文章编号: 1001-0920(2017)07-1319-06 DOI: 10.13195/j.kzyjc.2016.0492

基于拉普拉斯方法的大规模高斯过程分类算法

马 彪[1], 贺建军[1,2†], 李厚杰[1]

(1. 大连民族大学 信息与通信工程学院，辽宁 大连 116600；
2. 大连理工大学 电子信息与电气工程学部，辽宁 大连 116024)

摘 要: 基于KL散度的大规模变分高斯过程分类算法(KLSP)需要同时对诱导变量的均值向量和协方差矩阵进行优化,这会给模型求解带来一定的挑战. 基于拉普拉斯方法建立一种改进算法: 首先为诱导变量的后验分布构造一个易于计算的下界; 然后利用拉普拉斯方法计算该下界的一个高斯逼近作为诱导变量的后验分布函数的近似表达式,将问题转换为一个只与均值向量有关的凸优化问题,从而降低了模型的求解难度. 仿真实验结果表明,所提出的改进算法在速度和精度上都较原始算法有了明显提高.

关键词: 大规模数据分类；高斯过程模型；拉普拉斯方法；变分方法

中图分类号: TP391 **文献标志码:** A

Large-scale Gaussian process classification via Laplace's method

MA Biao[1], *HE Jian-jun*[1,2†], *LI Hou-jie*[1]

(1. College of Information and Communication Engineering，Dalian Minzu University，Dalian 116600，China；2. Faculty of Electronic Information and Electrical Engineering，Dalian University of Technology，Dalian 116024，China)

Abstract: Scalable variational Gaussian process classification based on KL divergence(KLSP) is generaly plagued by the fact that both the mean and covariance matrix of inducing variables need to be optimized, therefore an improved algorithm is proposed by using the Laplace's method. Firstly, a lower bound of the posterior distribution of inducing variables is constructed, which can be computed easily. Then a Gaussian approximation of this lower bound is computed, which is by using the Laplace's method taken as the approximation the posterior distribution of inducing variables. As a result, the objective function of the proposed algorithm is a concave function with respect to the mean of inducing variables only, which can be solved easily. The experimental results show that, compared with the original algorithm, the speed and accuracy of the proposed algorithm are improved obviously.

Keywords: large-scale data classification；Gaussian process model；Laplace's method；variational method

0 引 言

高斯过程模型[1]是继支持向量机之后又一种受到人们广泛关注的核机器学习方法,现已大量地用于回归[2]、分类[3]、关系学习[4]、半监督学习[5]、排序学习[6]、多任务学习[7]、多示例多标记学习[8]等各种机器学习问题中. 由于需要对协方差矩阵求逆,高斯过程模型在训练阶段的计算复杂度通常为$O(n^3)$(其中n为训练样本数目),只能处理训练数据规模较小的问题. 为了降低算法的计算复杂度,人们提出了构建稀疏模型的问题解决策略. 对于回归问题,由于可以直接计算得到潜变量的后验概率分布的分析表达式,模型的求解较为简单,人们已经提出了很多行之有效的稀疏高斯过程回归算法[9-10]; 对于分类问题,由于定义的似然函数通常是一个S形函数,并不能得到后验概率分布的分析表达式,大部分回归算法并不能直接推广到分类问题上,目前只有IVM[11]、FIFC[12]和KLSP[13]等几种稀疏高斯过程分类算法. KLSP算法是基于变分原理提出的一种具有较高精度的算法,但是该算法需要同时对诱导变量的均值向量和协方差矩阵进行求解,而且需要保证协方差矩阵为半正定矩阵,这就给模型求解造成一定的难度. 拉普拉斯方法[14]是一种在逼近后验概率时广泛使用的方法,而

收稿日期: 2016-04-23；**修回日期:** 2016-08-31.

