一维分片多项式空间

魏华祎

2019 年 10 月 17 日

从向量空间到函数空间

分片多项式空间

函数空间有下面两个要素决定

- 函数空间的基函数, 注意基函数的选取不是唯一的.
- 函数空间的自由度

分片线性多项式空间

给定区间 $I=[x_0,x_1]$, 其上的线性多项式空间定义如下：

$$P_1(I)=\{v:v(x)=c_0+c_1x,x\in I,c_0,c_1\in\mathbb{R}\}, \tag{1}$$

注解 1.

- 上面表达式选取的基函数为单项式 $(1,x)$.
- 函数空间的自由度个数就是基函数的个数.
- 函数空间自由度的值 (c_0,c_1), 就是相对于基函数的坐标.
- 给定基函数, 则区间 I 上所有的线性函数和 $\mathbb{R}^2$ 空间中的点建立了一一对应关系.

另一种基函数的选择是重心坐标函数 (节点基函数):

$$\lambda_0(x)=\frac{x_1-x}{x_1-x_0},\lambda_1(x)=\frac{x-x_0}{x_1-x_0}. \tag{2}$$

注解 2.

- 性质
- 代数意义
- 几何意义
- 选择重心坐标做为基函数的好处是什么?

对于 $v \in P_1(I)$, 有

$$v(x) = v(x_0)\lambda_0 + v(x_1)\lambda_1$$

Sylvester 插值

Sylvester 插值多项式 [2] 是构造高次基函数的基础, 它的定义如下

$$R_i(k,\lambda) = \begin{cases} \frac{1}{i!}\prod_{l=0}^{i-1}(k\lambda - l), & 1 \le i \le k \\ 1, & i = 0 \end{cases}$$

其中 $\lambda \in [0,1]$, k 表示区间 $[0,1]$ 的等分数。易知

- $R_i(k,\lambda)$ 是关于 λ 的 i 次多项式。
- 当 $\lambda = \frac{l}{k}, l = 0, 1, \cdots, i-1$ 时, $R_i(k,\lambda) = 0$。
- 当 $\lambda = \frac{i}{k}$ 时 $R_i(k,\lambda) = 1$。

k 次多项式空间 $P_k(I)$

给定区间 $I = [x_0, x_1]$, 其上的线性多项式空间定义如下：

$$P_k(I) = \{v : v(x) = c_0 + c_1 x + \cdots + c_k x^k, x \in I, c_0, c_1, \cdots c_k \in \mathbb{R}\}, \tag{3}$$

区间 $[x_0, x_1]$ 上的个 $k \ge 1$ 次基函数共有

$$n_{dof} = k + 1,$$

其计算公式如下:

$$\phi_{m,n} = \frac{1}{m!n!}\prod_{l_0=0}^{m-1}(k\lambda_0 - l_0)\prod_{l_1=0}^{n-1}(k\lambda_1 - l_1).$$

其中 $m \ge 0$, $n \ge 0$, 且 $m + n = k$, 这里规定:

$$\prod_{l_i=0}^{-1}(k\lambda_i - l_i) := 1, \quad i = 0, 1$$

k 次基函数的面向数组的计算构造向量：

$$P = (\frac{1}{0!}, \frac{1}{1!}, \frac{1}{2!}, \cdots, \frac{1}{k!})$$

构造矩阵：

$$A := \begin{pmatrix} 1 & 1 \\ k\lambda_0 & k\lambda_1 \\ k\lambda_0 - 1 & k\lambda_1 - 1 \\ \vdots & \vdots \\ k\lambda_0 - (k-1) & k\lambda_1 - (k-1) \end{pmatrix}$$

对 A 的每一列做累乘运算, 并左乘由 P 形成的对角矩阵, 得矩阵:

$$B = \mathsf{diag}(P) \begin{pmatrix} 1 & 1 \\ \lambda_0 & \lambda_1 \\ \prod_{l=0}^{1}(k\lambda_0 - l) & \prod_{l=0}^{1}(k\lambda_1 - l) \\ \vdots & \vdots \\ \prod_{l=0}^{k-1}(k\lambda_0 - l) & \prod_{l=0}^{k-1}(k\lambda_1 - l) \end{pmatrix}$$

