



计算方法

刘景铖

计算机软件新技术国家重点实验室
南京大学



回顾

上节课:

- 最小二乘法: 线性方程组无解
- 正交系统中的最小二乘法



正交系统中的最小二乘法：离散三角级数

给定数据点 $\{x_j, y_j\}_{j=0}^{2m-1}$ ，其中 $x_j = -\pi + \frac{j}{m}\pi, j = 0, 1, \dots, 2m-1$

能否找到系数 $\{a_i, b_i\}$ 使得, $\forall j$

$$y_j \approx \frac{a_0}{2} + a_n \cos nx_j + \sum_{k=1}^{n-1} (a_k \cos kx_j + b_k \sin kx_j)$$

使用最小二乘法建模：

- 思考：这里的变量是什么，线性关系是什么？
- 记误差向量为

$$E(x_j) := f(x_j) - \left(\frac{a_0}{2} + a_n \cos nx_j + \sum_{k=1}^{n-1} (a_k \cos kx_j + b_k \sin kx_j) \right)$$

- 考虑最小化 $\|E(x)\|_2^2 = \sum_{j=0}^{2m-1} E(x_j)^2$
- 这里的离散三角级数也有正交性
- 因此也可以证明

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j$$
$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j$$



离散三角级数的正交性

设 $\phi_0 = 1$,

$$\phi_k(x) = \cos(kx), k = 1, 2, \dots, n,$$

$$\phi_{n+k}(x) = \sin(kx), k = 1, 2, \dots, n-1$$

则函数族 $\phi_0(x), \phi_1(x), \dots$ 在 $[-\pi, \pi]$ 上是正交的

正交性的离散版本:

考虑对 $[-\pi, \pi]$ 进行 $2m$ 等分:

$$x_j = -\pi + \frac{j}{m}\pi, j = 0, 1, \dots, 2m-1$$

则 $\sum_{j=0}^{2m-1} \phi_k(x_j) \phi_l(x_j) = 0, \forall k \neq l$



离散三角级数的正交性

正交性的离散版本:

考虑对 $[-\pi, \pi]$ 进行 $2m$ 等分:

$$\begin{aligned}\phi_k(x) &= \cos(kx), k = 1, 2, \dots, n, \\ \phi_{n+k}(x) &= \sin(kx), k = 1, 2, \dots, n-1\end{aligned}$$

$$x_j = -\pi + \frac{j}{m}\pi, j = 0, 1, \dots, 2m-1$$

$$\text{则 } \sum_{j=0}^{2m-1} \phi_k(x_j) \phi_l(x_j) = 0, \forall k \neq l$$

主要思路:

同样使用积化和差, 例如

$$\cos(kx_j) \sin(lx_j) = \frac{1}{2} (\sin(l+k)x_j + \sin(l-k)x_j)$$

其他的乘积类似



离散三角级数的正交性

正交性的离散版本:

考虑对 $[-\pi, \pi]$ 进行 $2m$ 等分:

$$\begin{aligned}\phi_k(x) &= \cos(kx), k = 1, 2, \dots, n, \\ \phi_{n+k}(x) &= \sin(kx), k = 1, 2, \dots, n-1\end{aligned}$$

$$x_j = -\pi + \frac{j}{m}\pi, j = 0, 1, \dots, 2m-1$$

$$\text{则 } \sum_{j=0}^{2m-1} \phi_k(x_j) \phi_l(x_j) = 0, \forall k \neq l$$

我们将使用如下引理:

设整数 r 不能整除 $2m$, 则

$$\sum_{j=0}^{2m-1} \cos(rx_j) = 0, \sum_{j=0}^{2m-1} \sin(rx_j) = 0.$$

$$\text{并且, } \sum_{j=0}^{2m-1} \cos^2(rx_j) = m, \sum_{j=0}^{2m-1} \sin^2(rx_j) = m.$$

证明将会使用到复数, 而且本质上和傅里叶变换的正交性证明一样



傅里叶变换 (Fourier Transform): 多项式乘法

给定: 最多 n 次的多项式 $p(x), q(x)$ 的系数

目标: 求它们的乘积 $r(x) = p(x)q(x)$ 的系数

多项式的表示方式:

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n$$

1. 多项式的系数 $a_0, a_1, \dots, a_n$

– 计算 $r(x) = p(x)q(x)$ 需要 $O(n^2)$

– $c_k = \sum_{j=0}^k a_j b_{k-j}$

2. 多项式插值: 选定基点 $x_0, x_1, \dots, x_m$, 写下 $p(x_0), p(x_1), \dots, p(x_m)$, 则多项式 $p(x)$ 是唯一确定的

