**高斯过程回归方法的实现与简单应用**

**The Implement and Simple Application of Gaussian Process Regression**

电子工程系 无43

丁文浩 （2014011079）

dwh14@mails.tsinghua.edu.cn

**Abstract**

本文所探讨的主要对象是一种不基于模型参数的回归方法，该方法称为高斯过程回归（Gaussian Process Regression，GPR），这是一种基于高斯过程的协方差矩阵生成的后验概率判断模型。高斯过程回归的主要问题在于核函数、均值函数和似然函数的选择以及大量训练集数据的近似计算问题，所以本文分别对于这两个方面进行了研究，最终得到较快的计算速度和准确度较高的均方误差（Mean Square Error）计算结果。

本文中的实验所采用的数据集有两部分，第一部分是一维时间序列数据集（time-serial），针对这一数据集我首先采用了人工选择核函数模型的方式，然后通过自己编写的训练过程函数以及均值判断函数来进行测试，最终的MSE能够达到0.4左右；在此基础上应用[核函数空间搜索的方式 **[1]**](#refer1)，并采用GPR的[gpml **[2]**](#refer2)工具包实现了基于最大似然函数的自动生成核函数算法，最终的MSE计算结果能够达到0.1左右。第二部分是对于第二个非时间序列多维数据集，直接采用[gpml **[2]**](#refer2) 工具包进行多维数据的训练，首先分析各位数据之间的相关性，通过手动选择较为合理的核函数能够达到0.02左右的MSE；在此基础上针对大数据量的训练集加速问题进行了BCM（Bayesian Committee Machine）近似优化，能够在提高速度的前提下仍然达到MSE为0.03左右的计算误差。

最后限于时间的关系，尝试了其他的贝叶斯推断方法和似然函数表示方法，并对于得到的结果进行了简要的分析，但是并没有得出一套完整的选择理论

来进行合理的均值函数和似然函数选择，这也是将来需要完善的地方。

**1.Introduction**

高斯过程指的是一组随机变量的集合，这个集合里面的任意有限个随机变量都服从联合高斯分布。一个高斯过程完全由它的均值函数和协方差函数决定，这一性质决定了其在机器学习当中具备数据拟合的[良好性质 **[3]**](#refer3)。监督学习分为回归（regression）和分类（classification）两类，本文研究的回归问题可以用如下的方式表达：假设有训练集其中的 为d维的输入矢量， 为输出矢量中的标量，回归问题的本质是根据训练数据集建立映射关系 ，进而可以对于所有输入的数据给出预测的结果。

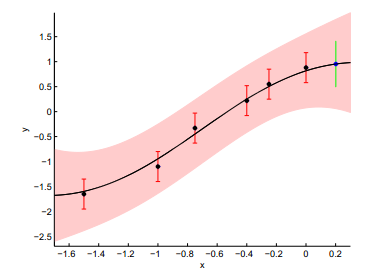


图1 高斯过程回归

在监督学习中通常有两种方法来确定这个映射函数，一类是采用参数化回归，典型的方式是[最小二乘法拟合**[4]**](#refer4)，但是这类指定参数的模型极容易产生过拟合和泛化能力较弱的问题，造成预测效果对于参数较为敏感；另外一类方法是采用极大似然判断，在概率函数空间进行贝叶斯推断，在[Rasmussen 和 C. K. I. Williams**[5]**](#refer5) 的书中对于该模型有详细的介绍。如下图所示，我们能够得到预测的最好值，以及相应的预测方差。

为了能够最好的描述相关性，核函数的选择十分重要，在[David Kristjanson Duvenaud **[6]**](#refer6) 的2014年的工作中，他们提出了一种通过贪心算法的方式自动组合核函数，他们采用的衡量模型好坏的方式是基于最大似然函数的贝叶斯信息准则（Bayesian information criterion），从而能够避免模型的复杂度过大，并且为了提高生成的核函数的可视化效果利用高斯过程的性质将核函数拆分成和的形式。

另一方面，GPR适合小型数据集，对于大量数据的计算过程花费时间较长，因为其时间复杂度为，所以需要采用一些近似求解的方法，目前的工作中有如下几种近似方法：

[Silverman **[7]**](#refer7) 提出的回归因子子集化方法（Subset of Regressors），通过随机或者贪心的方式来近似估计概率分布的形式，并且修改原来的最大似然函数表达式，该方法可以将时间复杂度降至。

[Williams and Seeger **[8]**](#refer8) 建立了一种The Nystrom方法，使用 代替公式中的 ，该算法可以实现均值预测的时间复杂度为，方差预测的时间复杂度为 。

[Lawrence et al **[9]**](#refer9) 提出可以通过一种基于熵差参数的贪心算法来有效地降低训练集的数据量，尽管这样造成了数据集的浪费，但是可以大大地提高速度，他们将这个方法称为informative vector machine（IVM），该方法的时间复杂度也能够达到 。

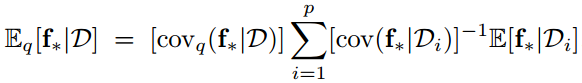
[Seeger. M **[10]**](#refer10) 在他的2003年的一项工作中提出了一种投影过程近似的方法（Projected Process Approximation），这也是一种非退化的方式，但是可以利用所有的数据集。思路是把m个数据集作为训练集，把n-m个作为条件从而得到条件概率：

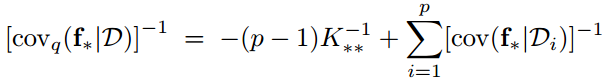
其活跃子集I的选择有两种方式，一个是贪心算法，另一个是[on-line fashion**[11]**](#refer11)。

[Tresp **[12]**](#refer12) 提出的Bayesian Committee Machine（BCM）方法可以加速高斯回归过程，主要思想是将数据集分割成，从而得到后验概率：



这里的c是一个归一化常数，从而能够得到最终的均值和方差的预测值：





对于上面几种算法主要分为两类，一类是对于后验概率分布的近似求解，另一类是对于训练数据集的采样，在[Rasmussen**[5]**](#refer5) 的书中对于上述几种方法做出了实验比较。

本文的结构安排如下：首先介绍了高斯过程回归的数学模型分析，然后给出本文的实验中采用的自动生成核函数算法和高维数据加速算法及其分析，接下来展示在两个数据集上的数值结果与方法的对比，最后对于后续可以改进的问题进行一些探讨。