易知, 只需从 B 的每一列中各选择一项相乘 (要求二项次数之和为 k, 其中取法共有

$$n_{dof} = k + 1$$

构造指标矩阵：

$$I = \begin{pmatrix} k & 0 \\ k-1 & 1 \\ \vdots & \vdots \\ 0 & k \end{pmatrix}$$

则第 i 个 k 次基函数可写成如下形式

$$\phi_i = B_{m,0} B_{n,1},$$

其中 $m = I_{i,0}, n = I_{i,1}$, 并且 $m + n = k$.

$$\begin{aligned} \nabla \prod_{j=0}^{m-1}(k\lambda_0 - j) =& k \sum_{j=0}^{m-1} \prod_{0 \le l \le m-1, l \ne j} (k\lambda_0 - l) \nabla\lambda_0, \\ \nabla \prod_{j=0}^{n-1}(k\lambda_1 - j) =& k \sum_{j=0}^{n-1} \prod_{0 \le l \le m-1, l \ne j} (k\lambda_1 - l) \nabla\lambda_0, \end{aligned}$$

$$\boldsymbol{D}^0 = \begin{pmatrix} k & k\lambda_0 & \cdots & k\lambda_0 \\ k\lambda_0 - 1 & k & \cdots & k\lambda_0 - 1 \\ \vdots & \vdots & \ddots & \vdots \\ k\lambda_0 - (k-1) & k\lambda_0 - (k-1) & \cdots & k \end{pmatrix},$$

$$\boldsymbol{D}^1 = \begin{pmatrix} k & k\lambda_1 & \cdots & k\lambda_1 \\ k\lambda_1 - 1 & k & \cdots & k\lambda_1 - 1 \\ \vdots & \vdots & \ddots & \vdots \\ k\lambda_1 - (k-1) & k\lambda_1 - (k-1) & \cdots & k \end{pmatrix},$$

把 $\boldsymbol{D}^0$ 和 $\boldsymbol{D}^1$ 的每一列沿行的方向做累乘运算, 然后取它们的下三角矩阵, 最后把下三角矩阵的每一行再求和, 即可得到矩阵 $\boldsymbol{B}$ 的每一列各个元素的求导后系数. 可得到矩阵 $\boldsymbol{D}$, 其元素定义为

$$\boldsymbol{D}_{i,j} = \sum_{m=0}^{j} \prod_{k=0}^{j} D_{k,m}^{i}, \quad 0 \le i \le 1, , 0 \le j \le k-1.$$

最后,可以用如下的方式来计算 $\boldsymbol{B}$ 的梯度：

$$\begin{aligned}\nabla \boldsymbol{B} =& \mathrm{diag}(\boldsymbol{P}) \begin{pmatrix} 0 & 0 \\ \boldsymbol{D}_{0,0}\nabla\lambda_0 & \boldsymbol{D}_{1,0}\nabla\lambda_1 \\ \vdots & \vdots \\ \boldsymbol{D}_{0,k-1}\nabla\lambda_0 & \boldsymbol{D}_{1,k-1}\nabla\lambda_1 \end{pmatrix} \\ =& \mathrm{diag}(\boldsymbol{P}) \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \begin{pmatrix} \nabla\lambda_0 & \\ & \nabla\lambda_1 \end{pmatrix} \\ =& \boldsymbol{F} \begin{pmatrix} \nabla\lambda_0 & \\ & \nabla\lambda_1 \end{pmatrix},\end{aligned}$$

其中

$$\boldsymbol{F} = \mathrm{diag}(\boldsymbol{P}) \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix}. \tag{4}$$

插值

线性插值

给定区间 $I = [x_0, x_1]$, 和一个定义在 I 上的连续函数 f, 其插值定义如下：

$$\pi_1 f(x) = f(x_0)\lambda_0 + f(x_1)\lambda_1$$

注解 3.

