基本线性代数和矩阵

一个问题

Fibonacci 数列满足公式 $F_n = F_{n-1} + F_{n-2}$ ，这个数列的前几项是

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597 ...

计算比如第100项 F_{100} 是一个常见递归编程题，各位白帽子基本上都会做，也知道通过某些技巧避免无效计算。可是各位有没有考虑过直接计算 F_{100} 的非递归方式呢?

其实白帽子朋友们可能更疑惑的是，这个不是一个编程问题么，它和线性代数及矩阵有什么关系?

线性方程和方程组

线性的意义

一个最简单的线性方程看起来是这样的，这里 a b 是常量，x y是变量:

$$a \cdot x + b = y$$

线性方程的意义在于所有变量的关系都是一次方，即使有 n 个不同的变量 x 比如:

$$a_0 x_0 + a_1 x_1 + a_2 x_2 + \ldots + a_{n-1} x_{n-1} \quad = \quad y$$

这个方程有个简单的写法，用了 $\sum$ 符号和上下标表示求和:

$$\sum_{i=0}^{n-1} a_i x_i \quad = \quad y$$

不仅 x 可以很多，y 也可以很多，这个线性方程就成了方程组，比如是这样（简化起见去掉了常数项）:

$$\begin{aligned}
a_{00} x_0 + a_{01} x_1 + a_{02} x_2 + \ldots + a_{0,n-1} x_{n-1} &= y_0 \\
a_{10} x_0 + a_{11} x_1 + a_{12} x_2 + \ldots + a_{1,n-1} x_{n-1} &= y_1 \\
a_{20} x_0 + a_{21} x_1 + a_{22} x_2 + \ldots + a_{2,n-1} x_{n-1} &= y_2 \\
\ldots & \\
a_{m-1,0} x_0 + a_{m-1,1} x_1 + a_{m-1,2} x_2 + \ldots + a_{m-1,n-1} x_{n-1} &= y_{n-1}
\end{aligned}$$

这样的方程组也有一种更加清晰的写法，比如：

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \vdots & \ddots & & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

在这里，我们可以定义 A 等于

$$A = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \vdots & \ddots & & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$$

同样的定义 X 和 Y ，这样，这个看似复杂的方程组就可以写成

$$AX = Y$$

这就是常见的线性代数的表达方式。这里 A , X 和 Y 称为矩阵。如果把矩阵表达方式还原为线性方程组，只需要把 A 的每一行和 X 相乘就得到 Y 每一行的值，比如：

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \ldots + a_{0,n-1}x_{n-1}$$

练习题

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 6 & 3 \\ 9 & 5 & 3 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

当 X 取值为如下时，Y 矩阵是什么？

$$\begin{bmatrix} 7 \\ 4 \\ 7 \\ 2 \end{bmatrix}$$

可以手工计算，也可以写 `python` 程序计算。

白帽子朋友们也可以思考一下这个问题的逆问题：如果已知 Y 和 A ，怎么求 X 呢？由 A X 求 Y 很直接但是从 A 和 Y 求 X 是一个比较难的问题，请在深入阅读部分继续探索线性代数的其他内容之后自行得到答案。

矩阵

矩阵就是一个 $m \times n$ 个元素的方阵，比如这就是一个 2×2 的矩阵：

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

有白帽子朋友就会问，矩阵能是立体的或者更多维度么？那个叫做张量（“Tensor”），也有自己的一套数学方法，请自行学习在此不做更多表述。

矩阵的常见计算

对于白帽子的数据科学工作来说，常见矩阵计算里比较重要的是矩阵乘法和矩阵分解。

矩阵乘法

TBD

练习题

TBD

矩阵分解

TBD

练习题

TBD

矩阵计算的意义

矩阵计算的意义在于把数据的复杂操作转换成计算，并可以通过矩阵分解等操作表达数据里抽象的特征。

Fibonacci 数列问题

矩阵计算的意义是把“动作”和“操作”这些行为变得可以计算，比如说求 Fibonacci 特定项的这个递归操作可以转换成矩阵乘法的序列，从而可以用计算机简单的编码完成复杂的操作。

如果定义

$$U_n = \begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix}$$

那么把 Fibonacci 数列的公式 $F_n = F_{n-1} + F_{n-2}$ 表达成矩阵方式只需要定义 $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ 然后 $U_n = A \times U_{n-1}$ 就等价于 $F_n = F_{n-1} + F_{n-2}$ 。这样矩阵表达方式的好处在于把 F_n 求值的递归公式表述成如下矩阵 A 的 n 次方：

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

这样一个复杂的递归问题变成了一个简单矩阵循环自己乘自己 n 次即可，编码难度极度降低。这个 Fibonacci 数列矩阵称之为 Fibonacci Q-Matrix，它的计算方法类似于求一个数的n 次方，可以方便的在 $\mathrm{O}(\log n)$ 时间复杂度完成，参见这个 `python` 代码范例。

其他矩阵乘法的例子

同样的矩阵方法应用到神经网络的求解计算，比如现在最火热的话题“深度学习”里面的神经网络自动求导，它其实就是利用矩阵乘法表达了求导的链式规则，把每一层的值和导数放在同一个矩阵里，经过矩阵的多次乘法既传播了神经元的值也同时完成了求导的链式规则，用矩阵乘法大大简化了编码复杂度。关于神经网络的相关知识，后面的机器学习部分会提到。

更多的矩阵计算表述操作的例子还有，如果白帽子朋友做过图像处理也会知道，图像的旋转和缩放等操作可以通过乘以相应矩阵完成；魔方的还原也可以通过矩阵乘法操作完成；一些移位和替换类的加密解密操作也是通过矩阵乘法完成。请各位白帽子自行阅读探索。

复习一个知识点：求导的链式规则

如果一个函数 f 是对 y 的函数写成 $f(y)$ ，而 y 是对 x 的函数，那么 $f(y)$ 对于 x 的导数可以先对 y 求导再乘以 y 对 x 求导即可，称之为“链式规则”：

$$\frac{d}{dx}[f(y)] = \frac{d}{dy}[f(y)]\frac{dy}{dx}$$

对于多层的函数嵌套，只需要每一层乘以相应的求导链即可，这正好和矩阵乘法表示一致。对于矩阵 $F(Y)$ ，后面机器学习部分会提到矩阵求导。

图像水印问题

矩阵的另一个意义是它是信息表达的一种抽象方式。比如图像的每个像素点值对应于矩阵的每个元素，那么对于矩阵分解等操作会得到该图像抽象特征。

TBD

深入阅读

TBD

想法笔记