– 计算 $r(x) = p(x)q(x)$ 只需要 $O(n)$

– 令 $m \geq 2n + 1$, 计算 $r(x_0), r(x_1), \dots, r(x_m)$ 只需要写下 $p(x_0)q(x_0), p(x_1)q(x_1), \dots, p(x_m)q(x_m)$



多项式乘法

给定：最多 n 次的多项式 $p(x), q(x)$ 的系数

目标：求它们的乘积 $r(x) = p(x)q(x)$ 的系数

想法：

1. 给定 $p(x), q(x)$ 的系数
2. 令 $m \geq 2n + 1$ ，选定基点 $x_0, x_1, \dots, x_m$ ，计算 $p(x_0), \dots, p(x_m)$ 和 $q(x_0), \dots, q(x_m)$
3. 计算 $r(x_j) = p(x_j)q(x_j)$
4. 对 $\{r(x_j)\}$ 插值，得到 $r(x) = p(x)q(x)$ 的系数

愿望：

- 第2步只需要 $O(n \log n)$
- 第4步也只需要 $O(n \log n)$
- 这样总的步骤就只需要 $O(n \log n)$



复数 (Complex number)

$$z = a + b i = r e^{i \theta}, i = \sqrt{-1}$$

单位根

$$n\text{次单位根} = \{z \in \mathbb{C} : z^n = 1\}$$

$$2\text{次单位根} = \{\pm 1\}$$

$$4\text{次单位根} = \{\pm 1, \pm i\}$$

在复平面上的单位圆上，等距分布的 n 个点

$$e^{\frac{2\pi l}{n} i} = \cos \frac{2\pi l}{n} + i \sin \frac{2\pi l}{n}, l = 0, 1, \dots, n-1$$

在这节课中，我们主要关心 $n = 2^k$ 次单位根

共轭复数: $\bar{z} = a - b i$



复数 (Complex number):单位根

开平方

1

2次单位根

+1

-1

4次单位根

+1

-1

+i

-i

8次单位根

+1

-1

+i

-i

$\sqrt{i}$

$-\sqrt{i}$

$\sqrt{-i}$

$-\sqrt{-i}$

- 从上往下，开平方
- 从下往上，平方
- 如果 x 是一个单位根，则 $-x$ 也是



离散傅里叶变换 (Discrete Fourier Transform)

DFT

- 输入：多项式的系数 $a_0, a_1, \dots, a_n$

$$p(x) = \sum_{j=0}^n a_j x^j$$

- 输出：选定 $x_0, x_1, \dots, x_m$ 为 $m+1$ 次单位根，计算 $p(x_0), \dots, p(x_m)$

例子：计算 $[1, 1, 1, 1]$ 的 DFT.

- $p(x) = 1 + x + x^2 + x^3$
- 4次单位根 $= \{\pm 1, \pm i\}$
- $p(1) = 4$
- $p(x) = 0$ otherwise

注意：书上的定义中还要除上 $\sqrt{n+1}$



离散傅里叶变换 (Discrete Fourier Transform)

DFT

- 输入：多项式的系数 $a_0, a_1, \dots, a_n$

$$p(x) = \sum_{i=0}^n a_i x^i$$

- 输出：选定 $x_0, x_1, \dots, x_m$ 为 $m+1$ 次单位根，计算 $p(x_0), \dots, p(x_m)$

逆变换（插值）

- 输入：给定 $x_0, x_1, \dots, x_m$ 为 $m+1$ 次单位根，以及 $p(x_0), \dots, p(x_m)$
- 输出：多项式的系数 $a_0, a_1, \dots, a_n$



快速傅里叶变换 (Fast Fourier Transform)

FFT: 分治

输入: n 次多项式 $p(x) = \sum_{i=0}^n a_i x^i$

输出: $p(x)$ 在 $n+1$ 次单位根上的值

计算 $(n+1)/2$ 次多项式在
 $(n+1)/2$ 次单位根上的值

计算 $(n+1)/2$ 次多项式在
 $(n+1)/2$ 次单位根上的值

希望只需要 $O(n)$ 次操作合并



快速傅里叶变换 (Fast Fourier Transform)

