретных пар строк или столбцов совпадают.

На рис. 8.5 представлены ковариационные матрицы мемориала Линкольна. Более темные пятна определяют строки и столбцы на изображении с наибольшим сходством. Постарайтесь понять, откуда берутся видимые структуры в этих ковариационных матрицах.

На рис. 8.5 (в центре) представлена ковариационная матрица строк MM^T. Большой темный прямоугольник в середине представляет большие скалярные произведения, полученные из любых двух рядов, проходящих через все белые колонны мемориала. Эти полосы света и тьмы жестко коррелируют, а интенсивно темные области способствуют образованию больших скалярных произведений. Светлые ряды, соответствующие небу, фронтону и лестнице, одинаково коррелированы и когерентны, но не имеют темных областей, чтобы сделать их скалярные произведения достаточно большими.

Правое изображение представляет ковариационную матрицу столбцов M^TM. Все пары столбцов матрицы сильно коррелируют друг с другом, как положительно, так и отрицательно, но столбцы матрицы через белые колонны здания имеют малый вес, а следовательно, небольшое скалярное произведение. Вместе они формируют шахматную доску из чередующихся темных и светлых полос.

Матричное умножение и пути

Квадратные матрицы могут быть умножены сами на себя без транспонирования. Действительно, $A^2 = A \times A$ называется *квадратом* матрицы A. В более общем смысле A^k называют k-й степенью матрицы.

Степени матрицы A имеют вполне естественную интерпретацию, когда A представляет *матрицу смежности* (adjacency matrix) графа или сети. Матрица смежности $A[i, j] = 1$, когда (i, j) является ребром в сети. В противном случае, когда i и j не являются прямыми соседями, $A[i, j] = 0$.

Для таких матриц 0/1 произведение A^2 дает число путей длиной два в A. В частности:

$$A^2[i, j] = \sum_{k=1}^{n} A[i, k]\, A[k, j].$$

Существует только один путь длиной два от i до j для каждой промежуточной вершины k, так как (i, k) и (k, j) оба являются ребрами в графе. Сумма этих путей рассчитывается по скалярному произведению выше.

Но вычислительные возможности матриц имеют смысл даже для более общих матриц. Они моделируют эффекты диффузии, распределяя вес каждого элемента среди связанных элементов. Такие вещи происходят в знаменитом алгоритме Google PageRank и других итерационных процессах, таких как распространение инфекции.

Умножение матриц и перестановки

Матричное умножение зачастую используется только для того, чтобы изменить порядок элементов в конкретной матрице. Напомним, что высокопроизводительные процедуры умножения матриц являются ослепительно быстрыми, достаточно того, что зачастую они способны выполнять такие операции быстрее, чем логика специального программирования. Они также обеспечивают способ описания таких операций в терминах алгебраических формул, сохраняя тем самым компактность и удобочитаемость.

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad M = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix} \quad PM = \begin{pmatrix} 31 & 32 & 33 & 34 \\ 11 & 12 & 13 & 14 \\ 41 & 42 & 43 & 44 \\ 21 & 22 & 23 & 24 \end{pmatrix}$$

Рис. 8.6. Умножение матрицы на матрицу перестановок переставляет ее строки и столбцы

Самая известная матрица перестановок вообще ничего не делает. *Единичная матрица* (identity matrix) — это матрица $n \times n$, состоящая из всех нулей, кроме тех, которые расположены вдоль главной диагонали. Для $n = 4$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Убедитесь, что $AI = IA = A$, т.е. что умножение на единичную матрицу коммутативно.

Обратите внимание: в каждой строке и столбце I содержится ровно один ненулевой элемент. Матрицы с этим свойством называются *матрицами перестановок* (permutation matrix), поскольку ненулевой элемент в позиции (i, j) можно интерпретировать как то, что элемент i находится в позиции j перестановки. Например, перестановка (2, 4, 3, 1) определяет следующую матрицу перестановок:

$$P_{(2431)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Обратите внимание, что единичная матрица соответствует перестановке (1, 2, …, n).

Ключевым моментом здесь является то, что мы можем умножить A на соответствующую матрицу перестановок, чтобы переставить строки и столбцы, как нам угодно. На рис. 8.7 показано, что происходит, когда мы умножаем наше изображение на "обратную" матрицу перестановок r, где единицы лежат вдоль малой диагонали. Поскольку матричное умножение обычно не коммутативно, мы получаем разные результаты для Ar и rA. Убедитесь сами почему.

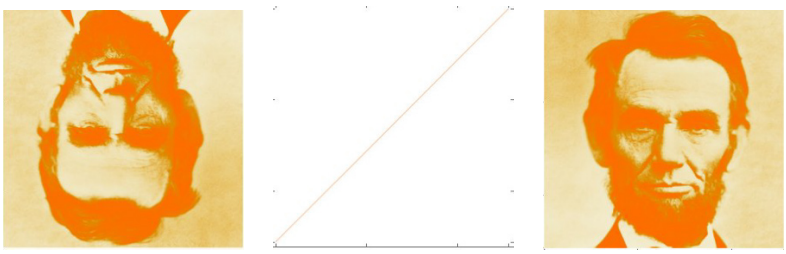

Рис. 8.7. Умножение матрицы Линкольна M на матрицу обратной перестановки r (в центре). Произведение rM переворачивает Линкольна вверх ногами (слева), в то время как Mr переносит пробор волос на другую сторону головы (справа)

Вращение точек в пространстве

Умножение чего-либо на правильную матрицу может иметь магические свойства. Мы видели, как набор из n точек на плоскости (т.е. в двух измерениях) может быть представлен $(n \times 2)$-мерной матрицей S. Умножение таких точек на правильную матрицу может привести к естественным геометрическим преобразованиям.

Матрица вращения (rotation matrix) R_θ осуществляет преобразование точек, поворачивая их вокруг начала координат на угол θ. В двух измерениях R_θ определяется как

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

В частности, после соответствующего умножения/поворота точка (x, y) переходит в

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix}$$

Для $\theta = 180° = \pi$ радиан, $\cos(\theta) = -1$ а $\sin(\theta) = 0$, так что это сводится к $(-x, -y)$, делая правильную вещь и помещая точку в противоположный квадрант.

Для нашей $(n \times 2)$-мерной скалярной матрицы S мы можем использовать функцию транспонирования для правильной ориентации матрицы. Попытайтесь доказать, что

$$S' = (R_\theta S^T)^T$$

делает именно то, что нам нужно.

Естественные обобщения R_θ существуют для поворота точек в произвольных измерениях. Кроме того, произвольные последовательности преобразований могут быть реализованы в ходе умножения цепочек матриц вращения, расширения и отражения, что дает компактное описание сложных манипуляций.

8.2.4. Единичные матрицы и инверсия

Единичные операции играют большую роль в алгебраических структурах. Для числового сложения единичным элементом является нуль, так $0 + x = x + 0 = x$. Такую же роль играет единица при умножении, так $1 \cdot x = x \cdot 1 = x$.