**2.Model**

**2.1 高斯过程回归模型**

首先介绍一下高斯回归过程的数学模型，不同于普通的回归过程，在GPR中不需要指定映射函数 的具体形式，n个训练数据的观测值 背认为是从n维的高斯分布中采样出来的一个n维的点，类似的有 也可以认为是从高斯过程中采样得到的一个无穷维的点，通常情况下，这个f被假设给定一个高斯过程先验，并且不失一般性地假设均值为0，故有：

而观测值y通常可以表示为：

因为一般情况下观测值都含有一定的噪声，上式中的K是协方差矩阵，也就是我们需要通过训练得到的参数，而对于测量的结果可以表示为：

上式中的 就是我们的预测值，可以看到整个过程我们并不需要知道 的具体形式，而求解过程也就转变成了求得K矩阵的过程，具体而言有：

进而利用联合高斯分布的性质可以得到条件概率分布 ，且有条件分布也符合高斯分布的结论：

对于高斯分布我们就可以把均值作为最佳的估计值，故有：

综上所述，GPR的核心问题就变成了求协方差矩阵K的过程，而根据Mercer定理，我们和已使用核函数来表达协方差，那么核函数的选择方式就变成了研究的重点，本文的一个重要部分就是针对如何选择合适的核函数组合进行了探究。

**2.2 参数训练**

在选定模型之后需要进行核函数参数的拟合，最常用的方法是最大后验估计，即：

一般情况下我们对于{θ}没有任何先验的知识，所以最大后验就退化成了最大似然，通常情况下用对数表示：

训练的过程就是采用共轭梯度下降（Conjugate Gradient）的方式来求得上述最小值条件下的 {θ}。该方法是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算Hesse矩阵并求逆的缺点。在高斯似然函数的条件下即需要求得：

这里的α的值为。

**2.3 核函数选择**

通常使用的核函数都是有基本的核函数经过加或者乘操作获得的，用于高斯过程回归的基本核函数有如下几种：

**Basic kernel function list**

**常数模型**

**平方指数模型**

**周期函数模型**

**线性核函数**

**有理二次平滑模型**

**余弦函数模型**

**白噪声模型**

表 1 基本核函数

根据这些核函数的图像我们就可以通过人工的方式分析训练数据的趋势从而选择能够描述训练数据的核函数，比如训练数据存在明显的周期振荡，我们就应该在核函数里面包含PER核函数来进行描述。为了能够更加高效与合理地选择，本文将会实现一种自动核函数生成的贪心算法。

**3.Method**

**3.1 核函数的自动构建算法**

[David Kristjanson Duvenaud **[6]**](#refer6)提出了一种能够使用机器来自动生成核函数并进行最优估计的方法，他认为人工构造核函数对于复杂模型是不现实的，并且核函数都是由基本核函数生成的，所以完全可以使用一种生成机制来构造。因此他实现了一种基于贪心算法的构造方式，并称为Automatic Bayesian Covariance Discovery (下面简称为ABCD算法)。

3.1.1 构造原则解析

核函数的构造方法主要是采用加和乘的操作：

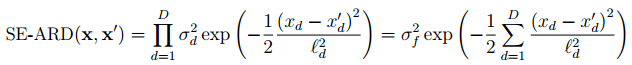
首先分析两个核函数相乘带来的影响，一般而言，两个正定的核函数相乘的结果仍然是正定的，几种典型的形式有：

（1）N个LIN函数相乘构造出多项式回归函数，且自由度为N；

（2）乘上一个SE核函数可以将全局结构转化成局部结构，比如SE\*PER可以描述局部周期性。

（3）乘上一个LIN核函数表示将整个模型进行线性的放大。

除了构造更加复杂的模型，相乘还可以构造高维的核函数，方法是对于每一个独立的输入维度都定义一个核函数，然后将各个维度的核函数相乘即可得到多输入的核函数模型，比如gpml中的SE-ARD（automatic relevance determination）函数：



以两个维度为例可以从二维输入变量生成一个二维核函数，从而能够定义二维高斯过程：

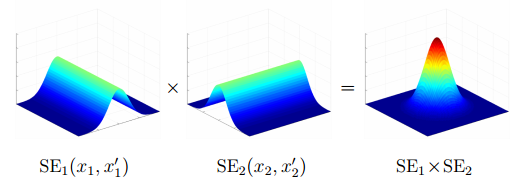


图2 核函数相乘

接下来介绍核函数相加带来的影响，根据高斯过程的性质有如下的结论：

这里的 和 代表不同的核函数，由此得到了独立分布的和的形式。最常用的加性核函数就是白噪声（WN），这是因为观测值通常是由噪声的，通过加入WN这一项我们希望能够将信号与噪声部分进行分离。

在高维情况下同样也可以进行加法的操作如下：

该过程可以表示成如下的操作：

上述二维过程的加法用图像表示为下图：

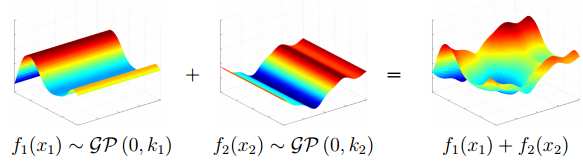


图3 核函数相加

除了加法和乘法的操作之外还有一个较为常用的操作是突变点函数（Change point），其表达式如下：



其中的σ是logistic函数：

该核函数的作用在于能够平滑连接两个不同的核函数，这在时间序列训练数据中比较常用，由于某些特殊的原因，在某一个时间点整个模型发生了改变，需要用新的核函数来表达，比如下图中的突变产生：

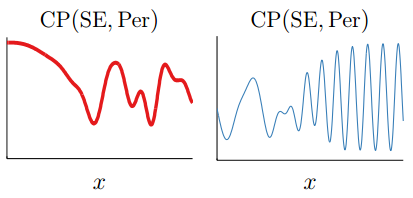


图4 突变点函数

3.1.2 ABCD算法的组成

该算法主要由四部分组成，通过这四个部分的结合可以得到一套完整的核函数生成分析工具。

**An open-ended language of models**

许多学习算法的搜索空间是有限的，比如图模型，但是对于机器来说，无限空间的搜索可能更容易得到新的高效的模型。

**A search through model space**

由基函数生成的核函数空间是一个无限维空间，所以我们需要一个能够无限迭代生成核函数的搜索过程，这个过程类似于人类通过对于模型的分析不断添加新的基函数的过程。

**A model comparison procedure**

搜索函数的重点在于如何判断核函数模型的好坏，我们采用的指标是近似的边缘似然值，并且通过Bayesian Information Criterion（BIC）方法来限制函数复杂度的增长，这一点在后面的分析中会详细介绍。