输入: n 次多项式 $p(x) = \sum_{i=0}^n a_i x^i$

输出: $p(x)$ 在 $n+1$ 次单位根上的值

FFT: 分治

$$E(z) = a_0 + a_2 z + a_4 z^2 + a_6 z^3 + \dots$$

$$O(z) = a_1 + a_3 z + a_5 z^2 + a_7 z^3 + \dots$$

观察:

$$p(z) = E(z^2) + z O(z^2)$$

$$p(\{n+1\text{次单位根}\}) \leftarrow E(\{n+1\text{次单位根}\}^2), O(\{n+1\text{次单位根}\}^2)$$

$$\{n+1\text{次单位根}\}^2 = \{(n+1)/2\text{次单位根}\}$$

$$\text{特别地 } p(-z) = E(z^2) - z O(z^2)$$

$$E(z), O(z) \text{ 的次数为 } (n+1)/2 \text{ 次}$$

$$\text{FFT}(a_0, a_1, \dots, a_n) \leftarrow \text{FFT}(a_0, a_2, \dots), \text{FFT}(a_1, a_3, \dots)$$



快速傅里叶变换 (Fast Fourier Transform)

Figure 2.7 The fast Fourier transform (polynomial formulation)

function FFT(A, ω)

Input: Coefficient representation of a polynomial $A(x)$
of degree $\leq n-1$, where n is a power of 2
 ω , an n th root of unity

Output: Value representation $A(\omega^0), \dots, A(\omega^{n-1})$

if $\omega = 1$: return $A(1)$

express $A(x)$ in the form $A_e(x^2) + xA_o(x^2)$

call $\text{FFT}(A_e, \omega^2)$ to evaluate A_e at even powers of ω

call $\text{FFT}(A_o, \omega^2)$ to evaluate A_o at even powers of ω

for $j = 0$ to $n-1$:

 compute $A(\omega^j) = A_e(\omega^{2j}) + \omega^j A_o(\omega^{2j})$

return $A(\omega^0), \dots, A(\omega^{n-1})$

Source: Algorithms by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani



快速傅里叶变换 (Fast Fourier Transform)

- 记 $T(n)$ 为 n 阶FFT需要的操作次数

- $T(n) = 2T\left(\frac{n}{2}\right) + Cn$

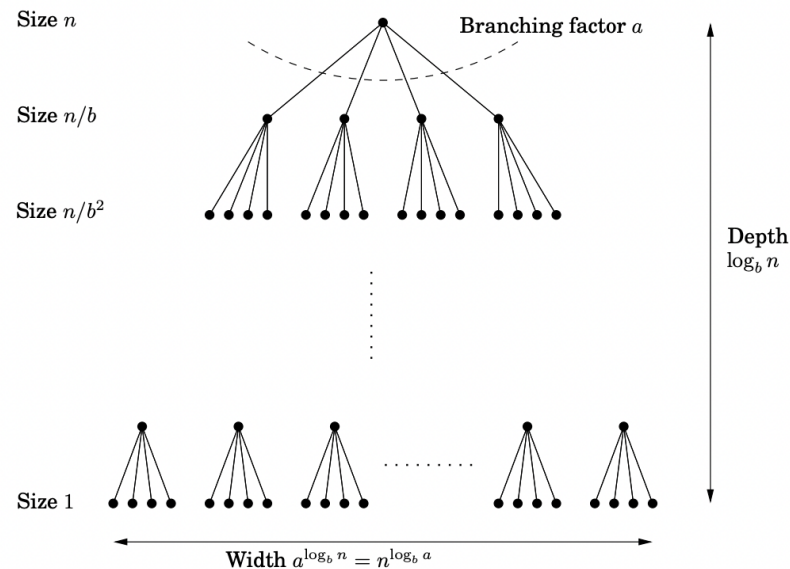
- $T(n) = 4T\left(\frac{n}{4}\right) + 2Cn = \dots$

- 设 $n = 2^k$,

- $T(n) = 2^k T\left(\frac{n}{2^k}\right) + kCn$

- $T(n) = O(n \log n)$

Figure 2.3 Each problem of size n is divided into a subproblems of size n/b .



then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a. \end{cases}$$



快速傅里叶逆变换 (Inverse FFT)

逆变换（插值）

- 输入：给定 $x_0, x_1, \dots, x_m$ 为 $m+1$ 次单位根，以及 $p(x_0), \dots, p(x_m)$
- 输出：多项式的系数 $a_0, a_1, \dots, a_n$

考虑 $n+1$ 次单位根： $\omega = e^{\frac{2\pi}{n+1}i} = \cos \frac{2\pi}{n+1} + i \sin \frac{2\pi}{n+1}$

