线性代数引论

itpyi
大唐西京慈恩翻译学院

创建日期： 2025-09-28
修改日期： 2025-09-28

目录

线性代数用代数方法研究线性关系. 线性关系可以看作正比例关系的推广. 代数方法是一种以抽象化、统一化为核心的数学方法, 它从不同问题中提取相似的结构, 在抽象层面上研究这些结构, 以此透过各类数学问题纷繁复杂的表象, 揭示其内在的本质联系.

在这篇引论中, 我们通过两个简单的例子, 使读者对于用代数方法研究线性关系获得一个初步的印象.

1. 兔子数列的通项公式

假设有一对兔子, 它们从出生后第 2 个月起, 每个月都生一对兔子. 兔子永远不会死. 问: 第 n 个月末, 兔子总共有多少对? 这个数列的前几项是:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... \tag{1}$$

我们想要找到这个数列的通项公式. 为此, 不难写出递推关系:

$$F(n) = F(n-1) + F(n-2). \tag{2}$$

我们希望从这个递推关系中求解通项公式.

观察发现, 这个递推关系满足如下性质:

- 如果 $x(n)$ 满足递推关系, 那么 $cx(n)$ 也满足递推关系, 其中 c 是任意常数.
- 如果 $x(n)$ 和 $y(n)$ 都满足递推关系, 那么 $x(n) + y(n)$ 也满足递推关系.

我们称上述两条性质为线性性质. 我们不妨先一般地研究满足线性性质的递推关系.

让我们从最简单的例子入手, 考虑只含有相邻两项的递推关系. 不难验证, 等比数列的递推关系 $A(n) = rA(n-1)$ 是线性的, 它的通项公式是 $A(n) = r^n A(0)$. 我们反过来问另外一个问题, 考虑相邻两项的递推关系 $A(n) = R(A(n-1))$, 在 R 是什么样的函数时, 递推关系是线性的? 由线性性质可知, R 必须满足:

- $R(cx) = cR(x)$,
- $R(x + y) = R(x) + R(y)$.

由第一个条件就能推出, $R(x) = xR(1)$, 即 R 是一个正比例函数, 其比例系数为 $R(1)$. 由此可见, 线性性质将仅包含相邻两项的递推关系严格地限制在了等比数列的范围内.

我们现在完全理解了仅含相邻两项的线性递推关系, 这是一个巨大的进展!

和等比数列相比, 兔子数列的递推关系多了一项. 我们能否通过一些等价变换, 将兔子数列的递推关系化为仅含相邻两项的递推关系? 可以, 但是有代价. 我们引入一个新的数列 $G(n)$, 定义为 $G(n) = F(n-1)$. 由此, 我们得到了仅含相邻两项的递推关系:

$$\begin{cases} F(n) = F(n-1) + G(n-1), \\ G(n) = F(n-1). \end{cases} \tag{3}$$

代价是一个数列变成了两个数列, 并且递推关系是缠绕在一起的——优雅的说法是"耦合"的. 不过仍有两个好消息.

一是 G 只是将 F "平移" 了一项, 所以 G 也满足兔子数列的递推关系. 由线性性质知, 任何组合 $aF(n) + bG(n)$ 也满足兔子数列的递推关系. 这给我们带来了相当大的自由度, 因为我们可以研究任意组合 $aF(n) + bG(n)$, 而不必局限于 F 或 G. 我们可以引入变换

$$\begin{cases} F' = aF + bG, \\ G' = cF + dG, \end{cases} \tag{4}$$

由此得到的 F' 和 G' 也满足兔子数列的递推关系.

二是线性性质并没有消失. 不过现在的线性性质变成了一个二元的线性性质:

- 如果数列对 $(f(n), g(n))$ 满足递推关系, 那么 $(cf(n), cg(n))$ 也满足递推关系, 其中 c 是任意常数.
- 如果数列对 $(f_1(n), g_1(n))$ 和 $(f_2(n), g_2(n))$ 都满足递推关系, 那么 $(f_1(n) + f_2(n), g_1(n) + g_2(n))$ 也满足递推关系.

更值得庆贺的是, 变换 (4) 是保持二元线性性质的: 它得出的 F' 和 G' 的递推关系仍然满足二元的线性性质.