**A model description procedure**

仅仅完成核函数的构造是不够的，我们需要一种方式来分析构造出来的核函数模型为什么有效，这就需要一系列自然语言的描述方法，需要针对不同核函数的特性进行系统的分解，主要采用的方式是将核函数拆分成和的形式，这一部分在后面的分析中有详细介绍。

**3.2 大规模训练集的近似估计算法**

典型的GPR方法适合训练数据在几百个的情况，对于像第二个数据集这种10000个训练集的情况将会花费大量的时间进行计算，主要计算量在于求解一个n\*n大小的矩阵的逆，所以整个过程的时间复杂度是 ，针对这一问题，之前已经提到了多种近似推断的方法，这里我采用的方式是BCM方法，该方法的主要思想是将整个训练集划分成M各子集，在每一个子集上分别训练一个最优的估计器，最后将这M个估计器采用按权重的方法进行合并，从而得到最终的结果。

通常情况下有后验概率：

我们做出这样一种近似假设：

这个近似的成立是有一定的条件的，首先要求测试数据集要足够大，这样就可以在处处决定f了；另外一个条件是要求子集 与 的相关性要足够的小，对于这一点可以采用聚类的方式，这个在后面的算法分析中有详细介绍。

使用上式中的近似表达之后我们可以得到如下最终的整个训练集上的后验概率密度表达式：

基于这个表达式我们就可以得到在测试数据集下的近似目标均值与方差：

其中的C表示近似协方差矩阵的逆矩阵：

通过上述的方法我们就可以把原来需要进行的K阶矩阵的求逆过程转变成了K/M阶矩阵的求逆加一个 阶矩阵的求逆，从而能够大大缩短计算的时间。

**4.Analysis of Method**

**4.1 对于ABCD算法的深入分析**

4.1.1 BIC准则

在实现过程中对于模型的优劣判断采用边缘似然函数，但是这样产生的问题是，自由参数更多的模型往往更容易受到青睐，而参数的增多将会导致模型的复杂度大大增加，造成可视化分析的麻烦。所以采用BIC准则去除模型的参数对判断指标的影响，具体的BIC准则公式如下：

其中的 是最优模型参数下的边缘似然值，M是模型的参数个数，N是训练数据集的个数。尽管BIC准则假设数据是独立同分布的，而这在实际中除了白噪声一般不满足，但是鉴于其简洁性和实用性，仍然是一种很好的度量方法。

4.1.2 参数初始化

每一个新的核函数的迭代开始的时候都需要设置初始参数，但是如果每一次都是随机选取初始参数不利于最优解的形成，所以采用的方式是将上一次已经得到的最优参数直接用于下一次的初始值，新加入的参数还是使用随机的方式获得。

4.1.3 嵌套限制措施

BIC方法的使用可以避免产生大量的参数，但是带来的结果是生成很多函数和的乘积的形式，这会给后面的可视化操作带来很大的困难，造成核函数模型不能够被很好地解释，基于此问题产生了两种分支：

* ABCD-accuracy —— 不加任何限制，得到精确描述的核函数模型；
* ABCD-interpretability —— 便于可视化解释的模型，在搜索过程中禁止嵌套表达式的生成。

4.1.4 生成结果的自然语言描述

在之前的算法中我们提过，对于自动生成的核函数需要有一定的自然语言描述机制来进行分析，并能够自动生成描述语句，这样才能对我们的后续工作产生更多帮助。

这种机制的基础是之前的对于核函数操作的描述，这项工作的细节由[James Robert Lloyd **[13]**](#refer13)等人完成，他们的主要贡献是构造了一种GP模型的语言，这种语言有两个特性，第一是所有核函数的描述都可以简化成“积之和”的形式，第二点是所有的积的形式都可以构造出一个连续的模型，这样的好处是我们可以将积拆分。

**4.2 BCM方法的深入分析**

在上述的BCM方法中提到了近似成立的一个条件是要求各个子数据集之间具有尽可能少的相关性，对于具有明显类别性的数据集可以通过实现对数据集的聚类来实现这一条件，但是对于无明显分类的数据集则不需要单独考虑这一问题。

在实际应用中可以使用kmeans方法进行简单地聚类，使用随机产生的训练数据（500个）及特定的测试数据得到的普通BCM与聚类BCM方法的时间及准确性比较如下：

**Time MSE**

精确GPR 15.31 seconds 0.163158

普通BCM 4.68 seconds 0.167466

聚类BCM 4.74 seconds 0.165415

表2 BCM方法比较

通过比较可以发现，精确计算的时间是使用BCM方法的4倍左右，使用聚类方法的BCM虽然花费时间较长，但是能够降低MSE误差。

**5.Algorithm**

**5.1 ABCD算法实现流程**

**Algorithm 1. Automatic Bayesian Covariance Discovery**

1.initial kernel parameters D = {}

2.set iteration number is K = k

for I = 1 : k

D\_temp ← D + {new\_kernel}

D\_temp ← D\*{new\_kernel}

get best kernel parameters set {θ} and liklihood

according to BIC set : D = D\_temp

end for

该过程可以使用流程图进行更为简单地表达：

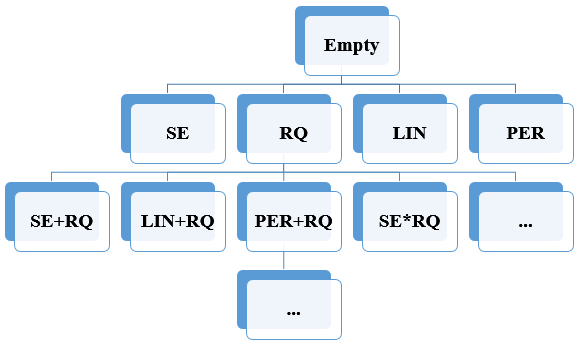


图5 ABCD算法流程图

**5.2 BCM算法实现流程**

**Algorithm 2. Bayesian Information Criterion**

1.split data set and targets into M subsets Di

2.for every subsets Di train a module

3.use this formular to get approximate mean value:

where

**6.Numerical Result**

**6.1 一维时间序列预测**

在第一个数据集上使用了三种方法进行验证，第一种是自己实现的GPR算法，第二种是利用gpml工具包实现，第三种是在第二种的基础上增加ABCD算法进行实现。

6.1.1 自己实现GPR函数方案（人工选择）

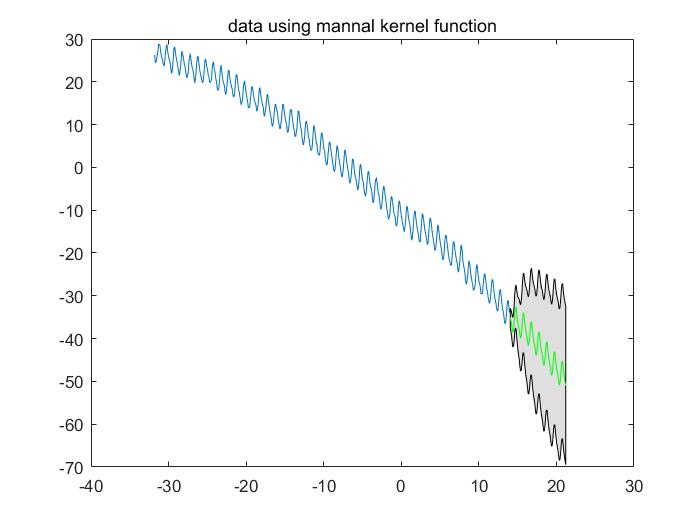


图6 自己实现的GPR

一开始自己根据相关论文上的相关数学结论自己实现了GPR的整个计算过程和训练过程，得到上图所示的预测结果与方差。

6.1.2 使用gpml工具包函数方案（人工选择）

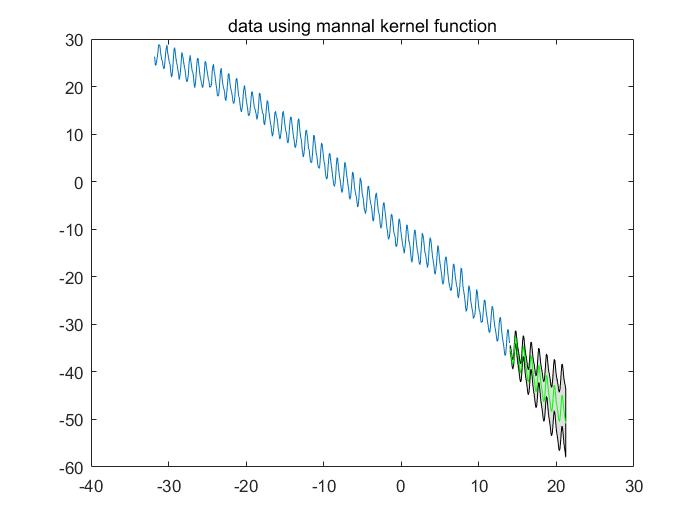
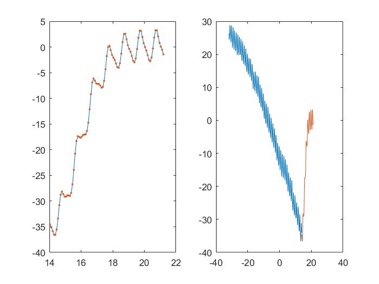
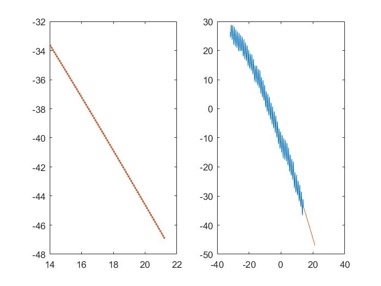


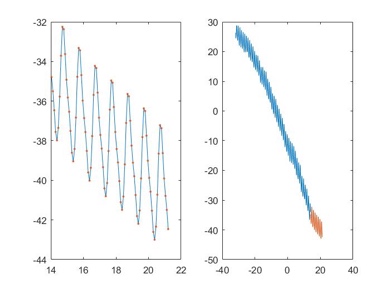
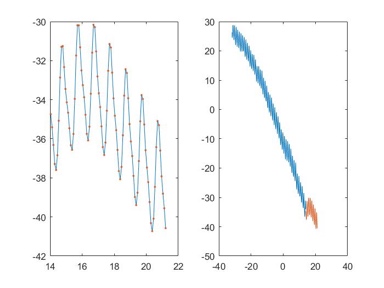
图7使用gpml工具包实现GPR

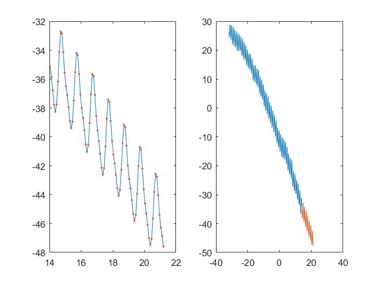
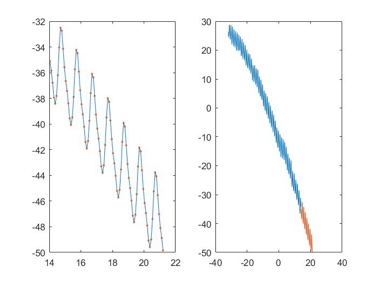
上图是使用了gpml工具包的结果，核函数模型选择的是相关论文的一个结论，并且使用了工具包中的minimize函数进行训练得到。

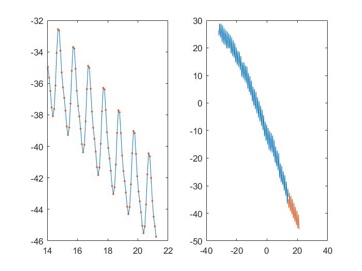
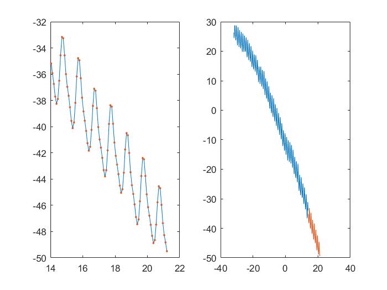
6.1.3 使用ABCD算法实现方案