基金项目: 国家自然科学基金项目(61503058, 61374170)；辽宁省自然科学基金项目(201602190, 2015020084, 2015020099)；辽宁省教育厅科学技术研究项目(L2014540, L2015127)；大连市青年科技之星项目(2016RQ072)；中央高校基本科研业务费专项资金项目(DC201501055, DC201501060201).

作者简介: 马彪(1962−), 男, 工程师, 学士, 从事智能控制与信息处理的研究；贺建军(1983−), 男, 副教授, 博士, 从事机器学习与数据挖掘的研究.

†通讯作者. E-mail: jianjunhe@live.com

且利用拉普拉斯方法逼近后验概率时不需要对协方差矩阵进行优化,只需对潜变量的均值向量进行优化即可,这将极大地降低模型的求解难度. Opper等[15]最近发现,利用变分法逼近潜变量的后验概率分布与利用拉普拉斯方法逼近潜变量的后验概率分布在一定的条件下是等价的.

受以上信息启发,本文在KLSP算法的基础上,利用拉普拉斯方法建立一种新的大规模高斯过程分类算法,使该算法在保留KLSP算法的精度优势的前提下,具有比KLSP算法更低的计算复杂度.

1 模型建立

设X为特征空间,$S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$为包含$n$个样本的训练集,其中$x_i\in X$和$y_i\in\{+1,-1\}$分别是第$i$个样本的特征向量和类别标记. 为便于描述,下面将用$D=\{x_1,x_2,\cdots,x_n\}$和$Y=[y_1,y_2,\cdots,y_n]^{\mathrm{T}}$分别表示$S$中所有训练样本的特征向量构成的集合和类别标记构成的集合. 构建分类算法实质上就是利用训练集S在特征空间中建立一个能正确输出待预测样本的真实标记的函数$f(x)$.

高斯过程模型的基本思想是,在服从某个高斯过程分布的潜变量函数簇中寻找最有可能输出待预测样本的真实标记的函数. 要实现该目的,通常包括定义先验概率、定义似然函数、计算后验概率和计算预测概率4个核心模块. 与传统高斯过程模型直接利用潜变量的先验概率和似然推导其后验概率的方式不同,目前常用的稀疏高斯过程模型通常是利用一组中间诱导变量来计算潜变量的后验概率分布,这样可以有效降低算法的计算复杂度. 不同稀疏模型的主要区别在于如何利用诱导变量计算后验概率的问题,本文算法的主要创新也是在后验概率的计算上,其他3个模块与KLSP算法是一样的. 但是,为了便于理解,下面将按照顺序对这4个模块进行描述,其中“计算后验概率”模块可以体现本文的动机和创新点.

1.1 定义先验概率

假设潜变量函数$f(x)$服从如下零均值高斯过程分布:

$$f(x)\sim GP(0,k(x,x')),\tag{1}$$

其中$k(x,x')$表示协方差函数. 本文采用的协方差函数为$k(x,x')=\alpha\mathrm{e}^{-||x-x'||^2/\beta}$.

假设$U=\{\bar{x}_1,\bar{x}_2,\cdots,\bar{x}_m\}\subset D$是诱导子集(本文采用Chen[16]建立的快速聚类算法对训练样本进行聚类,然后将得到的m个聚类中心作为U),$x_*\in X$为待预测样本,则根据式(1)可以得到$f(x)$在D、U和x_*上的函数值F_D、F_U和f_*的先验概率分布和条件先验概率分布,即

$$\begin{cases}p(F_D|D)=N(F_D|0,K_D),\\ p(F_U|D)=N(F_U|0,K_U),\\ p(F_D|F_U,D)=N(F_D|K_{DU}K_U^{-1}F_U,K_D-\\ \qquad K_{DU}K_U^{-1}K_{DU}^{\mathrm{T}}),\\ p(f_*|F_U,D,x_*)=N(f_*|K_{*U}K_U^{-1}F_U,K_{**}-\\ \qquad K_{*U}K_U^{-1}K_{*U}^{\mathrm{T}}).\end{cases}\tag{2}$$