Единичный элемент при умножении матриц — это единичная матрица со всеми единицами по главной диагонали. Умножение на единичную матрицу коммутативно, поэтому $IA = AI = A$.

Операция *инверсии* (inverse) сводится к приведению элемента x к его единичному элементу. Для численного сложения обратная величина x равна $(-x)$, поскольку $x + (-x) = 0$. Инверсной операцией умножения является *деление*. Мы можем инвертировать число, умножив его на обратное, таким образом $x(1/x) = 1$.

Люди обычно не говорят о делении матриц. Однако они очень часто инвертируют их. Мы говорим, что A^{-1} — это *мультипликативная инверсия* (multiplicative inverse) матрицы A, если $AA^{-1} = I$, где I — это единичная матрица. Инверсия является важным частным случаем деления, поскольку $AA^{-1} = I$ подразумевает $A^{-1} = I/A$. Это на самом деле эквивалентные операции, поскольку $A/B = AB^{-1}$.

На рис. 8.8 (слева) демонстрируется инверсия изображения Линкольна, которая очень похожа на случайный шум. Но ее умножение на изображение дает тонкую главную диагональ единичной матрицы, хотя и наложенную на фон числовой ошибки. Вычисления с плавающей точкой по своей сути неточны, а такие алгоритмы, как инверсия, которые подразумевают многократные сложения и умножения, нередко накапливают ошибки в процессе.

Как мы можем вычислить обратную матрицу? Существует замкнутая форма для нахождения инверсии A^{-1} матрицы A размером 2×2, а именно:

$$A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

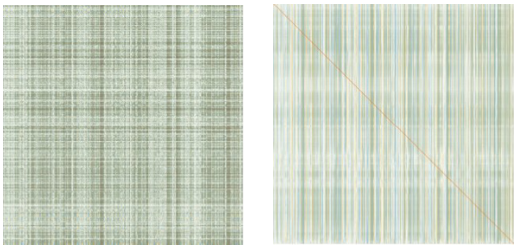

Рис. 8.8. Инверсия Линкольна не очень похожа на человека (слева), но MM^{-1} создает единичную матрицу по модулю небольших ненулевых членов из-за проблем с числовой точностью

В более общем смысле, существует подход к инвертированию матриц в ходе решения линейной системы с использованием исключения по Гауссу.

Заметим, что эта замкнутая форма для инверсии делится на нуль всякий раз, когда произведения диагоналей равны, т.е. $ad = bc$. Это говорит нам о том, что такие матрицы не являются обратимыми, или *сингулярными* (singular), что означает отсутствие инверсии. Так же, как мы не можем делить числа на нуль, мы не можем инвертировать сингулярные матрицы.

Матрицы, которые мы можем инвертировать, называются *несингулярными* (non-singular), и жизнь становится лучше, когда наши матрицы обладают этим свойством. Проверка матрицы на сингулярность заключается в выяснении, является ли ее *определитель* (determinant) ненулевым. Для матриц 2×2 определителем является разность между произведением ее диагоналей, точнее, знаменатель в формуле инверсии.

Кроме того, определитель определен только для квадратных матриц, поэтому только квадратные матрицы инвертируемы. Стоимость вычисления этого детерминанта составляет $O(n^3)$, поэтому он дорого обходится в больших матрицах, и действительно настолько же дорог, как попытка инвертировать саму матрицу с использованием исключения по Гауссу.

8.2.5. Инверсия матриц и линейные системы

Линейные уравнения определяются суммой переменных, взвешенных по постоянным коэффициентам:

$$y = c_0 + c_1 x_1 + c_2 x_2 + \ldots c_{m-1} x_{m-1}.$$

Таким образом, коэффициенты, определяющие систему из n линейных уравнений, могут быть представлены в виде матрицы C размером $n \times m$. Здесь каждая строка представляет уравнение, а каждый из m столбцов — коэффициенты отдельной переменной.

Мы можем аккуратно оценить все n этих уравнений на конкретном входном векторе X размером $m \times 1$, умножив CX. Результатом будет вектор $n \times 1$, сообщающий значение $f_i(X)$ для каждого из n линейных уравнений, $1 \le i \le n$. Частным случаем здесь является аддитивный член c_0. Для правильной интерпретации соответствующий столбец в векторе X должен содержать все единицы.

Если мы обобщим X как матрицу $m \times p$, содержащую p различных точек, наше произведение Cx приведет к матрице $n \times p$, оценивающей каждую точку по каждому уравнению в умножении одной матрицы.

Но основной операцией в системах из n уравнений является их решение, т.е. определение вектора X, необходимого для получения целевого значения Y для каждого уравнения. Предположим, дан вектор $n \times 1$ значений решений Y и матрица коэффициентов C, мы ищем X, так что $CX = Y$. Инверсия матриц может использоваться для решения линейных систем. Умножение обеих сторон $CX = Y$ на инверсное значение C дает:

$$(C^{-1}C)X = C^{-1}Y \rightarrow X = C^{-1}Y.$$

Таким образом, система уравнений может быть решена за счет инверсии C с последующим умножением C^{-1} на Y.

Исключение по Гауссу (Gaussian elimination) — это еще один подход к решению линейных систем, который, я надеюсь, вы видели раньше. Напомним, что он решает уравнения, выполняя операции сложения/вычитания строк, чтобы упростить матрицу C уравнений до тех пор, пока она не сведется к единичной матрице. Это упрощает считывание значений переменных, поскольку каждое уравнение приведено к виду $X_i = Y'_i$, где Y' — это результат применения этих же операций со строками к исходному целевому вектору Y.

$$[A \mid I] = \left[\begin{array}{ccc|ccc} 6 & 4 & 1 & 1 & 0 & 0 \\ 10 & 7 & 2 & 0 & 1 & 0 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] =$$

$$= \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 5 & 3 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 0 & 0 & 1 & -5 & 2 & 2 \end{array}\right]$$

$$\rightarrow A^{-1} = \left[\begin{array}{ccc} 1 & -1 & 1 \\ 0 & 1 & -2 \\ -5 & 2 & 2 \end{array}\right]$$

Рис. 8.9. Инверсия матриц может быть вычислена с использованием исключения по Гауссу

Вычисление инверсной матрицы может быть выполнено таким же образом, как показано на рис. 8.9. Мы выполняем операции со строками, чтобы упростить матрицу коэффициентов до единичной матрицы *I*, чтобы создать инверсную. Я рассматриваю это как алгоритм Дориана Грея: матрица коэффициентов *C* украшает единичную матрицу, а цель *I* стареет в обратном направлении.[2]

Поэтому мы можем использовать инверсию матриц для решения линейных систем и решатели линейных систем — для инвертирования матриц. Таким образом, две проблемы в некотором смысле эквивалентны. Вычисление инверсии удешевляет оценку нескольких векторов *Y* для данной системы *C*, сводя ее к умножению на одну матрицу. Но это может быть сделано еще более эффективно с использованием *разложения LU матрицы* (LU-decomposition), обсуждаемого в разделе 8.3.2. Исключение по Гауссу оказывается более численно устойчивым, чем инверсия и, как правило, является предпочтительным методом при решении линейных систем.