使用ABCD算法的迭代次数设置为10次，每一次搜索的最优参数训练此处选择为200次，选择的均值函数是零均值函数，似然函数为高斯似然函数，得到如下结果：









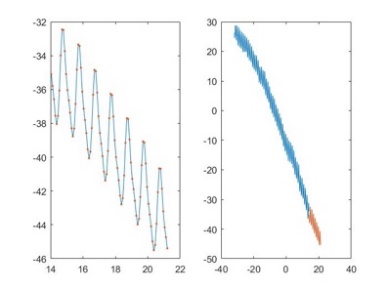
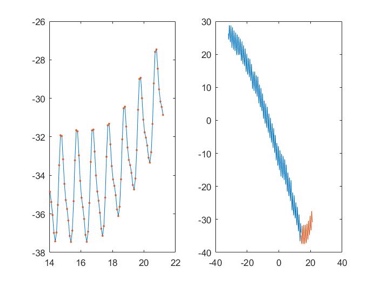


图 8 ABCD算法预测趋势，共有10张图片，每一张图的左侧是预测结果的趋势，右侧是将预测结果与原始训练数据合并之后的趋势。

得到的核函数组合和参数如下表所示：

|  |  |
| --- | --- |
| **Time** | **Kernel function** |
| 1 | 'covSEiso' |
| 2 | 'covSEiso' + 'covPeriodic' |
| 3 | 'covSEiso' + 'covPeriodic' + 'covLINiso' |
| 4 | 'covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso' |
| 5 | ('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' |
| 6 | ('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' + 'covPeriodic' |
| 7 | ('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' + 'covPeriodic' + 'covPeriodic' |
| 8 | ('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' + 'covPeriodic' + 'covPeriodic' + 'covRQiso' |
| 9 | (('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' + 'covPeriodic' + 'covPeriodic' + 'covRQiso') \* 'covRQiso' |
| 10 | ((('covSEiso' + 'covPeriodic' + 'covLINiso' + 'covRQiso')\* 'covSEiso' + 'covPeriodic' + 'covPeriodic' + 'covRQiso') \* 'covRQiso') \* 'covSEiso' |

表 3 ABCD算法得到的核函数

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mean** | **Kernel function** | **Likli** |
| 1 | [] | [3.4253;3.8029] | 0.7537 |
| 2 | [] | [0.5393;1.8138;-0.3508;0.6926;0.4292] | -1.2368 |
| 3 | [] | [0.23245;0.6507; 0.7109;  -0.0004; 0.8459; 0.0968] | -1.3902 |
| 4 | [] | [1.2571;0.4978;0.8069;-0.00037;  1.6342;0.4816;-0.7242;-0.4813;  - 0.7323] | -1.4820 |
| 5 | [] | [1.9658;-0.3013;0.4248;-0.0004;  1.3349;0.2962;-0.2888;-0.5638;  -0.3014;4.9589;0.0130] | -1.2080 |
| 6 | [] | [1.0535;0.2292;0.6993;-0.00026;  0.6233;0.4006;0.3202;0.2243;  0.0579;3.5025;0.1435;0.8820;  0.3943;-0.0133] | -1.3900 |
| 7 | [] | [1.1105;0.1708;-0.4651;0.6935;0.3019  0.5180;0.3825;-0.0407;-0.0407;  3.3676;0.3038;1.0215;1.3555;0.2030;  0.5046;1.2845;0.1457] | -1.4892 |
| 8 | [] | [0.7114;-0.2120;0.5564;3.4811e-5;  0.62880.6010;1.1587;0.3229;  0.3901;3.4488;0.1870; 0.6475;  1.3650; 0.7190;1.1198;0.6279;  0.2197; 0.1235; 0.1489;-0.04279] | -1.5793 |
| 9 | [] | [0.8298;0.4436;1.0409;0.2769;-  0.1815;0.0634;0.4985;0.3307;  0.6498;2.0866;-0.4957;0.4679;  -0.00058;1.1799;1.4998;1.4044;  0.1933;0.4581;0.3520;0.0592;  3.2871;-0.0624;-0.1107;] | -1.4801 |
| 10 | [] | [0.8213;-0.0050;0.5929;-6.7135e-05;  1.0460;0.3702;0.5432;0.1652;  0.2554;4.0207;1.2024;1.0504;  0.6634;-0.3582;1.2718;1.3416;  0.2331;0.4984;0.9496;0.2242;  3.5970;-0.4510;0.8310;4.1560;  -0.5722;] | -1.4352 |

表 4 各组核函数的参数

|  |  |  |
| --- | --- | --- |
| **Time** | **MSE** | **Max liklihood** |
| 1 | 7.75086 | -1204.5556 |
| 2 | 1261.9505 | -339.4210 |
| 3 | 53.3582 | -247.4802 |
| 4 | 21.8450 | -179.3495 |
| 5 | 0.4978 | -197.4651 |
| 6 | 3.0551 | -213.8623 |
| 7 | 0.1811 | -258.6919 |
| 8 | 8.4285 | -228.2156 |
| 9 | 103.0526 | -249.7222 |
| 10 | 9.2824 | -247.6584 |