其中: F_D是$f(x)$在样本集D上的函数值构成的列向量,即$F_D=[f_1,f_2,\cdots,f_n]^{\mathrm{T}}$,$f_i=f(x_i)$; K_D是协方差函数$k(x,x')$在D上的值构成的方阵,第i行j列上的元素是$k(x_i,x_j)$; F_U是$f(x)$在诱导集U上的函数值构成的列向量; K_U是协方差函数$k(x,x')$在U上的值构成的方阵; K_{DU}表示D和U中样本之间的协方差函数值构成的矩阵; $f_*=f(x_*)$是$f(x)$在待预测样本x_*上的值; K_{*U}表示x_*和U中样本之间的协方差函数值构成的行向量; $K_{**}=k(x_*,x_*)$.

1.2 定义似然函数

与传统高斯过程二分类算法一样,联合似然函数$p(Y|F_D)$可以定义为

$$p(Y|F_D)=\prod_{i=1}^{n}p(y_i|f_i)=\frac{1}{1+\mathrm{e}^{-y_if_i}}.\tag{3}$$

1.3 计算后验概率

在传统高斯过程模型中,F_D的后验概率可由贝叶斯公式

$$p(F_D|D,Y)=\frac{p(Y|F_D)p(F_D|D)}{\int p(Y|F_D)p(F_D|D)\mathrm{d}F_D}$$

计算得到. 由于在分类问题中,似然函数$p(Y|F_D)$是一个非高斯函数,并不能得到$p(F_D|D,Y)$的分析表达式,而直接利用逼近的方法计算$p(F_D|D,Y)$需要较高的计算复杂度. 为了降低计算复杂度,可以引入一个诱导变量F_U,利用F_U的后验概率$p(F_U|D,Y)$来计算得到$p(F_D|D,Y)$,即

$$p(F_D|D,Y)=\int p(F_D|F_U,D)p(F_U|D,Y)\mathrm{d}F_U.\tag{4}$$

其中$p(F_U|D,Y)$的表达式如下:

$$\begin{cases}p(F_U|D,Y)=\dfrac{p(Y|F_U)p(F_U|D)}{\int p(Y|F_U)p(F_U|D)\mathrm{d}F_U},\\ p(Y|F_U)=\int p(Y|F_D)p(F_D|F_U,D)\mathrm{d}F_D.\end{cases}\tag{5}$$

由于$p(F_U|D,Y)$中也包含非高斯函数$p(Y|F_D)$,仍然不能得到它的分析表达式. KLSP算法的基本思想是利用变分法来计算$p(F_U|D,Y)$的一个高斯逼近$q(F_U)=N(F_U|\mu,\Sigma)$,即通过最小化KL散度$\mathrm{KL}(q(F_U)||p(F_U|D,Y))$来计算$q(F_U)=N(F_U|\mu,\Sigma)$. 由于需要同时对$\mu$和$\Sigma$进行优化,而且要保证$\Sigma$

是半正定矩阵,在模型求解时有一定的难度. 本文利用拉普拉斯方法来计算$p(F_U|D,Y)$的一个高斯逼近$q(F_U)=N(F_U|\mu,\Sigma)$,即

$$\begin{cases}\mu=\underset{F_U}{\operatorname{argmax}}\log p(F_U|D,Y),\\ \Sigma=-(\nabla\nabla\log p(F_U|D,Y)|_{F_U=\mu})^{-1}.\end{cases}\tag{6}$$

由式(6)可以看出,利用拉普拉斯方法计算$q(F_U)=N(F_U|\mu,\Sigma)$只需通过优化$\underset{F_U}{\operatorname{argmax}}\log p(F_U|D,Y)$就可以同时得到$\mu$和$\Sigma$,因此,无论是计算复杂度还是模型求解难度都低于KLSP算法. 下面对其中的技术细节进行详细描述.