8.2.6. Ранг матриц

Система уравнений правильно *определена* (determined), когда существует *n* линейно независимых уравнений и *n* неизвестных. Например, линейная система

$$2x_1 + 1x_2 = 5,$$
$$3x_1 - 2x_2 = 4.$$

определена правильно. Единственным решением является точка ($x_1 = 2, x_2 = 1$).

Напротив, системы уравнений *недоопределены* (underdetermined), если есть строки (уравнения), которые могут быть выражены как линейные комбинации других строк. Линейная система

$$2x_1 + 1x_2 = 5,$$
$$4x_1 + 2x_2 = 10.$$

недоопределена, поскольку второй ряд в два раза больше первого. Должно быть понятно, что для решения неопределенной системы линейных уравнений недостаточно информации.

Ранг (rank) матрицы измеряет количество линейно независимых строк. Матрица $n \times n$ должна иметь ранг n для всех операций, которые должны быть на ней правильно определены.

Ранг матрицы можно вычислить, применив метод исключения по Гауссу. Если она недоопределена, то некоторые переменные исчезнут в ходе операций сокращения строк. Существует также связь между недоопределенными

[2] В романе Оскара Уайльда “Портрет Дориана Грея” главный герой остается красивым, в то время как его портрет со временем стареет.

системами и сингулярными матрицами: напомним, что они были идентифицированы по нулевому детерминанту. Вот почему разница в перекрестном произведении здесь $(2 \cdot 2 - 4 \cdot 1)$ равна нулю.

Матрицы характеристик зачастую имеют более низкий ранг, чем мы могли бы пожелать. Файлы примеров, как правило, содержат повторяющиеся записи, что приводит к идентичности двух строк матрицы. Также вполне возможно, что несколько столбцов эквивалентны: представьте, что каждая запись содержит рост, измеренный в футах и метрах, например.

Такие вещи, безусловно, случаются, и плохо, когда это происходит. На нашем изображении мемориала Линкольна некоторые алгоритмы потерпели неудачу численно. Оказалось, что наше изображение 512×512 имело ранг только 508, поэтому не все строки были линейно независимыми. Чтобы сделать его полноценной матрицей, вы можете добавить небольшое количество случайного шума к каждому элементу, что повысит ранг без серьезных искажений изображения. Эта нелепость может позволить вашим данным проходить через алгоритм без предупреждающего сообщения, но это указывает на количество проблем, которые могут возникнуть.

Линейные системы могут быть "почти" низкого ранга, что приводит к большей опасности потери точности из-за числовых проблем. Это формально фиксируется матричным инвариантом, *коэффициентом перекоса* (condition number), который в случае линейной системы измеряет, насколько чувствительно значение X к небольшим изменениям Y в $Y = AX$.

Помните о капризах числовых расчетов при оценке ваших результатов. Например, рекомендуется вычислять AX для любого предполагаемого решения X и видеть, насколько AX *действительно* хорош по сравнению с Y. Теоретически разница будет равна нулю, но на практике вы можете быть удивлены тем, насколько грубыми являются вычисления.

8.3. Разложение матриц

Разложение (factoring) матрицы A в матрицы B и C представляет собой особый аспект деления. Мы видели, что любая несингулярная матрица M имеет инверсию M^{-1}, поэтому единичная матрица I может быть разложена как $I = MM^{-1}$. Это доказывает, что некоторые матрицы (например, I) могут быть разложены, а кроме того, они могут иметь много различных разложений. В этом случае каждая возможная несингулярная M определяет разные разложения.

Разложение матриц является важной абстракцией в науке о данных, приводящей к кратким представлениям функций и идеям, таким как тематическое моделирование. Она играет важную роль в решении линейных систем за счет специальных разложений, таких как разложение LU матрицы.

К сожалению, найти такие разложения проблематично. Разложение *целых чисел* — это сложная проблема, хотя эта сложность исчезает, когда вам доступны числа с плавающей запятой. Разложение матриц оказывается сложнее: для конкретной матрицы точное разложение может быть невозможно, особенно если мы ищем разложение $M = XY$, где X и Y имеют заданные размеры.

8.3.1. Разложение матрицы признаков

Многие важные алгоритмы машинного обучения можно рассматривать с точки зрения разложения матрицы. Предположим, нам дана матрица признаков A размером $n \times m$, где в соответствии с обычным соглашением строки представляют элементы/примеры, а столбцы представляют признаки примеров.

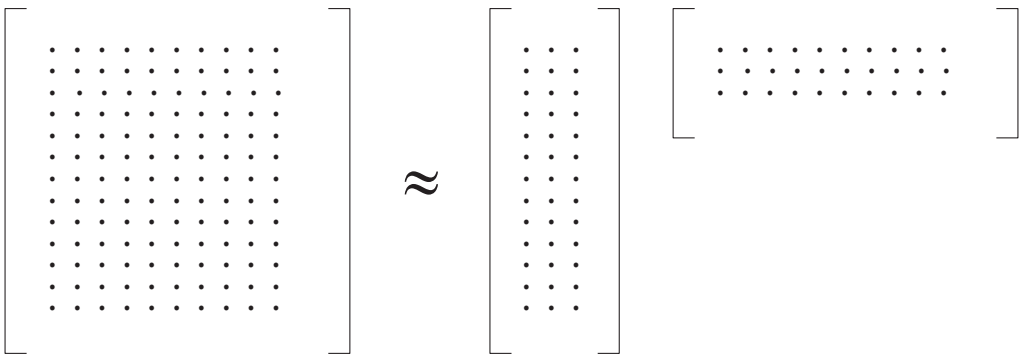

Рис. 8.10. Разложение матрицы признаков $A \approx BC$ дает B как более краткое представление элементов и C как более сжатое представление признаков A^T

Теперь предположим, что мы можем *разложить* (factor) матрицу A, т.е. выразить ее как произведение $A \approx BC$, где B — это матрица $n \times k$, а C — матрица $k \times m$. С учетом, что $k < \min(n, m)$, как показано на рис. 8.10, это хорошо по нескольким причинам.

- *Совместно B и C обеспечивают сжатое представление матрицы A*. Матрицы признаков, как правило, большие и неудобные для работы. Разложение обеспечивает способ кодирования всей информации большой матрицы в две меньшие матрицы, которые вместе будут меньше исходной.
- *B служит меньшей матрицей признаков для элементов, заменяя A*. Разложение матрицы B имеет n строк, точно так же, как исходная матрица A. Но у нее существенно меньше столбцов, поскольку $k < m$. Это означает, что "большая часть" информации в A теперь закодирована в B. Меньшее количество столбцов означает меньшую матрицу и меньшее количество параметров, подходящих для любой модели, построенной с использованием этих новых признаков. Эти более абстрактные признаки могут

также представлять интерес для других приложений, поскольку являются краткими описаниями строк набора данных.