表5各组MSE与最大似然值

6.1.4 三种方法的结果比较

**Method 核函数 MSE**

自己实现GPR SE+PER+LIN+LIN 0.437321

GPML LIN\*SE + SE\*(PER+RQ) 0.142740

ABCD算法 (SE + PER + LIN + RQ)\* SE + PER + PER 0.1811

表6 三种方法结论比较

**6.2 多维输入非时间序列预测（使用gpml toolbox）**

6.2.1 随机训练数据输入

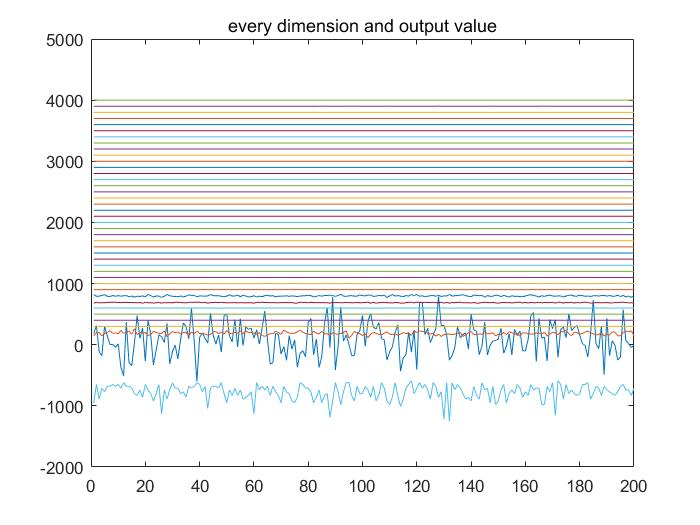


图9 每一维与输出值关系（前200个数据点）

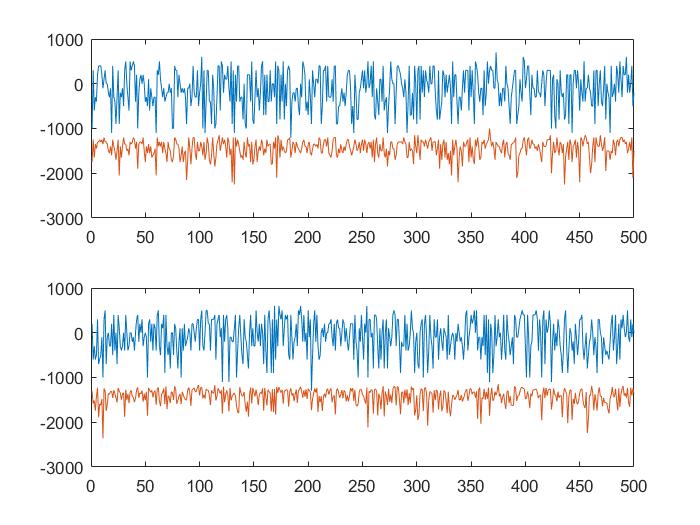


图10第七维输入输出数据的关系（前500个数据点）

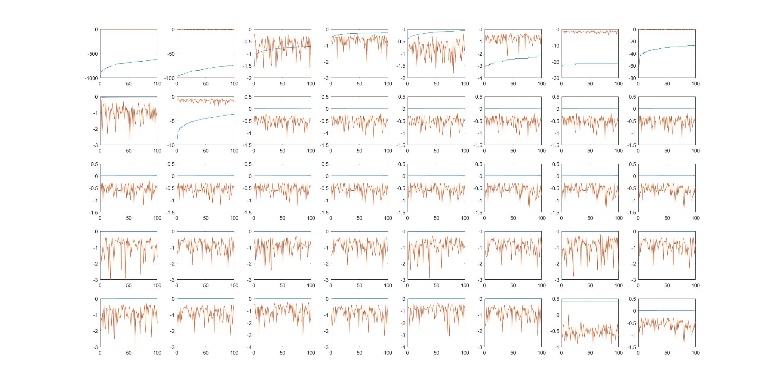


图11每一维训练数据与输出结果（前100个数据点）

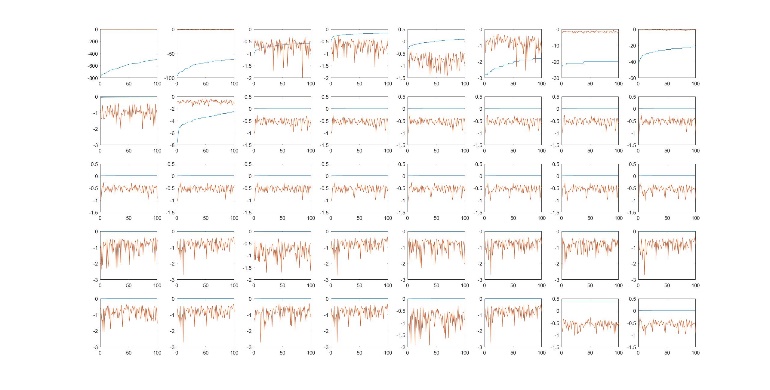


图12手动选择核函数联合输入结果（前100个数据点）

6.2.2 BCM方法近似

通过图11的对比可以看到训练数据集在分布上并没有明显的相关性，所以没有必要采用K-means等聚类的方法，并且考虑到该数据集的复杂程度，在其上采用ABCD算法并不能得到一个很好的核函数组合，故采用人工选择的核函数：