这给了我们一个思路. 通过调整变换 (4) 的参数 a, b, c, d, 我们可以得到各种各样的数列对 (F', G'), 它们一方面各自满足兔子数列的递推关系, 另一方面一起满足一些形式各异的二元线性递推关系. 最简单的二元线性递推关系是两个不耦合的一元递推关系——两个等比数列. 因此我们期待存在一组参数 a, b, c, d, 使得变换 (4) 将递推关系解耦:

$$\begin{cases} F'(n) = r_1 F'(n-1), \\ G'(n) = r_2 G'(n-1). \end{cases} \tag{5}$$

如果成功, 我们就得到了两个满足兔子数列递推关系的等比数列 $F'(n) = r_1^n F'(0)$, $G'(n) = r_2^n G'(0)$. 我们实际上不必真的计算这个解耦的过程, 而只需要将结果代入兔子数列的递推关系求解 r_1 和 r_2. 由于递推关系是线性的, 代入时也不必考虑 $F'(0)$ 和 $G'(0)$. 于是我们得到方程:

$$r^2 = r + 1. \tag{6}$$

注意这个方程对 r_1 和 r_2 是相同的. 这个方程的解是

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}. \tag{7}$$

因此, 我们得到了

$$F'(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n F'(0), \quad G'(n) = \left(\frac{1-\sqrt{5}}{2}\right)^n G'(0). \tag{8}$$

一般地, 我们知道 $sr_1^n + tr_2^n$ 满足兔子数列的递推关系.

现在我们希望得到以 $F(0) = F(1) = 1$ 为初始条件的兔子数列. 不妨假设所求数列具有 $F(n) = sr_1^n + tr_2^n$ 的形式. 代入初始条件, 我们得到

$$\begin{cases} s + t \quad\quad = 1, \\ sr_1 + tr_2 = 1. \end{cases} \tag{9}$$

解之可得

$$s = \frac{r_1}{\sqrt{5}}, \quad t = -\frac{r_2}{\sqrt{5}}. \tag{10}$$

于是得到通项公式

$$F(n) = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right). \tag{11}$$

问题解决!

我们的最后一步实际上有些试探的成分: 我们并没有证明所求数列一定具有 $F(n) = sr_1^n + tr_2^n$ 的形式, 虽然计算结果显示前两项为 1, 1 时数列的确能写成这个形式. 我们现在来补上这个缺陷. 由于兔子数列的每一项是由前两项唯一决定的, 所以确定了数列的前两项就确定了整个数列. 假设数列的前两项是 $F(0)$ 和 $F(1)$, 那么为了得到 s, t 我们就要解方程

$$\begin{cases} s + t \quad\quad = F(0), \\ sr_1 + tr_2 = F(1). \end{cases} \tag{12}$$

如果这个方程一定有解, 我们就得到了通项公式. 通过消元法, 不难看出方程的解为

$$s = \frac{F(1) - r_2 F(0)}{r_1 - r_2}, \quad t = \frac{r_1 F(0) - F(1)}{r_1 - r_2}. \tag{13}$$

只要 $r_1 \neq r_2$, 方程就一定有解. 这个条件是成立的, 因此我们得到的两个等比数列确实可以生成出任意初始条件的兔子数列.

回顾我们处理这个问题的过程, 我们首先抽象出了线性性质, 研究了单个变元的线性递推关系, 然后将兔子数列的递推关系化为了二元线性递推关系, 并且通过保持线性性质的变换将双变元的递推关系解耦为两个单变元的递推关系, 最后通过代入初始条件求解通项公式. 在解耦的过程中, 我们并没有真的计算具体的变换, 而是通过解一个多项式方程得到了解耦后的系数. 读者不难根据结果自己计算出具体的变换. 敏感的读者可能会产生疑问: 解耦一定能成功吗? 如果解出来 $r_1 = r_2$ 怎么办? 我们不妨看看另一个例子.

考虑递推关系 $H(n) = 2H(n-1) - H(n-2)$. 这一递推关系实际上就是等差数列的递推关系. 这个递推关系同样是线性的, 我们也可以将其化为二元线性递推关系:

$$\begin{cases} H(n) = 2H(n-1) - I(n-1), \\ I(n) \ \ = H(n-1). \end{cases} \tag{14}$$

我们同样可以尝试解耦, 代入 $H'(n) = r_1 H'(n-1)$, $I'(n) = r_2 I'(n-1)$, 得到方程

$$r^2 = 2r - 1. \tag{15}$$

这个方程的解是 $r_1 = r_2 = 1$. 也就是说, 我们无法将递推关系解耦为两个不同的等比数列. 如果试图用等比数列来生成满足该递推关系的数列, 我们只能得到常数数列, 而得不到公差非零的等差数列. 如果我们尝试解耦, 我们会发现尽最大努力也只能做到部分解耦:

$$\begin{cases} H'(n) = H'(n-1) + I'(n-1), \\ I'(n) \ = I'(n-1), \end{cases} \tag{16}$$