- *如果 $f \in P_1(I)$ 则 $f = \pi_1 f$.*
- *一般情况下, $f \neq \pi_1 f$.*
- *计算插值只需要计算函数在网格点处的值, 很容易实现.*
- *两者之间会差多远？$\|f - \pi_1 f\|_0$.*

注解 4. *用范数衡量函数和函数插值之间的误差*

- *无穷范数*
$$\|v\|_\infty = \max_{x\in I} |v(x)|$$
- *L^2 范数*
$$\|v\|_2 = \left(\int_I v^2 \,\mathrm{d}x\right)^{\frac{1}{2}}$$

注解 5. *误差分析中的常用的两个不等式*

- *三角不等式*

$$\|v+w\|_2 \le \|v\|_2 + \|w\|_2$$

- *Cauchy-Schwarz 不等式*

$$\int_I vw \,\mathrm{d}x \le \|v\|_2 \|w\|_2$$

命题 1. *插值误差满足下面估计*

$$\|f-\pi_1 f\|_2 \le Ch^2\|f''\|_2$$
$$\|(f-\pi_1 f)\|_2 \le Ch\|f''\|_2$$

Proof. 利用微积分基本定理, 把函数和函数的导数联系起来.
利用罗尔中值定理和微积基本定理把, 函数的 1 阶导数和 2 阶导数联系起来. □

连续线性分片插值

给定区间 $I=[a,b]$, 分成 n 段, 共有

$$a = x_0 < x_1 < \cdots < x_n = b$$

可以定义连续函数 f 在区间 I 上的连续分片线性插值函数为

$$\pi_1 f = \sum_{i=0}^{n} f(x_i)\phi_i(x)$$

命题 2. *连续分片线性插值误差满足下面估计*

$$\|f-\pi_1 f\|_{2,I}^2 \le C\sum_{i=0}^{n-1} h_i^4 \|f''\|_{2,I_i}^2$$
$$\|(f-\pi_1 f)\|_{2,I}^2 \le C\sum_{i=0}^{n-1} h_i^2 \|f''\|_{2,I_i}^2$$

L^2 投影

给定 $f \in L^2(I)$, 定义 L^2 投影 $P_h f \in V_h$,

$$(P_h f, v)_I = (f,v)_I, \forall v \in V_h$$
$$(f-P_h f, v)_I = 0, \forall v \in V_h$$

注解 6.

- $f - P_h f \perp V_h$.
- $P_h f$ 和 $\pi_1 f$ 是不同的.
- 推导 P_h 对应的线性系统.
- 计算 $P_h f$ 需要求解线性代数系统, 计算量比插值要大.

定理 4.1.

$$\|f - P_h f\|_{2,I} \leq \|f - v\|_{2,I}, \forall v \in V_h.$$

定理 4.2.

$$\|f - P_h f\|_{2,I}^2 \leq C \sum_{i=0}^{n-1} h_i^4 \|f''\|_{2,I_i} \leq Ch \|f''\|_{2,I}^2$$

Proof. 利用投影的正交性质, 最优最近性及插值的误差估计. □

数值积分

所有的数值积分公式本质上都是给一组积分点 $(x_0, x_1, \cdots, x_{m-1})$ 及相应的权重 $(w_0, w_1, \cdots, w_{m-1})$.

$$\int_I f(x)\,\mathrm{d}x \approx |I|(w_0 f(x_0) + w_1 f(x_1) + \cdots + w_{m-1} f(x_{m-1})) = |I| \sum_{i=0}^{m-1} w_i f(x_i)$$

注解 7.

- $\sum_{i=0}^{m-1} w_i = 1$.
- $|I|$ 是区间的长度.
- 数值积分的理论基础是积分中值定理.
- 上面数值积分公式形式同样适用于高维区域上的积分.