根据选取的子集的训练集的大小不同在训练时间和得到的MSE上会有较大的差别，这里分别比较了batch为100、200、500、1000的效果：

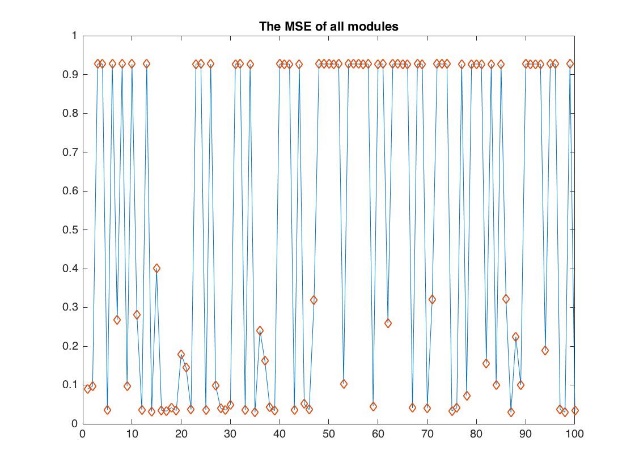


图13 batch = 100时的各个子集的MSE结果

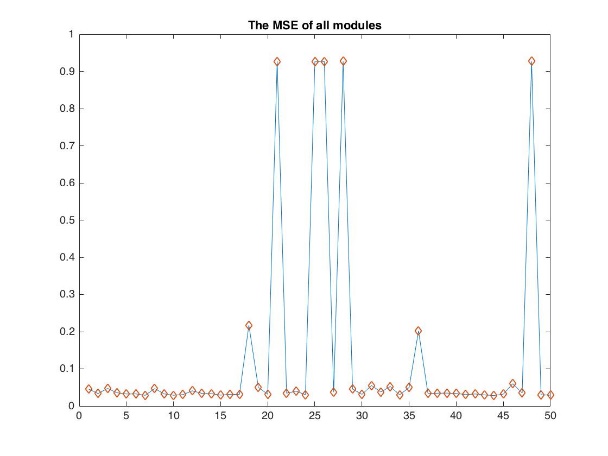


图14 batch = 200时的各个子集的MSE结果

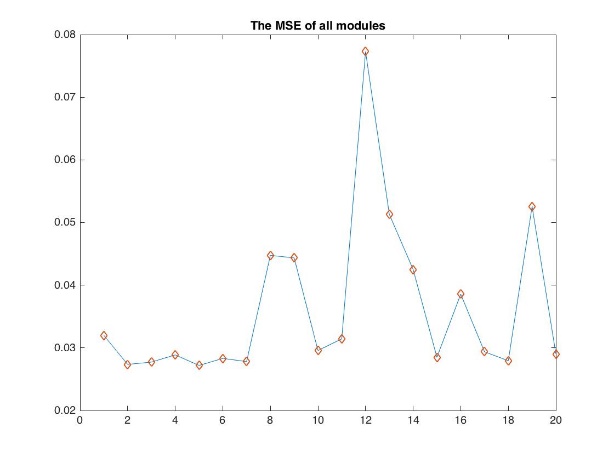


图15 batch = 500时的各个子集的MSE结果

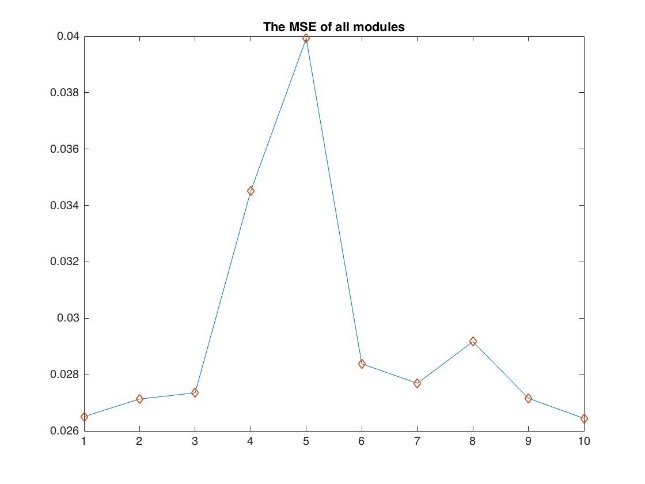


图16 batch = 1000时的各个子集的MSE结果

从上面的四幅图中可以看出当训练子集的大小取不同的值时MSE有很大的差别。

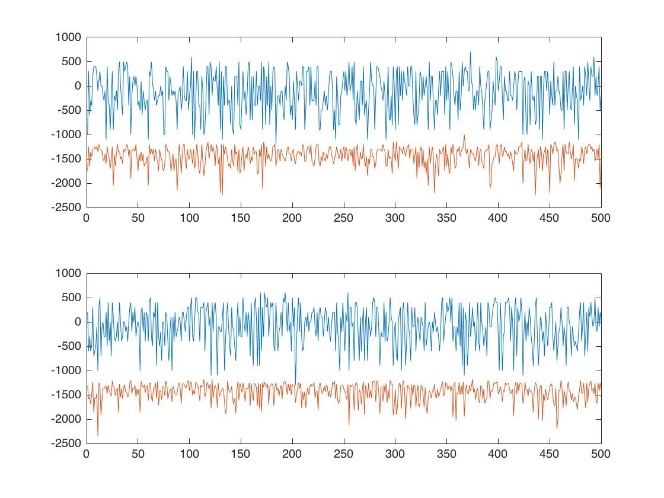


图17 第七维输入输出数据的关系（前500个数据点）

6.2.3 两种方法的结果比较

|  |  |  |
| --- | --- | --- |
| **Method** | **MSE** | **Time** |
| 随机选取训练集 | 0.025 | 2462.9s**\*** |
| BCM(batch = 100) | 39.130 | 92.55s |
| BCM(batch = 200) | 0.0309 | 87.62s |
| BCM(batch = 500) | 0.0306 | 149.07s |
| BCM(batch = 1000) | 0.0311 | 316.06 |

表7 BCM加速结果比较

\*注：随机选取训练集指的是使用随机数在10000个样本里随机选择3000个作为训练样本。由于10000个样本的运行时间过长，并不能进行全部数据集的测试。

**6.3 不同推断方法与似然函数的比较（gpml）**

|  |  |  |
| --- | --- | --- |
| **Bayes Inference** | **MSE** | **Time** |
| Gauss | 0.025569 | 83.19s |
| Laplace Approximatios | 0.025569 | 75.51s |
| Variational Bayes | 0.025569 | 295.49s |

表8 不同贝叶斯推断方法比较

|  |  |  |
| --- | --- | --- |
| **Liklihood function** | **MSE** | **Time** |
| Gauss | 0.025569 | 83.19s |
| Laplace | 1.644717 | 68.69s |

表9 不同似然函数形式比较

**7.Discussion**

**7.1 自己实现GPR和gpml方法对比**

为了能够深入了解GPR原理，在参考论文资料的情况下自己实现了GPR的全部过程，其中的训练过程采用matlab内部的函数fminsearch进行自由导数（derivative-free method）无约束最小值求解，对比指标是最大似然值的相反数。

通过与gpml的对比可以看出，由于核函数的实现上存在差异，所以并不能很好地缩小预测方差，这在图6与图7中可以看出，明显gpml方法的方差较小。

另外在训练速度上，由于gpml才用了近似方法来进行共轭梯度下降，所以速度明显快于我自己实现的方法，并且其采用的一些方法可以相对有效地避免陷入局部最小值。

**7.2人工选择核函数与ABCD生成对比**

在第一个数据集上的最后一种方法采用ABCD算法自动生成核函数，图8展示了整个迭代过程，该贪心算法假定每一次选择的最小BIC核函数组合一定是最优的，然后进行下一次迭代，为了简单起见迭代过程只采用两种操作：加和乘。

从图中可以看出第一次迭代就可以得到整体的线性下降趋势，即选择covSE核函数，从而得到一个较小的似然值，第二次迭代考虑到训练数据具有一定的局部波动性，所以增加了covPER函数进行局部性描述。但是此时破坏了全局性的描述所以呈现出了增长趋势的局部振荡，第三次迭代则纠正了这一不正确的全局趋势。后面的迭代过程逐渐进行纠正最终在第七次达到了最好的MSE误差。

随着ABCD继续迭代模型复杂度开始增加，自由参数的数量也在增加，加剧了数据描述的困难程度，进而出现了较差的似然值和MSE结果，此时再继续迭代已经没有意义，因此可以得出结果，该方法的迭代次数不宜过多，否则容易产生过拟合的情况。