DFT: $p(\omega^l) = \sum_{j=0}^n a_j \omega^{lj}$

FFT: 快速计算 DFT 的算法

$$p(x) = \sum_{j=0}^n a_j x^j$$

Inverse FFT: $a_l = \frac{1}{n+1} \sum_{j=0}^n p(\omega^j) \omega^{-lj}$

也是一个 DFT, 唯一区别是把 $\omega \rightarrow \omega^{-1}$, 并且前面多了系数 $\frac{1}{n+1}$



傅里叶变换的矩阵表达形式

给定 $\{a_j\}$, 求 $\{p(\omega^l)\}$

$$p(\omega^l) = \sum_{j=0}^n a_j \omega^{lj}$$

$$p(x) = \sum_{j=0}^n a_j x^j$$

$$\begin{pmatrix} \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \cdots & \omega^n \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^n & \cdots & \omega^{n^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} p(\omega^0) \\ p(\omega^1) \\ \vdots \\ p(\omega^n) \end{pmatrix}$$

注意：对复数向量的内积，通常需要取共轭复数后再做内积

单位根的共轭复数 $\omega \rightarrow \omega^{-1}$ ，这对应于Hermitian

对复数矩阵 A ，“正交”的正确定义是指Unitary: $A^H A = I$



傅里叶逆变换的矩阵表达形式

给定 $\{p(\omega^l)\}$ ，求 $\{a_j\}$

$$a_l = \frac{1}{n+1} \sum_{j=0}^n p(\omega^j) \omega^{-lj}$$

$$p(x) = \sum_{j=0}^n a_j x^j$$

$$\frac{1}{n+1} \begin{pmatrix} \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^{-1} & \cdots & \omega^{-n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{-n} & \cdots & \omega^{-n^2} \end{pmatrix} \begin{pmatrix} p(\omega^0) \\ p(\omega^1) \\ \vdots \\ p(\omega^n) \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

注意：对复数向量的内积，通常需要取共轭复数后再做内积

单位根的共轭复数 $\omega \rightarrow \omega^{-1}$ ，这对应于Hermitian

对复数矩阵 A ，“正交”的正确定义是指Unitary: $A^H A = I$



傅里叶变换的矩阵表达形式

为什么这是一个逆变换？

$$\begin{pmatrix} \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^{-1} & \cdots & \omega^{-n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{-n} & \cdots & \omega^{-n^2} \end{pmatrix} \begin{pmatrix} \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \cdots & \omega^n \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^n & \cdots & \omega^{n^2} \end{pmatrix} = (n+1)I$$

考虑乘积的 (i, j) 位置上的元素:

$$\sum_{k=0}^n \omega^{-ik} \omega^{kj} = \sum_{k=0}^n \omega^{k(j-i)} = \begin{cases} \frac{1 - \omega^{(n+1)(j-i)}}{1 - \omega^{j-i}}, & j \neq i \\ n+1, & j = i \end{cases}$$

注意：对复数向量的内积，通常需要取共轭复数后再做内积

单位根的共轭复数 $\omega \rightarrow \omega^{-1}$ ，这对应于Hermitian

对复数矩阵 A ，“正交”的正确定义是指Unitary: $A^H A = I$



回到离散三角级数的正交性

设整数 r 不能整除 $2m$, 则

$$\sum_{j=0}^{2m-1} \cos(rx_j) = 0, \sum_{j=0}^{2m-1} \sin(rx_j) = 0.$$

$$\text{并且, } \sum_{j=0}^{2m-1} \cos^2(rx_j) = m, \sum_{j=0}^{2m-1} \sin^2(rx_j) = m.$$

证明: 设整数 r 不能整除 $2m$

$$\begin{aligned} x_j &= -\pi + \frac{j}{m} \pi, j = 0, 1, \dots, 2m-1 \\ e^{iz} &= \cos z + i \sin z \\ \sum_{j=0}^{2m-1} \cos(rx_j) + i \sum_{j=0}^{2m-1} \sin(rx_j) &= \sum_{j=0}^{2m-1} e^{i r x_j} \\ &= e^{-i r \pi} \sum_{j=0}^{2m-1} e^{i r j \pi / m} = e^{-i r \pi} \frac{1 - e^{i 2 \pi r}}{1 - e^{i r \pi / m}} = 0 \end{aligned}$$

实部和虚部都必须为零!