由式(5)可以看出,$\log p(F_U|D,Y)$的表达式主要包括$\log p(Y|F_U)$、$\log\int p(Y|F_U)p(F_U|D)\mathrm{d}F_U$和$\log p(F_U|D)$三项,其中$\log p(Y|F_U)$的具体表达式为

$$\log\int p(Y|F_D)p(F_D|F_U,D)\mathrm{d}F_D.$$

由于包含一个高维积分,计算$\log p(Y|F_U)$关于F_U的导数是一个很困难的事情,利用式(6)求解$q(F_U)=N(F_U|\mu,\Sigma)$的设想实际上也是行不通的. 为了解决$\log p(Y|F_U)$计算困难的问题,本文不直接利用$\log p(F_U|D,Y)$,而是利用它的一个下界来计算$q(F_U)=N(F_U|\mu,\Sigma)$,即

$$\begin{aligned}&\log p(F_U|D,Y)=\\&\log p(Y|F_U)+\log p(F_U|D)-\\&\log\int p(Y|F_U)p(F_U|D)\mathrm{d}F_U\geqslant\\&\int(\log p(Y|F_D))p(F_D|F_U,D)\mathrm{d}F_D+\\&\log p(F_U|D)-\log\int p(Y|F_U)p(F_U|D)\mathrm{d}F_U.\end{aligned}\tag{7}$$

由于$\log\int p(Y|F_U)p(F_U|D)\mathrm{d}F_U$不包含$F_U$,可以去掉,利用下式计算$q(F_U)=N(F_U|\mu,\Sigma)$:

$$\begin{cases}\mu=\underset{F_U}{\operatorname{argmax}}\,\psi(F_U),\\ \Sigma=-(\nabla\nabla\psi(F_U)|_{F_U=\mu})^{-1}.\end{cases}\tag{8}$$

其中$\psi(F_U)$的表达式为

$$\begin{aligned}\psi(F_U)=&\int(\log p(Y|F_D))p(F_D|F_U,D)\mathrm{d}F_D+\\&\log p(F_U|D)=\\&\sum_{i=1}^{n}\Big(\int\log p(y_i|f_i)p(f_i|F_U,D)\mathrm{d}f_i\Big)-\\&\frac{1}{2}(F_U^{\mathrm{T}}K_U^{-1}F_U+\log|2\pi K_U|).\end{aligned}\tag{9}$$

分别对$\psi(F_U)$求一阶和二阶偏导数,可得

$$\begin{cases}\nabla\psi(F_U)=K_U^{-1}K_{DU}^{\mathrm{T}}d-K_U^{-1}F_U,\\ \nabla\nabla\psi(F_U)=\\ -K_U^{-1}K_{DU}^{\mathrm{T}}\mathrm{diag}(w)K_{DU}K_U^{-1}-K_U^{-1}.\end{cases}\tag{10}$$

其中

$$\begin{aligned}&d=[d_1,d_2,\cdots,d_n]^{\mathrm{T}},\\&d_i=\int\frac{y_i}{1+\mathrm{e}^{y_if_i}}p(f_i|F_U,D)\mathrm{d}f_i,\\&w=[w_1,w_2,\cdots,w_n]^{\mathrm{T}},\\&w_i=\int\frac{\mathrm{e}^{y_if_i}}{(1+\mathrm{e}^{y_if_i})^2}p(f_i|F_U,D)\mathrm{d}f_i.\end{aligned}$$

由于是一元积分,d_i和w_i可以通过Gauss-Hermite求积公式计算得到,$\mathrm{diag}(w)$表示以向量w为对角元的对角矩阵.

由式(10)可以看出,在本文算法中,目标函数的一、二阶导数的计算复杂度为$O(nm)$和$O(nm^2)$;而在KLSP算法中,目标函数的一阶导数的计算复杂度为$O(nm^2)$,且二阶导数很难求得. 因此,本文算法的计算复杂度明显低于KLSP算法.

