RBF 基本原理

RBF 是一种对多维散点数据进行数值拟合的插值方法.
给定一组包含n条数据的输入
$\mathbf{x} \in \mathbb{R}^m, \quad \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$
RBF 对这组输入求得一个插值函数
$y = f(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi\left(\|\mathbf{x} - \mathbf{x}_i\|\right),$
其中$\phi(x)$是手动指定的RBF kernel,我们选用的是
$thin\_plate : \phi(x) = x^2 log(x).$

基于这个插值得到的函数,
对每一个新的m维变量$\mathbf{x}$都能求得一个对应的$y = f(\mathbf{x})$

可以看到插值函数$y = f(\mathbf{x})$由已知输入$\{\mathbf{x}_i\}_{i=1,\dots,n}$和
一组权重$\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix}^T \in \mathbb{R}^n$构成
求得了这组权重我们就求得了这个插值函数.

RBF的构造过程就是通过给定输入$\mathbf{x} \in \mathbb{R}^m, \quad \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$求权重的过程

记$\mathbf{\Phi} = \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,n} \\ \vdots & \ddots & \vdots \\ \phi_{n,1} & \cdots & \phi_{n,n} \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad \phi_{i,j} = \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right)$
通过解$\mathbf{\Phi w} = \mathbf{y}$这样一个方程我们就可以求得$\mathbf{w}$
考虑Smoothing/Regularization,这个方程被写为$(\mathbf{\Phi} + \lambda \mathbf{I})\mathbf{w} = \mathbf{y}$

基于LU分解,我们可以解这个方程求得权重$\mathbf{w}$

RBF 算法伪代码

给定一组输入$X = (x_{ij}) \in \mathbb{R}^{m \times n}, \quad \mathbf{y} = (y_j) \in \mathbb{R}^{n \times 1}$
其中X表示由n组m维向量组成的$m \times n$矩阵,$\mathbf{y}$表示n组向量对应的n个y
即我们有n组输入$(\mathbf{x}, y)$,每个$\mathbf{x}$由m个分量x_i构成

RBF (插值)算法伪代码如下:

1. 对输入矩阵X的n组向量两两之间求$L2$范数欧式距离
   得到一个矩阵$D = (d_{ij}) \in \mathbb{R}^{n \times n}$ where $d_{ij} = \|X_{*,i}, X_{*,j}\|$
2. 对矩阵D的每个元素$r = d_{ij}$求其对应的$\phi(d_{ij}) = r^2 log(r)$得到矩阵
   $A = (\phi(d_{ij})) \in \mathbb{R}^{n \times n}$
3. 考虑正则化,矩阵$A = A + \lambda \mathbf{I}$
4. 解线性方程组$A\mathbf{w} = \mathbf{y}$求得权重向量$\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix}^T \in \mathbb{R}^n$

在python中
步骤1中求矩阵列向量两两之间欧式距离的操作 可以通过 `scipy.cdist(X,X)` 来完成
步骤2中$r^2 log(r)$操作可以通过 `scipy.xlogy` 来完成
在 eigen/xtensor 中,我们都没有现成的函数可以使用,必须手动实现这两个功能

手动实现 `scipy.cdist` 时思路如下:

$$
\begin{aligned}
&d_{ij} = \|A_{*,i}, B_{*,j}\| \\
&= \sqrt{(A_{0i} - B_{0j})^2 + (A_{1i} - B_{1j})^2 + \cdots + (A_{(m-1)i} - B_{(m-1)j})^2} \\
&= \sqrt{(A_{0i}^2 + \cdots + A_{(m-1)i}^2) + (B_{0j}^2 + \cdots + B_{(m-1)j}^2) - 2(A_{0i}B_{0j} + \cdots + A_{(m-1)i}B_{(m-1)j})}
\end{aligned}
$$

所以
$cdist(A, B) = \sqrt{norm(A, by\_col) + norm(B, by\_col) - 2dot(A^T, B)}$
即我们可以通过对矩阵A, B按列求$norm$然后求矩阵乘
就可以**批量的求得矩阵A, B列向量两两之间的欧式距离**

手动实现 `scipy.xlogy` 时思路如下: $xlogy = x \times log(y)$
公式本身并不复杂,但需要注意的是,$xlogy$的参数是两个向量之间的欧式距离,值可能为0
而$log(0) = inf$,scipy中的xlogy能自动处理这种情况
手动实现时需要我们自己手动处理输入为0的情况
在eigen中,我们这样处理

```
logr = (logr.array().isInf() == 1).select(0, logr);
```

在xtensor中,我们这样处理

```
xt::log(xt::where(input > 0, input, 1));
```

相应地,求得BBF插值函数之后,给定新输入求值过程如下:

插值函数由已知的X和$\mathbf{w}$构成:$y = \sum_{i=1}^{n} w_i \phi\left(\|\mathbf{x} - \mathbf{x}_i\|\right)$
给定一个新的输入$\mathbf{x} = (x_i) \in \mathbb{R}^{m \times 1}$,求值过程如下:

1. 求新输入向量和已知X的n个列向量之间的欧式距离得到一个向量
   $D = (d_{ij}) \in \mathbb{R}^{1 \times n}$

2. 对矩阵D的每个元素$r = d_{ij}$求其对应的$\phi(d_{ij}) = r^2 log(r)$得到矩阵 $A = (\phi(d_{ij})) \in \mathbb{R}^{1\times n}$
3. $dot(D, \mathbf{w})$求得y

关于怎样解线性方程组$Aw = y$的讨论

fsdf

参考链接

一份对RBF求解过程清晰直观的介绍 https://yuki-koyama.github.io/mathtoolbox/rbf-interpolation/

RBF 理论详解 Ken Anjyo, J. P. Lewis, and Frédéric Pighin. 2014. Scattered data interpolation for computer graphics. In ACM SIGGRAPH 2014 Courses (SIGGRAPH '14). ACM, New York, NY, USA, Article 27, 69 pages. DOI: https://doi.org/10.1145/2614028.2615425 http://scribblethink.org/Courses/ScatteredInterpolation/scatteredinterpcoursenotes.pdf