其中 $H' = H$, $I' = H - I$. 由此可见, 我们只能得到一个等比数列 I', 而 H' 仍然是耦合的. 这时我们可以通过观察发现, H' 是一个等差数列. 我们仍然可以得到通项公式, 只是过程没有之前那么优雅了. 读者可以尝试一个更邪恶的例子: $H(n) = 4H(n-1) - 4H(n-2)$. 这一递推关系的解是 $H(n) = (A + Bn)2^n$. 此时, 即使已经部分解耦得到

$$\begin{cases} H'(n) = 2H'(n-1) + I'(n-1), \\ I'(n) \ = 2I'(n-1), \end{cases} \tag{17}$$

要看出这个一般解也需要较强的注意力.

上述两个例子说明, 解耦不一定能成功, 解耦失败的情况和多项式方程有重根的情况有着千丝万缕的联系. 我们已经用结构化分析的方法取得了很大的成功, 读者也许会期待, 同样结构化的方法也能帮助我们处理解耦失败的情况. 事实上, 情况的确如此, 具体的解决方案, 读者将在学完线性变换、特征值、特征多项式、幂零映射、Jordan 标准型等内容后, 在习题中重新看到这个问题, 那时一定会有不一样的体验.

2. 纠错码的构造

我们再来看一个貌似截然不同的问题. 现在小甲要给小乙发送一串数字, 每个数字在 0 和 1 中取值. 中间有一个小丙, 他会随机地选择什么都不做或者翻转其中一位数字, 也就是将 0 变成 1, 或者将 1 变成 0. 请问小甲应该如何让小乙接收到正确的信息?

我们先讨论最简单的情况: 小甲要发送的数字只有一位.

- 小甲可以直接发送 0 或 1, 但是这一小乙就无法知道小丙是否翻转了这一位数字.
- 为了让小乙能够知道小丙是否翻转了数字, 小甲可以发送两位数字, 00 或 11, 分别代表他希望发送的 0 和 1. 此时如果小丙翻转了其中一位, 小乙收到的数字就是 01 或 10, 他就能知道小丙翻转了数字. 但是他无法知道小甲到底想发送 0 还是 1, 因为 01 既可以由 00 翻转而来, 也可以由 11 翻转而来.
- 小甲可以发送三位数字, 000 或 111, 分别代表他希望发送的 0 和 1. 此时如果小丙翻转了其中一位, 小乙就会看到有一位数字和另外两位数字不同. 这样小乙就能知道小丙翻转了数字, 并且他也能知道小甲到底想发送 0 还是 1, 因为 001, 010, 100 都只能由 000 翻转而来, 而 110, 101, 011 都只能由 111 翻转而来.

容易想象, 如果小甲希望发送更多位数字, 他只需要重复第三个方案即可.

但是这样效率就太低了. 比如说小甲希望发送 4 位数字, 那么他就要发送 12 位确保小乙能正确接收. 是否有更高效的方案? 我们从前面的观察发现, 为了让小乙能够正确解读小甲发送的信息, 任何两个未被翻转的数字串之间必须至少有三位不同. 于是小甲需要 16 个数字串, 每两个数字串之间至少有三位不同, 他用它们来分别表示 4 位数字的 16 种可能. 小甲设计好方案之后, 做了两张表, 一张给自己一张给小乙, 表中列出了 16 个数字串分别对应的 4 位数字. 见表 1.

希望发送的数字	发送的数字串
0000	0000000
0001	0000111
0010	0011001
0011	0011110
0100	0101010
0101	0101101
0110	0110011
0111	0110100
1000	1001011
1001	1001100
1010	1010010
1011	1010101
1100	1100001
1101	1100110
1110	1111000
1111	1111111

表 1. 小甲和小乙的编码表

我们先不考虑小乙如何解读, 只考虑小甲选择字符串的成本. 小甲需要存储这样一张大表, 这张表有 $2^4 = 16$ 行, 每行 7 位, 总共要存 $16 \times 7 = 112$ 位. (小甲可以不写左边那一栏, 需要决定发送什么数字时, 只需要从他想发的信息左边的数字开始二分查找即可.) 可以想象, 如果小甲希望发送 N 位数字, 他就需要存储一张有 2^N 行, 每行 M 位的表格, 其中 M 是小甲选择的字符串的长度. 这样小甲就需要存储 $2^N \times M$ 位数字.