- *C^T служит меньшей матрицей признаков, заменяя A^T.* Транспонирование матрицы признаков превращает столбцы/признаки в строки/элементы. Разложенная матрица C^T имеет m строк и k столбцов признаков, представляющих их. Во многих случаях m оригинальных "признаков" стоит моделировать сами по себе.

Рассмотрим репрезентативный пример из анализа текста. Возможно, мы хотим представить n документов, каждый твит или другое сообщение в социальной сети с точки зрения используемого им словаря. Каждый из m наших признаков будет соответствовать отдельному словарному слову, и $A[i, j]$ будет записывать, как часто словарное слово w_j (скажем, *cat* (кот)) появлялось в сообщении номер i. Рабочий словарный запас в английском языке довольно большой с длинным хвостом, поэтому, вероятно, мы можем ограничить его m = 50 000 наиболее часто используемых слов. Большинство сообщений будут короткими, не более нескольких сотен слов. Таким образом, наша матрица признаков A будет очень разреженной, изобилующей огромным количеством нулей.

Теперь предположим, что мы можем разложить $A = BC$, где внутреннее измерение k относительно мало. Скажем, k = 100. Теперь каждое сообщение будет представлено строкой B, содержащей только сто чисел вместо полных 50 000. Это значительно упрощает сравнение текстов на предмет сходства. Эти k измерений можно рассматривать как аналог "тем" в документах, поэтому все сообщения о спорте должны использовать другой набор тем, нежели те, что касаются отношений.

Теперь матрица C^T может рассматриваться как содержащая вектор признаков для каждого из словарных слов. Это интересно. Мы ожидаем, что слова, которые применяются в одинаковых контекстах, будут иметь похожие тематические векторы. Названия цветов, такие как *yellow* (желтый) и *red* (красный), скорее всего, будут выглядеть примерно одинаково в пространстве тем, в то время как *baseball* (бейсбол) и *sex* (секс) должны иметь довольно отдаленные отношения.

Обратите внимание, что эта матрица "слово-тема" потенциально полезна в любой задаче, стремящейся использовать язык в качестве признака. Связь с сообщениями социальных сетей в значительной степени исчезла, поэтому она будет применима к другим областям, таким как книги и новости. Действительно, такие сжатые *вектора представления слов* (word embedding) являются очень мощным инструментом в *обработке естественного языка* (Natural Language Processing — NLP), как будет обсуждаться в разделе 11.6.3.

8.3.2. Разложение LU матрицы и детерминанты

Разложение LU матрицы (LU decomposition) — это конкретное матричное разложение, которое делит квадратную матрицу A на нижнюю и верхнюю треугольные матрицы L и U, так что $A = LU$.

Матрица является *треугольной* (triangular), если она содержит все нулевые слагаемые либо выше, либо ниже основной диагонали. Нижняя треугольная матрица L имеет все ненулевые члены ниже главной диагонали. Другим разложением, U, является верхняя треугольная матрица. Поскольку главная диагональ L состоит из всех единиц, мы можем упаковать все разложение в то же пространство, что и исходная матрица $n \times n$.

Основное значение разложения LU матрицы заключается в том, что оно оказывается полезным при решении линейных систем $AX = Y$, особенно при решении нескольких задач с одним и тем же A, но разными Y. Матрица L — это результат очистки всех значений выше главной диагонали с помощью исключения по Гауссу. Оказавшись в этой треугольной форме, остальные уравнения могут быть упрощены непосредственно. Матрица U отражает, какие строковые операции произошли в ходе построения L. Упрощение U и применение L к Y требует меньше работы, чем решение A с нуля.

Другое значение разложения LU матрицы заключается в получении алгоритма для вычисления детерминанта матрицы. *Детерминант* (determinant) A является произведением основных диагональных элементов U. Как мы видели, нулевой детерминант означает, что матрица не имеет полного ранга.

На рис. 8.11 иллюстрируется разложение LU матрицы мемориала Линкольна. Существует четкая текстура, видимая для двух треугольных матриц. Эта конкретная функция разложения LU матрицы (в системе Mathematica) использовала тот факт, что уравнения в системе можно переставлять без потери информации. То же самое не относится к изображениям, но мы видим точные реконструкции белых колонн мемориала, хотя и не на своем месте.

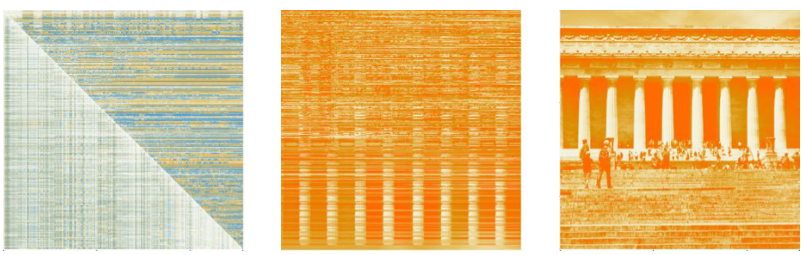

Рис. 8.11. Разложение LU матрицы мемориала Линкольна (слева) с произведением $L \cdot U$ (в центре). Строки матрицы LU были переставлены во время расчета, но при правильном порядке полностью реконструировали изображение (справа)

8.4. Собственные значения и собственные векторы

Умножение вектора U на квадратную матрицу A может иметь тот же эффект, что и умножение его на скаляр l. Рассмотрим следующую пару примеров. Действительно, проверьте их вручную:

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = -6 \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Оба эти равенства содержат произведения с одинаковым вектором U размером 2×1 как слева, так и справа. С одной стороны U умножается на матрицу A, а с другой — на скаляр λ. В подобных случаях, когда $AU = \lambda U$, мы говорим, что λ — это *собственное значение* (eigenvalue) матрицы A, а U — это связанный с ней *собственный вектор* (eigenvector).

Такие пары собственных векторов и собственных значений являются любопытной вещью. То, что скаляр λ может сделать то же самое с U, что и вся матрица A, говорит нам, что они должны быть специальными. Собственный вектор U и собственное значение λ должны кодировать много информации об A.

Кроме того, обычно существует несколько таких пар собственных векторов для любых матриц. Обратите внимание, что второй пример выше работает с той же матрицей A, но дает иное, чем U и λ.

8.4.1. Свойства собственных значений

Теория собственных значений ведет нас куда глубже в дебри линейной алгебры, чем я готов идти в этой книге. Однако мы можем резюмировать свойства, которые являются важными для нас.

- С каждым собственным значением связан собственный вектор. Они всегда ходят парами.
- В общем случае, для каждой матрицы $n \times n$ полного ранга существует n пар собственных векторов.
- Каждая пара собственных векторов симметричной матрицы взаимно *ортогональна* (orthogonal), так же, как ортогональны оси X и Y на плоскости. Два вектора ортогональны, если их скалярное произведение равно нулю. Заметьте, что $(0, 1) \cdot (1, 0) = 0$, как и $(2, -1) \cdot (1, 2) = 0$ из предыдущего примера.