值得注意的是, $e^{i r x_j} = e^{-i r \pi} \omega^{rj}$, 其中 ω 为 $2m$ 次单位根之一



傅里叶变换与离散三角插值的对比（选讲）

离散三角插值：给定数据点 $\{x_j, y_j\}_{j=0}^{2m-1}$ ，其中 $x_j = -\pi + \frac{j}{m}\pi, j = 0, 1, \dots, 2m-1$

能否找到系数 $\{a_i, b_i\}$ 使得, $\forall j$

$$y_j \approx \frac{a_0}{2} + a_n \cos nx_j + \sum_{k=1}^{n-1} (a_k \cos kx_j + b_k \sin kx_j)$$

最小二乘法解得

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j, \quad b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j$$

离散傅里叶变换相当于直接找到复平面上的系数 $\{c_k\}$ 使得, $\forall j$

$$y_j \approx \frac{1}{m} \sum_{k=0}^{2m-1} c_k e^{ikx_j}$$

由逆变换可得 $c_k = \sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}}$ ，进而由Euler公式可得

$$a_k + ib_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j (\cos kx_j + i \sin kx_j) = \frac{(-1)^k}{m} c_k.$$

傅里叶变换的unitary性质 $\leftrightarrow$ 三角级数的正交性



快速傅里叶变换的实现与应用（选讲）

- 参考资料：TAOCP II 4.3.3.C, by Donald E. Knuth
- 通过FFT来实现整数乘法只需要 $O(\log n)$ 位的精度
- 肉眼不可见的电子水印 digital watermarking



求解线性方程

给定矩阵 $A \in \mathbb{R}^{m \times n}$, 向量 $b \in \mathbb{R}^m$, 求 $x \in \mathbb{R}^n$ 使得 $Ax = b$

只有三种情况

- 对任意的向量 b , 存在唯一解
- 不可解, 或方程不一致 (inconsistent, over-determined)
- 存在无穷多组解 (under-determined)

性质: 只要存在两个不同的解, 则一定存在无穷多组解。

性质: 如果 $m > n$ 则存在 b 使得方程不可解。

性质: 如果 $m < n$ 则存在 b 使得方程存在无穷多组解。

注: 解线性方程组, 与矩阵求逆是两回事

下面我们假设 $m = n$



回顾高斯消元法

$$\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \end{array}$$

$$\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22} - \frac{a_{21}}{a_{11}}a_{12} & \dots & a_{2n} - \frac{a_{21}}{a_{11}}a_{1n} & b_2 - \frac{a_{21}}{a_{11}}b_1 \end{array}$$

可以一行一行地消，也可以一列一列地消
这里以行为例，主要涉及三种操作

- 交换行（交换两组方程）
- 给一行乘上一个数
- 在一行上减去另一行的一个倍数



回顾高斯消元法：交换行

令 $\sigma \in S_n$ 为 $[n]$ 上的一个置换

$$P_\sigma = \begin{bmatrix} \cdots & e_{\sigma(1)}^\top & \cdots \\ & \vdots & \\ \cdots & e_{\sigma(n)}^\top & \cdots \end{bmatrix}$$

其中向量 e_j 是只有在第 j 个分量上为1的单位向量



回顾高斯消元法：给一行乘上一个数

考虑对角矩阵

$$D = \begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & c_n \end{bmatrix}$$



回顾高斯消元法：在一行上减去另一行的一个倍数

在第 j 行减去第 i 行的 c 倍：

$$E = I - c e_j e_i^T$$

$e_j e_i^T$ 是一个矩阵，里面只有一个元素 (j, i) 是1

这是一个可逆的操作：

$$(I + c e_j e_i^T)(I - c e_j e_i^T) = I$$



回顾高斯消元法

$$Ax = b$$

$$E_1 Ax = E_1 b$$

$$E_2 E_1 Ax = E_2 E_1 b$$

...

$$E_k \cdots E_2 E_1 Ax = E_k \cdots E_2 E_1 b$$

如果一行一行地消，最后得到的将会是上三角阵
而且 $E_k \cdots E_2 E_1$ 是下三角阵

$$A = LU$$



回顾高斯消元法

- 期间可能需要交换行/列，以选择一个非零/更大的主元 (pivoting), 例子:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- 交换行：左乘置换矩阵
- 交换列：右乘置换矩阵
- 如果找不到非零主元，则 A 一定是奇异的(singular, non-invertible)
- 每消去一个元素，最多 $O(n)$ 次算术运算
- 最坏情况下总共需要 $O(n^3)$ 次算术运算
- BLAS (Basic Linear Algebra Subprograms)
- LAPACK