这样的指数爆炸显然是不可接受的. 实际上我们有更高效的方案. 我们可以画一个三角形, 如图 1. 取其顶点、边中点和面心, 共 7 个点, 编号为 0 到 6, 这些编号表示小甲要发送的 7 位数字串的码位: 小甲把第 0 位放在 0 点, 第 1 位放在 1 点, 以此类推. 小甲要求: 要发送的 7 位数字串需要满足: 把它们放在三角形上后, 每个四边形的四个顶点 (比如 0, 4, 5, 6) 上的数字之和是偶数.

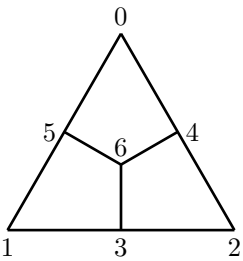

图 1. Hamming 编码

这竟然可以? 我们来一步步分析.

首先, 我们说明, 满足条件的数字串中任意两个一定有至少三位不同. 这是因为, 考虑我们要从一个满足条件的数字串变换到另一个.

- 如果我们翻转了 1 位数字, 这个点所在的四边形的四个顶点上数字之和就会变成奇数.

- 如果我们翻转了 2 位, 不难验证也会有四边形的四个顶点上数字之和变成奇数.
- 如果我们翻转了 3 位, 那么可以得到另一个满足条件的数字串, 比如翻转 1, 2, 3 位.

这说明我们可以保证信息可以被小乙正确解读.

其次, 我们说明, 满足条件的数字串一共有 16 个. 这是因为, 小甲可以在比如 1, 2, 3, 4 位上放他想发送的 4 位数字, 比如 1000, 于是七位数确定了四位:

$$(?, 1, 0, 0, 0, ?, ?). \tag{18}$$

根据 2, 3, 4 位的数字, 可以确定 6 上要放的数字, 上例变成

$$(?, 1, 0, 0, 0, ?, 0). \tag{19}$$

再根据 1, 3, 6 位上的数字, 可以确定 5 上要放的数字:

$$(?, 1, 0, 0, 0, 1, 0). \tag{20}$$

最后根据 4, 5, 6 位上的数字, 可以确定 0 上要放的数字:

$$(1, 1, 0, 0, 0, 1, 0). \tag{21}$$

一旦 1, 2, 3, 4 位的数字确定了, 0, 5, 6 上的数字就唯一确定了, 所以只有 16 种可能的选择.

这实际上就是著名的汉明码 (Hamming code), 用 7 位数字编码 4 位信息, 并且能够纠正 1 位错误. 感兴趣读者可以查阅相关资料了解更多细节.

最后我们分析这种方案如何降低小甲的存储成本. 我们先定义加法:

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 1 + 1 = 0, \tag{22}$$

以及乘法

$$0 \times a = 0, \quad 1 \times a = a \tag{23}$$

对任意 $a = 0$ 或 1. 这样得到的集合 $\{0, 1\}$ 连同加法和乘法构成了一个域, 我们称之为 $\mathbb{F}_2$. 我们还可以对一组数字定义加法和数乘如下:

$$(x_0, ..., x_6) + (y_0, ..., y_6) = (x_0 + y_0, ..., x_6 + y_6), \quad c(x_0, ..., x_6) = (cx_0, ..., cx_6). \tag{24}$$

而三角形给出的规则是:

$$\begin{cases} x_0 + x_4 + x_5 + x_6 = 0, \\ x_1 + x_3 + x_5 + x_6 = 0, \\ x_2 + x_3 + x_4 + x_6 = 0. \end{cases} \tag{25}$$

不难发现, 这个规则是线性的! 即

- 如果 $x = (x_1, ..., x_7)$ 满足规则, 那么 $cx = (cx_1, ..., cx_7)$ 也满足规则, 其中 c 是 $\mathbb{F}_2$ 中的任意元素;
- 如果 $x = (x_1, ..., x_7)$ 和 $y = (y_1, ..., y_7)$ 都满足规则, 那么 $x + y = (x_1 + y_1, ..., x_7 + y_7)$ 也满足规则.