- В результате этого собственные векторы могут играть роль измерений или *баз* (base) в некотором *n*-мерном пространстве. Это открывает много геометрических интерпретаций матриц. В частности, любая матрица может быть закодирована так, что каждое собственное значение представляет величину своего связанного собственного вектора.

8.4.2. Вычисление собственных значений

Можно найти *n* различных собственных значений матрицы ранга *n*, разложив ее *характеристическое уравнение* (characteristic equation). Начнем с определения равенства $AU = \lambda U$. Убедитесь, что оно остается неизменным, когда мы умножаем на единичную матрицу *I*, поэтому

$$AU = \lambda IU \rightarrow (A - \lambda I)U = 0.$$

Для нашего примера матрицы мы получаем

$$A - \lambda I = \begin{bmatrix} -5-\lambda & 2 \\ 2 & -2-\lambda \end{bmatrix}$$

Обратите внимание, что наше равенство $(A - \lambda I)U = 0$ остается верным, если мы умножим вектор *U* на любое скалярное значение *c*. Это подразумевает, что существует бесконечное количество решений, а следовательно, линейная система должна быть недоопределена.

В такой ситуации детерминант матрицы должен быть равен нулю. С матрицей 2×2 детерминант является просто перекрестным произведением $ad - bc$, поэтому

$$(-5-\lambda)(-2-\lambda) - 2 \cdot 2 = \lambda^2 + 7\lambda + 6 = 0.$$

Решение для λ с квадратичной формулой дает $\lambda = -1$ и $\lambda = -6$. В более общем смысле детерминант $|A - \lambda I|$ является многочленом степени *n*, а следовательно, корни этого характеристического уравнения определяют собственные значения *A*.

Вектор, связанный с любым заданным собственным значением, может быть вычислен в результате решения линейной системы. Согласно нашему примеру, мы знаем, что

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

для любого собственного значения λ и ассоциированного собственного вектора $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$. После того как мы зафиксировали значение λ, у нас будет система из

n уравнений и n неизвестных, и, таким образом, мы можем решить его для значений U. Для $\lambda = -1$

$$-5u_1 + 2u_2 = -1u_1 \rightarrow -4u_1 + 2u_2 = 0,$$
$$2u_1 + -2u_2 = -1u_2 \rightarrow 2u_1 + -1u_2 = 0.$$

это имеет решение $u_1 = 1$, $u_2 = 2$, что дает соответствующий собственный вектор.

Для $\lambda = -6$ получаем

$$-5u_1 + 2u_2 = -6u_1 \rightarrow 1u_1 + 2u_2 = 0,$$
$$2u_1 + -2u_2 = -6u_2 \rightarrow 2u_1 + 4u_2 = 0.$$

Эта система недоопределена, поэтому $u_1 = 2$, $u_2 = -1$ и любая постоянная, кратная этому, квалифицируется как собственный вектор. Это имеет смысл: поскольку U находится с обеих сторон равенства $AU = \lambda U$, для любой константы c вектор $U' = c\,U$ одинаково удовлетворяет определению.

Более быстрые алгоритмы для вычисления собственного значения/вектора основаны на таком подходе разложения матрицы, как *разложение QR* (QR decomposition). Другие алгоритмы пытаются избежать решения полной линейной системы. Например, альтернативный подход многократно использует $U' = (AU)/\lambda$, чтобы вычислять лучшие и лучшие приближения к U, пока он не сходится. Когда условия правильные, это может быть намного быстрее, чем решение полной линейной системы.

Наибольшие собственные значения и связанные с ними векторы, по правде говоря, более важны, чем остальные. Почему? Потому что они вносят больший вклад в аппроксимацию матрицы A. Таким образом, высокопроизводительные системы линейной алгебры используют специальные процедуры для нахождения k наибольших (и наименьших) собственных значений, а затем итерационные методы для восстановления векторов для каждого.

8.5. Разложение по собственным значениям

Любая симметричная матрица M размером $n \times n$ может быть разложена на сумму из n своих собственных векторных произведений. Представим n собственных пар (λ_i, U_i) для $1 \le i \le n$. По договоренности сортировка осуществляется по размеру, поэтому $\lambda_i \ge \lambda_{i-i}$ для всех i.

Поскольку каждый собственный вектор U_i является матрицей $n \times 1$, его умножение на транспозицию дает матричное произведение $U_i U_i^T$ размером $n \times n$. Оно имеет те же размеры, что и исходная матрица M. Мы можем вычислить линейную комбинацию этих матриц, взвешенную по соответствующему

собственному значению. Фактически это восстанавливает исходную матрицу, поскольку:

$$M = \sum_{i=1}^{n} \lambda_i U_i U_i^T .$$

Этот результат сохраняется только для симметричных матриц, поэтому мы не можем использовать его для кодирования нашего изображения. Но ковариационные матрицы всегда симметричны, и они кодируют основные характеристики каждой строки и столбца матрицы.

Таким образом, ковариационная матрица может быть представлена ее разложением по собственным значениям. Это занимает немного больше места, чем исходная матрица: n собственных векторов длиной n, плюс n собственных значений против $n(n + 1)/2$ элементов в верхнем треугольнике симметричной матрицы плюс главная диагональ.

Однако, используя только векторы, связанные с наибольшими собственными значениями, мы получаем хорошее приближение матрицы. Меньшие размеры вносят очень малый вклад в значения матрицы, поэтому их можно исключить с небольшой ошибкой. Этот метод уменьшения размеров очень полезен для создания меньших, более эффективных наборов объектов.

На рис. 8.12 (слева) показана реконструкция ковариационной матрицы M мемориала Линкольна по ее единственному наибольшему собственному вектору, т.е. $U_1 U_1^T$, вместе с ассоциированной с ней матрицей ошибок $M - U_1 U_1^T$. Даже один собственный вектор делает весьма существенную работу при реконструкции, восстанавливая такие функции, как большой центральный блок.

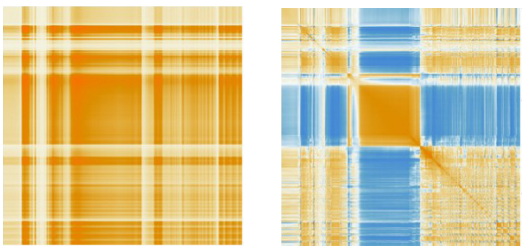

Рис. 8.12. Наибольшего собственного вектора мемориала Линкольна достаточно, чтобы захватить большую часть деталей его ковариационной матрицы

График на рис. 8.12 (справа) показывает, что ошибки возникают в неоднородных областях, поскольку для кодирования более тонких деталей требуются дополнительные векторы. На рис. 8.13 показан график ошибок при использовании одного, пяти и пятидесяти самых больших собственных векторов. Области ошибок становятся меньше, когда мы восстанавливаем более мелкие

детали, а величина ошибок уменьшается. Поймите, что даже 50 собственных векторов составляют менее 10% от 512, необходимых для восстановления идеальной матрицы, но этого достаточно для очень хорошего приближения.