设 m 是区间 $I = [x_0, x_1]$ 的中点, 下面给出几种常见的积分公式。

中点积分公式:

$$|I| f(m)$$

梯形积分公式:

$$|I| \left[\frac{1}{2} f(x_0) + \frac{1}{2} f(x_1)\right]$$

辛普森积分公式:

$$|I| \left[\frac{1}{6} f(x_0) + \frac{4}{6} f(m) + \frac{1}{6} f(x_1)\right]$$

实现

Numpy 多维数组

利用 Python 数据分析的课件讲解。

爱因斯坦求和

einsum 函数是 NumPy 的中最有用的函数之一。由于其强大的表现力和智能循环，它在速度和内存效率方面通常可以超越我们常见的 array 函数。但缺点是，可能需要一段时间才能理解符号，有时需要尝试才能将其正确的应用于棘手的问题。

Python code 1 利用 einsum 求和进行向量和矩阵相乘

```python
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
np.einsum('i', a) # 返回 a 本身
np.einsum('i->', a) # 对 a 求和
np.einsum('i, i->i', a, b) # a 与 b 对应元素相乘
```

Python code 2 利用 einsum 求和进行向量和矩阵相乘

```python
import numpy as np
A = np.arange(3)
B = np.arange(12).reshape(3, 4)
C = np.einsum('i, ij->i', A, B)
```

Python code 3 利用 einsum 求和进行矩阵相乘

```python
import numpy as np
A = np.array([[1, 1, 1], [2, 2, 2], [5, 5, 5]])
B = np.array([[0, 1, 0], [1, 1, 0], [1, 1, 1]])
C = np.einsum('ij, jk->ik', A, B)
```

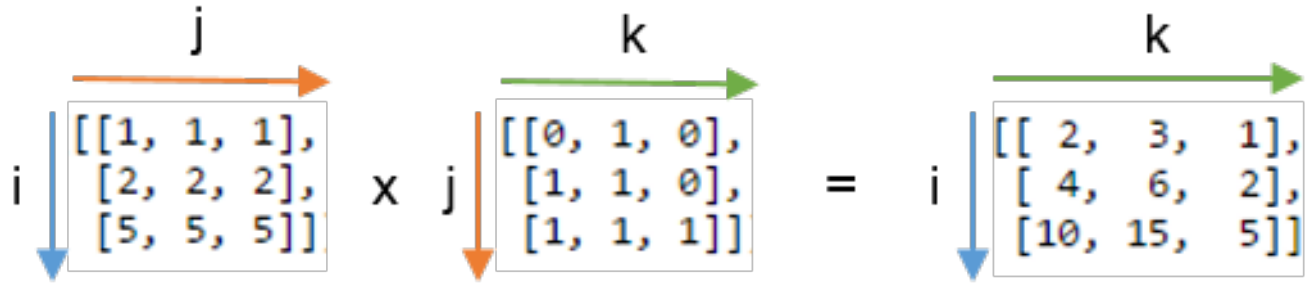

图 1: 两矩阵相乘示意图。

Python code 4 一维向量运算

```python
import numpy as np
A = np.arange(10)
B = np.arange(5, 15)
np.einsum('i->', A) # 求和
np.einsum('i, i->i', A, B) # 对应元素相乘
np.einsum('i, i->', A, B) # 内积 np.inner(A, B) 或 np.dot(A, B)
np.einsum('i, j->ij, A, B) # 外积 np.outer(A, B)
```

Python code 5 二维向量运算

```python
import numpy as np
np.einsum('ii', C) # 求迹 np.trace(C)
np.einsum('ij, ji->ij', C, D) #  C*D.T
np.einsum('ij, kl->ijkl', C, D) # C[:, :, None, None]*D
```

插值、积分与投影的 Python 实现

基于 Python 的 Numpy 实现相应的 L2 投影.

问题

[1]

参考文献

[1] F. B. Mats G. Larson. *The Finite Element Method: Theory, Implementation, and Applications*. Springer, 2010.

[2] 盛新庆. *计算电磁学要论*. 中国科学技术大学出版社, 2008.