手动选择核函数的过程本质上也是对数据趋势的一个判断，但是由于核函数的组合空间是无限维度的，所以人工选择工作量很大，但是可以通过ABCD方法得到的结论并运用自然语言描述的生成方法来更加精确地看待训练数据。

**7.3高维数据集的复杂性**

第二个数据集有40维输入数据，通过图9可以看到，上面的曲线是每一维的输入量，最下面的曲线是输出值（只取了前200个数据点），大概只有前7维的波动性比较大，也就是说其他维度的影响较小，因为其值基本位于一个很小的数量级。

图11和图12分别表示了按照每一维排序之后的输入与输出关系。一共有40张图，每一张图都表示了该输入通道与输出通道的关系，图11是训练集和结果，可以看到，输出量与每一维都没有明显的相关关系，这说明输出量是由多维共同决定的，这也是本数据集的难点所在，我们需要建立各个维度之间的关联性。图12是使用gpml工具包进行40维联合输入的结果，核函数选择的是：

其中的SE-ARD函数是可以自动建立各个维度之间的关联的，并且考虑到观测噪声，我们增加了一项WN函数。

由于训练集的数据量是10000，并且由于该数据集是非时间序列，所以在训练时从中随机选择了3000个作为训练样本，但是这样极大地浪费了数据集，所以需要采用更加快速的近似算法。

**7.4随机选取法与BCM方法时间对比**

通过在第二个数据集上随机选择训练子集合BCM方法的时间上的对比可以清楚的看到，近似方法在时间上得到了极大的加速，并且在准确度，即MSE指标上并没有特别明显的下降，这说明该近似方法是可以采用的。

并且对于表7中的对比可以发现，随着训练子集的大小下降运行速度获得了很大的提升，但是当自己大小选取的过小时，由于样本数目不足，造成训练的效果变差，MSE迅速上升。

进一步分析该加速的结果主要是有一求逆矩阵的规模减小造成的，并且采用了加权的方式来整合每一个模块的训练结果能够建立各个模块之间的关联，而不是进行分别的训练。

**7.5不同推断方法的对比**

在表8和表9中所示，使用拉普拉斯近似和变分贝叶斯推断方法所得到的MSE与高斯方法一致，但是在时间上有所差别，其中拉普拉斯近似方法是最快的。另外在尝试MCMC方法时由于采样次数太多导致整个程序运行速度太慢无法测出有效数据。

在似然函数方面，只测试了拉普拉斯似然函数的结果，但是得到的MSE效果并不理想。

**8.Conclusion**

本文主要实现了两种方法，一种是自动生成核函数的ABCD算法，另一种是针对大量训练集的近似求解BCM方法，并分别将这两种方法应用在两个不同的数据集上进行验证，并且均能够达到较好的效果。

对于大量数据的加速还可以使用近似推断的方法，比如[变分贝叶斯（Variational Bayes）**[16]**](#refer16)、拉普拉斯算法近似（Laplace Approximately）、[马尔科夫蒙特卡洛（MCMC）**[17]**](#refer17) 和Expectation Propagation（EP）等等，但是限于时间的原因，目前只是大体上了解了其中的原理，并没有来得及实现。并且尽管在gpml工具包中已经实现了上述的推断方法，但是在使用过程中出现了一些问题，导致使用近似方法的时间比直接使用高斯判断的时间还要长。

另一方面，由于第二个数据集是具有40维输入向量的，那么如何很好地表达各个维度之间的关系就是一个很重要的问题。对于这一问题我查阅了一些方法，比如对于[多响应变量情况下的优化**[14]**](#refer14) 还有[Bledar A. Konomi**[15]**](#refer15) 在2015的工作。但是同样是限于时间的原因，还没有来得及完整地理解他们的算法和理论。

**Reference**

[1] 《Structure Discovery in Nonparametric Regression through Compositional Kernel Search》David Duvenaud，James Robert Lloyd，Roger Grossez，Joshua B. Tenenbaumz，Zoubin Ghahramaniy

[2] <http://www.GaussianProcess.org/gpml>

[3] 《高斯过程回归方法综述》，***控制与决策***（2013），何志昆、刘光斌、赵曦晶、王明昊

[4] Charnes, A.; Frome, E. L.; Yu, P. L. (1976). 《The Equivalence of Generalized Least Squares and Maximum Likelihood Estimates in the Exponential Family》. Journal of the American Statistical Association. 71 (353): 169–171.

[5] 《Gaussian Processes for Machine Learning》，C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

[6]《Automatic Model Construction with Gaussian Processes》David Kristjanson Duvenaud，University of Cambridge

[7] 《Some Aspects of the Spline Smoothing Approach to Non-parametric Regression》，Silverman, B. W. (1985).Curve Fitting (with discussion). J. Roy. Stat. Soc. B, 47(1):1–52.

[8] 《Using the Nystr¨om Method to Speed Up Kernel Machines》，Williams, C. K. I. and Seeger, M. (2001).In Leen, T. K., Diettrich, T. G., and Tresp, V., editors, Advances in Neural Information Processing Systems 13, pages 682–688. MIT Press.

[9] 《Fast Sparse Gaussian Process Methods: The Informative Vector Machine》. Lawrence, N., Seeger, M., and Herbrich, R. (2003). In Becker, S., Thrun, S., and Obermayer, K., editors, Advances in Neural Information Processing Systems 15, pages 625–632. MIT Press.

[10] 《Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations》. PhD thesis, School of Informatics, University of Edinburgh

[11] 《Sparse On-Line Gaussian Processes. Neural Computation》, Csat´o, L. and Opper, M. (2002).

[12] 《A Bayesian Committee Machine》Tresp, V. (2000). Neural Computation, 12(11):2719–2741

[13] 《Automatic construction and natural-language description of nonparametric regression models》，James Robert Lloyd, David Duvenaud, Roger B. Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. In Association for the Advancement of Artifcial

Intelligence (AAAI), 2014.

[14] 《Gaussian process regression with multiple response variables》Bo Wanga,∗, Tao Chenb

[15] 《LOW-COST MULTI-DIMENSIONAL GAUSSIAN

PROCESS WITH APPLICATION TO UNCERTAINTY

QUANTIFICATION》Bledar A. Konomi1 & Guang Lin2

[16] 《Variational Bayes Method in Gaussian Process Regression》MENSAH.DAVID.KWAMENA

[17] 《MCMC Methods for Gaussian Process Models Using Fast Approximations for the Likelihood》Chunyi Wang，Radford M. Neal