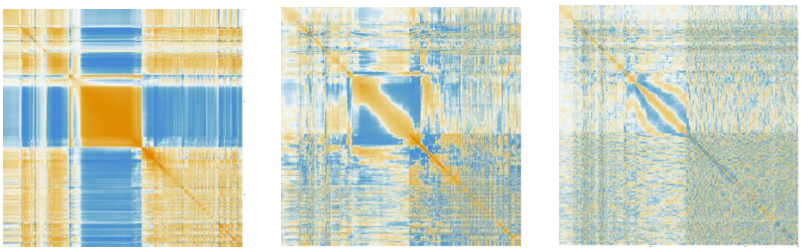

Рис. 8.13. Ошибка в реконструкции мемориала Линкольна по одному, пяти и пятидесяти самым большим собственным векторам

8.5.1. Разложение по сингулярному значению

Разложение по собственным значениям — это очень хорошая вещь. Но она работает только для симметричных матриц. *Разложение по сингулярным значениям* (singular value decomposition) является более общим подходом разложения матрицы, который аналогичным образом сводит матрицу к сумме других матриц, определенных векторами.

Разложение по сингулярным значениям действительной матрицы M размером $n \times m$ делит ее на три матрицы U, D и V с размерами $n \times n$, $n \times m$ и $m \times m$ соответственно. Это разложение имеет вид[3]

$$M = UDV^T.$$

Центральная матрица D обладает свойством *диагональной* матрицы, т.е. все ненулевые значения лежат на главной диагонали, как единичная матрица I.

Не беспокойтесь о том, как мы находим это разложение. Вместо этого давайте сосредоточимся на том, что это значит. Произведение UD имеет эффект умножения $U[i, j]$ на $D[j, j]$, поскольку все члены D равны нулю, кроме главной диагонали. Таким образом, D можно интерпретировать как меру относительной важности каждого столбца U или, через DV^T, важности каждой строки V^T. Эти весовые значения D являются *сингулярными значениями* (singular value) M.

Пусть X и Y — это векторы размерами $n \times 1$ и $1 \times m$ соответственно. *Внешним произведением* (outer product) $P = X \otimes Y$ матрицы является матрица $n \times m$, где $P[j, k] = X[j]Y[k]$. Традиционное матричное умножение $C = AB$ может быть выражено как сумма этих внешних произведений, а именно:

[3] Если M содержит выпуклые числа, то это обобщается до $M = UDV^*$, где V^* означает *сопряженное транспонирование* (conjugate transpose) V.

$$C = AB = \sum_k A_k \otimes B_k^T.$$

где A_k — это вектор, определенный k-м столбцом A, а B_k^T — это вектор, определенный k-й строкой B.

Таким образом, матрица M может быть выражена как сумма внешних произведений векторов, возникающих в результате разложения по сингулярным числам, а именно $(UD)_k$ и $(V^T)_k$ для $1 \le k \le m$. Кроме того, сингулярные значения D определяют, какой вклад каждое внешнее произведение вносит в M, поэтому достаточно взять только векторы, связанные с наибольшим сингулярным значением, чтобы получить приближение к M.

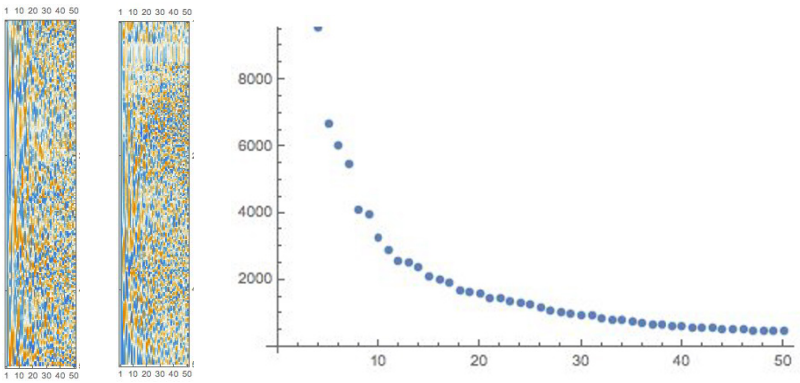

Рис. 8.14. Матрицы сингулярных значений в разложении Линкольна для 50 сингулярных значений

На рис 8.14 (слева) представлены векторы, связанные с первыми 50 сингулярными значениями портрета Линкольна. Если вы внимательно посмотрите, то увидите, что первые 50 векторов значительно более блочные, чем последующие, что указывает на то, что ранние векторы пролистывают базовую структуру матрицы, а последующие векторы добавляют больше деталей. На рис. 8.14 (справа) показано, как уменьшается среднеквадратичная ошибка между матрицей и ее реконструкцией при добавлении дополнительных векторов.

Эти эффекты становятся еще более яркими, когда мы смотрим на сами восстановленные изображения. На рис. 8.15 (слева) показано лицо Линкольна с пятью самыми сильными векторами, что составляет менее 1% от того, что доступно для идеальной реконструкции. Но даже в этот момент вы вполне сможете узнать его на опознании в полиции. На рис. 8.15 (в центре) демонстрируется бо́льшая детализация, когда мы включаем 50 векторов. Это выглядит так же хорошо, как исходное изображение, при печати, хотя график ошибок (рис. 8.15 (справа)) подчеркивает недостающие детали.

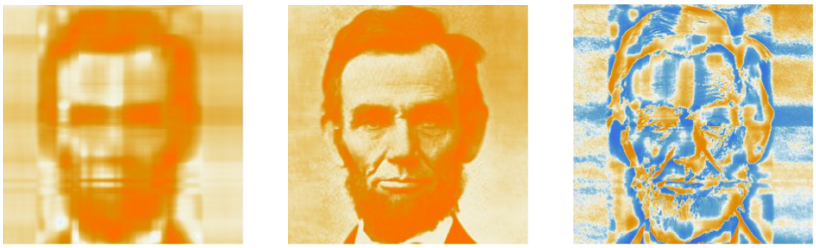

Рис. 8.15. Лицо Линкольна реконструировано из 5 (слева) и 50 (в центре) сингулярных значений с погрешностью для k = 50 (справа)

На заметку. Разложение по сингулярным значениям (Singular Value Decomposition — SVD) — это мощный метод уменьшения размерности любой матрицы признаков.

8.5.2. Анализ основных компонентов

Анализ основных компонентов (Principal Components Analysis — PCA) — это техника, тесно связанная с уменьшением размерности наборов данных. Подобно SVD, мы будем определять векторы для представления набора данных. Как и SVD, мы будем упорядочивать их по важности, чтобы можно было восстановить приблизительное представление, используя несколько компонентов. PCA и SVD настолько тесно связаны, что их невозможно отличить для наших задач. Они делают то же самое одинаково, но с разных сторон.