另一件几乎显然的事情是, 小甲希望发送的 4 位信息也可以看作是线性的. 采取这种线性的视角, 我们可以将小甲要发送的信息表示为

$$(a_1, a_2, a_3, a_4) = a_1(1, 0, 0, 0) + a_2(0, 1, 0, 0) + a_3(0, 0, 1, 0) + a_4(0, 0, 0, 1). \tag{26}$$

这样我们就把小甲要发送的 4 位数字的信息写成了 4 个基础信息的 “线性组合”. 我们可以把这 4 个基础信息对应的 7 位数字写出来:

$$\begin{aligned}(1,0,0,0) &\mapsto (1,1,0,0,0,1,0) = e_1,\\ (0,1,0,0) &\mapsto (0,0,1,0,0,1,1) = e_2,\\ (0,0,1,0) &\mapsto (1,0,0,1,0,0,1) = e_3,\\ (0,0,0,1) &\mapsto (1,0,0,0,1,1,1) = e_4.\end{aligned} \tag{27}$$

由线性性质知, 如果小甲希望发送的信息是 (a_1, a_2, a_3, a_4), 那么他可以发送

$$e = a_1e_1 + a_2e_2 + a_3e_3 + a_4e_4. \tag{28}$$

这是因为, 首先, e 的 $1, 2, 3, 4$ 位分别是 a_1, a_2, a_3, a_4; 其次, 由于 e_1, e_2, e_3, e_4 都满足规则, 由规则的线性性质知 e 也满足规则. 这样一来, 小甲就不必存储整张表格了, 他只需要存储 e_1, e_2, e_3, e_4, 共 28 位数字. 这样小甲的存储成本就大大降低了.

再考虑接收的问题. 如果他们使用表 1 中的传输方案, 小乙收到的数字是 0010101, 小乙如何知道小甲想发送什么数字? 他首先查表, 发现表中没有这个数, 说明有一位数字被翻转了. 他尝试将每一位数字翻转, 看看能否得到表中存在的数字. 读者可以尝试这个过程体会小乙的痛苦. 最终他会发现小甲希望发送的数字是 1011.

如果采用的是三角形的传输方案呢? 此时小乙可以把收到的数字放在三角形上, 计算有哪些四边形不满足条件, 然后翻转一位数字, 使得所有四边形都满足条件. 这个过程非常容易: 如果只有一个四边形不满足, 翻转它对应的顶点即可; 如果有两个四边形不满足, 翻转它们对应的三角形边中点即可; 如果三个四边形都不满足, 翻转面心 6 即可. 例如小乙收到的信息是 0010101. 计算可知, 下面两个四边形不满足条件, 因此需要翻转 3 号位, 得到 0011101. 再读 1-4 号位, 即可得到小甲希望发送的数字是 0111.

从这个例子中, 我们看到, 线性性质帮助我们极大提升了小甲的存储效率. 相比存储所有可能的信息, 小甲只需要存储一些基础信息, 通过线性组合就能生成所有可能的信息. 一般地, 如果小甲希望发送 N 位数字, 他只需要存储 N 个基础信息, 共 $M \times N$ 位数字, 其中 M 是小甲选择的字符串的长度. 比较存整个表需要的 $2^N \times M$ 位数字, 这无疑是一个巨大的提升. 在我们开始正式地研究抽象的线性性质后, 我们将看到, 这实际上是线性空间基和维数的应用.

线性性质同样让解读变得容易. 没有发生错误时, 小乙只需要读出协议中指定的若干位即可, 不需要查表. 发生错误时的情况更为复杂, 我们这里有很漂亮的结构帮助小乙纠错, 但是可以证明, 一般来说, 即使使用了线性性质, 纠错也是一个很困难的问题. 感兴趣的读者可以参考有关线性码解码复杂度问题、最小格矢问题的文献.

3. 结语

在这篇引论中, 我们通过两个例子, 展示了线性性质的强大力量. 线性性质让我们能够将复杂的问题转化为简单的问题, 也让我们能够通过存储少量的基础信息来生成大量的信息. 更值得注意的是, 线性性质可以出现在截然不同的场景中, 且都发挥了巨大的作用. 这表明, 在抽象层面研究线性性质本身, 就是一件非常有意义的事情. 在后续的学习中, 我们将一边研究抽象的线性性质, 一边通过各种例子介绍它如何出现在不同的数学和应用场景中. 读者将获得对线性性质的深刻理解, 也将初步体会到现代代数方法中抽象化、统一化思想的魅力与威力.