Основные компоненты определяют оси эллипсоида, наилучшим образом подходящие к точкам. Источником этого набора осей является центр тяжести точек. PCA начинает с определения направления, на которое нужно проецировать точки, чтобы объяснить максимальное количество отклонений. Это линия, проходящая через центр тяжести, которая в некотором смысле лучше всего подходит для точек, что делает ее аналогом линейной регрессии. Затем мы можем спроецировать на эту линию каждую точку, причем точка пересечения определяет конкретную позицию на линии относительно центроида (centroid). Эти проецируемые позиции теперь определяют первое измерение (или основной компонент) нашего нового представления, как показано на рис. 8.16.

Для каждого последующего компонента мы ищем линию l_k, которая ортогональна всем предыдущим линиям и объясняет наибольшее количество оставшейся дисперсии. То, что каждое измерение ортогонально друг другу, означает, что они действуют как оси координат, устанавливая связь с собственными векторами. Каждое последующее измерение постепенно становится менее

важным, чем предшествующее ему, потому что сначала мы выбрали наиболее перспективные направления. Более поздние компоненты лишь постепенно вносят более мелкие детали, и, следовательно, мы можем остановиться, когда они станут достаточно малы.

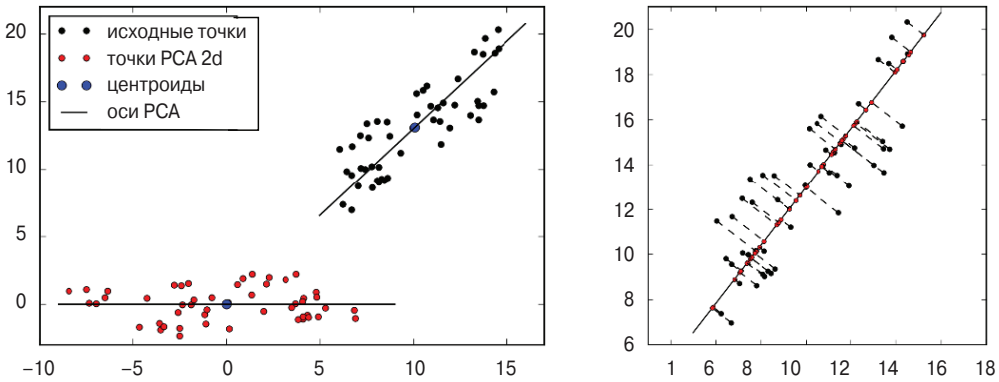

Рис. 8.16. PCA проецирует черные точки на ортогональную ось, повернутую так, чтобы получить альтернативное представление в красном (слева). Значения каждого компонента задаются проекцией каждой точки на соответствующую ось (справа)

Предположим, что размеры x и y практически идентичны. Мы ожидаем, что линия регрессии будет проецироваться вниз до $y = x$ в этих двух измерениях, поэтому их можно будет в значительной степени заменить одним измерением. PCA конструирует новые измерения как линейные комбинации исходных, объединяя те, которые сильно коррелируют, в пространство меньшего измерения. *Статистический факторный анализ* (statistical factor analysis) — это метод, который идентифицирует наиболее важные ортогональные измерения (измеряемые с помощью корреляции), которые объясняют бо́льшую часть дисперсии.

Относительно небольшого числа компонентов достаточно, чтобы охватить базовую структуру набора точек. Остаток, вероятно, является шумом, и как правило лучше удалить его из данных. После уменьшения размеров с помощью PCA (или SVD) мы должны получить более чистые данные, а не просто меньшее количество измерений.

> *На заметку*. PCA и SVD — это два разных подхода к вычислению одного и того же. Они должны служить одинаково хорошо для малоразмерных приближений матрицы признаков.

8.6. Случай из жизни: человеческий фактор

Впервые я был поражен мощью методов уменьшения размеров, таких как PCA и SVD, в ходе анализа исторических фигур для нашей книги *Who's Bigger*. Помните (раздел 4.7 "Случай из жизни"), как мы анализировали структуру и содержание Википедии, выделив в конечном итоге полдюжины функций, таких как PageRank, и длину статьи, для каждой из 800 000 + статей о людях в английском издании? Это свело каждую из персон к шестимерному вектору признаков, который мы проанализировали, чтобы оценить их относительную значимость.

Но все оказалось не так просто, как мы думали. Существенно разные люди были оценены выше по каждой конкретной переменной. Было непонятно, как их интерпретировать.

"В наших функциях так много различий и случайных шумов, — заметил мой соавтор Чарльз. — Давайте определим основные факторы, лежащие в основе этих наблюдаемых переменных, которые действительно показывают, что происходит".

Решением Чарльза был *факторный анализ* (factor analysis), являющийся вариантом PCA, который, в свою очередь, является вариантом SVD. Все эти методы сжимают матрицы признаков в меньший набор переменных, или факторов, так, чтобы эти факторы объясняли б*о*льшую часть различий в полной матрице признаков. Мы ожидали, что факторный анализ извлечет один основной фактор, определяющий индивидуальную значимость. Но вместо этого наши входные переменные дали *два* независимых фактора, объясняющих данные. Оба объяснили примерно одинаковые пропорции дисперсии (31% и 28%), что означает, что эти скрытые переменные были примерно одинаковой важности. Но самое интересное то, что показали эти факторы.

Факторы (или единичные векторы, или основные компоненты) являются просто линейными комбинациями оригинальных входных функций. Как правило, у них нет имен, поэтому обычно вы просто описываете их как Фактор 1 и Фактор 2. Но два наших фактора были настолько разными, что Чарльз дал им названия *gravitas* (авторитетность) и *celebrity* (известность), и вы можете понять на рис. 8.17 почему.

Наш фактор *gravitas* в значительной степени происходит от (или "загружается" в статистическом смысле) двух форм PageRank. Авторитетность, кажется, точно отражает понятия признания, основанного на достижениях. Фактор *celebrity*, напротив, сильнее влияет на количество просмотров страниц, правок и длин статей. Фактор известности лучше отражает популярные (некоторые могут сказать — вульгарные) представления о репутации. Популярность певцов, актеров и других артистов лучше измеряется известностью, чем авторитетностью.

Наивысший рейтинг по авторитетности

Личность	Авт.	Знач.	Изв./Авт.
Наполеон	8	2	C G
Карл Линней	13	31	C G
Платон	23	25	C G
Аристотель	27	8	C G
Ф. Д. Рузвельт	30	43	C G
Плутарх	32	258	C G
Карл II	33	78	C G
Елизавета II	35	132	C G
Королева Виктория	38	16	C G
Уильям Шекспир	42	4	C G
Плиний Старший	43	212	C G
Тацит	52	300	C G
Геродот	58	123	C G
Карл V	61	84	C G
Георг V	64	235	C G

Наивысший рейтинг по известности

Личность	Изв.	Знач.	Изв./Авт.
Гробовщик (рестлер)	2	2172	C G
Виджай	8	4456	C G
Эдж	10	2603	C G
Кейн	13	2229	C G
Джон Сина	16	2277	C G
Бейонсе	19	1519	C G
Triple H	26	1596	C G
Рей Мистерио	36	2740	C G
Бритни Спирс	37	689	C G
Энн Коултер	45	3376	C G
Джесси Маккартни	48	4236	C G
Роджер Федерер	57	743	C G
Эшли Тисдейл	60	4445	C G
Майкл Джексон	75	180	C G
Дуэйн Джонсон	78	1446	C G

Рис. 8.17. Факторы gravitas и celebrity отлично справляются с разделением двух типов известных людей

Чтобы почувствовать разницу между авторитетностью и известностью, сравните наши самые высокие показатели для каждого фактора на рис. 8.17. Верхние фигуры слева — это явно старомодные тяжеловесы, люди высокого ранга и достижений. Это философы, короли и государственные деятели. Имена, перечисленные на рис. 8.17 (справа), настолько известны, что четверке лучших достаточно на этой земле только имени. Это профессиональные борцы, актеры и певцы. Весьма показательно, что только две фигуры изображают авторитетность на нашем измерителе знаменитости, это *Бритни Спирс* (1981–) [566] и *Майкл Джексон* (1958–2009) [136], оба среди платоновских идеалов современной знаменитости.

Я нахожу удивительным, что эти методы обучения без учителя были способны разделить два разных типа славы без каких-либо помеченных обучающих примеров или даже предвзятого мнения о том, что они искали. Факторы/векторы/компоненты просто отражают то, что было в данных, которые будут найдены.

Этот континуум известности-авторитетности служит поучительным примером силы методов уменьшения размеров. Все факторы/векторы/компоненты должны по определению быть ортогональными друг другу. Это означает, что каждый из них измеряет разные вещи так, как не делают две коррелированные входные переменные. Вам стоит провести некоторый предварительный анализ данных по основным компонентам, чтобы попытаться выяснить, что они на самом деле означают в контексте вашего приложения. Вы можете присваивать

имена по своему усмотрению, хотите — кошки, хотите — собаки, поэтому выбирайте те имена, с которыми вам будет приятно жить.

8.7. Дополнительная информация

Существует много популярных учебников, посвященных линейной алгебре, в том числе [75, 76, 77]. Кляйн [78] представляет интересное введение в линейную алгебру для информатики с упором на программирование и приложения, такие как теория кодирования и компьютерная графика.

8.8. Упражнения

Основы линейной алгебры

8.1. *[3]* Придумайте пару квадратных матриц A и B так, чтобы:
(a) $AB = BA$ (коммутативно).
(b) $AB \neq BA$ (не коммутативно).
В общем случае умножение матриц не является коммутативным.

8.2. *[3]* Докажите, что сложение матриц ассоциативно, т.е. $(A + B) + C = A + (B + C)$ для совместимых матриц A, B и C.

8.3. *[5]* Докажите, что матричное умножение ассоциативно, т.е. $(AB)C = A(BC)$ для совместимых матриц A, B и C.

8.4. *[3]* Докажите, что $AB = BA$, если A и B — это диагональные матрицы одного порядка.

8.5. *[5]* Докажите, что если $AC = CA$ и $BC = CB$, то $C(AB + BA) = (AB + BA)C$.

8.6. *[3]* Являются ли матрицы MM^T и M^TM квадратными и симметричными? Объясните.

8.7. *[5]* Докажите, что $(A^{-1})^{-1} = A$.

8.8. *[5]* Докажите, что $(A^T)^{-1} = (A^{-1})^T$ для любой несингулярной матрицы A.

8.9. *[5]* Является ли LU разложение матрицы уникальной? Обоснуйте ответ.

8.10. *[3]* Объясните, как решить матричное уравнение $Ax = b$.

8.11. *[5]* Покажите, что если M является квадратной матрицей, которая не является обратимой, то либо L, либо U в LU-разложении $M = LU$ имеет нуль на своей диагонали.

Собственные значения и собственные векторы

8.12. *[3]* Пусть $M = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$. Найдите все собственные значения M. Имеет ли M два линейно независимых собственных вектора?

8.13. *[3]* Докажите, что собственные значения A и A^T одинаковы.

8.14. *[3]* Докажите, что собственные значения диагональной матрицы равны диагональным элементам.

8.15. *[5]* Предположим, что матрица A имеет собственный вектор v с собственным значением λ. Покажите, что v является также собственным вектором для A^2, и найдите соответствующее собственное значение. Как насчет A^k, для $2 \leq k \leq n$?

8.16. *[5]* Предположим, что A — это обратимая матрица с собственным вектором v. Покажите, что v является также собственным вектором для A^{-1}.

8.17. *[8]* Покажите, что собственные значения MM^T такие же, как у M^TM. Их собственные векторы также одинаковы?

Реализация проектов

8.18. *[5]* Сравните скорость библиотечной функции для умножения матриц с собственной реализацией алгоритма вложенных циклов.

- Насколько быстрее библиотека на произведениях случайных матриц $n \times n$ как функция от n, когда n становится большим?
- Как насчет произведения матрицы $n \times m$ и $m \times n$, где $n \ll m$?
- Насколько вы повышаете производительность своей реализации для вычисления $C = AB$, сначала транспонируя внутреннее значение B, чтобы все скалярные произведения вычислялись по строкам матриц для повышения производительности кеша?

8.19. *[5]* Реализация исключения по Гауссу для решения систем уравнений, $CX = Y$. Сравните вашу реализацию с популярной библиотечной подпрограммой по следующим параметрам.

(a) *Скорость*. Как сравнивается время выполнения для матриц с плотными и разреженными коэффициентами?

(b) *Точность*. Каковы размеры числовых остатков $CX - Y$, особенно при увеличении числа условий матрицы.

(c) *Стабильность*. Происходит ли сбой вашей программы в сингулярной матрице? Как насчет почти сингулярных матриц, созданных в результате добавления небольшого случайного шума к сингулярной матрице?

Вопросы на интервью

8.20. *[5]* Почему векторизация считается мощным методом оптимизации числового кода?

8.21. *[3]* Что такое разложение по сингулярным числам? Что такое сингулярное значение? Что такое сингулярный вектор?

8.22. *[5]* Объясните разницу между “длинным” и “широким” форматом данных. Когда каждый из них может применяться на практике?

Конкурсы Kaggle

8.23. Расскажите, кто на что смотрит, исходя из анализа их мозговых волн.
`https://www.kaggle.com/c/decoding-the-human-brain`

8.24. Решите, будет ли конкретный студент правильно отвечать на заданный вопрос.
`https://www.kaggle.com/c/WhatDoYouKnow`

8.25. Определите пользователей мобильных телефонов по данным акселерометра.
`https://www.kaggle.com/c/accelerometer-biometric-competition`