可以看出,Hessian矩阵$\nabla\nabla\psi(F_U)$是一个负定矩阵,这表明$\psi(F_U)$是一个凹函数,因此可以采用常用的优化方法求解$\psi(F_U)$的最大值点. 本文采用牛顿迭代法对其求解,迭代公式如下:

$$F_U^{\mathrm{new}}=F_U-\lambda_0(\nabla\nabla\psi(F_U))^{-1}\nabla\psi(F_U),\tag{11}$$

其中步长λ_0为一元函数

$$\varphi(\lambda)\triangleq\psi(F_U-\lambda(\nabla\nabla\psi(F_U))^{-1}\nabla\psi(F_U))\tag{12}$$

的最大值点,本文采用二分法对λ_0进行求解.

得到$\psi(F_U)$的最大值点$\hat{F}_U$后,将其代入式(8)即可得到μ和Σ,即

$$\mu=\hat{F}_U,\ \Sigma=-(\nabla\nabla\psi(F_U)|_{F_U=\hat{F}_U})^{-1}.$$

1.4　计算预测概率

在得到后验概率分布$p(F_U|D,Y)$的高斯逼近$q(F_U)=N(F_U|\mu,\Sigma)$后,可以得到f_*的如下后验概率分布:

$$\begin{aligned}&p(f_*|D,Y,x_*)=\\&\int p(f_*|F_D,D,x_*)p(F_D|D,Y)\mathrm{d}F_D=\\&\int p(f_*|F_D,D,x_*)\int p(F_D|F_U,D)\times\\&p(F_U|D,Y)\mathrm{d}F_U\mathrm{d}F_D=\\&\int p(f_*|F_U,D,x_*)p(F_U|D,Y)\mathrm{d}F_U\approx\\&\int p(f_*|F_U,D,x_*)q(F_U)\mathrm{d}F_U=\\&N(f_*|K_{*U}K_U^{-1}\mu,K_{**}+\\&K_{*U}(K_U^{-1}\Sigma K_U^{-1}-K_U^{-1})K_{*U}^{-1}).\end{aligned}\tag{13}$$

可以看出,f_*的后验概率$p(f_*|D,Y,x_*)$主要与μ和Σ有关. 根据$p(f_*|D,Y,x_*)$,可以得到样本x_*是正样本

的概率为

$$p(y_*=+1|D,Y,x_*)=\int\frac{1}{1+\mathrm{e}^{-f_*}}p(f_*|D,Y,x_*)\mathrm{d}f_*. \quad (14)$$

由于是一元积分, 式(14)同样可以利用Gauss-Hermite求积公式计算其值.

2 算法实现

本文算法的流程如算法1所示. 在训练和预测阶段都需要频繁地用到协方差矩阵K_U的逆矩阵, 而在实际问题中K_U有时会是一个奇异矩阵, 为了保证数值稳定性, 在计算K_U的逆矩阵时加入一个扰动项$10^{-7}\boldsymbol{I}$, 即$K_U^{-1}=(K_U+10^{-7}\boldsymbol{I})^{-1}$. $\lambda_0=\arg\max\limits_{\lambda}\psi(\hat{F}_U+\lambda\Delta F_U)$是一个一维凹优化问题, 本文采用二分法计算$\dfrac{\partial\psi(\hat{F}_U+\lambda\Delta F_U)}{\partial\lambda}$的零点的方式来计算步长$\lambda_0$. 此外, Gauss-Hermite求积公式的采样点个数为20.

算法1.

1) 训练阶段.

输入: S,m,α,β. 其中: S为训练数据集, m为诱导子集的样本个数, α、β为协方差函数$k(x,x')$的参数.

输出: U,μ,Σ,K_U^{-1}.

Step 1: 利用快速K均值聚类算法选取诱导子集U;

Step 2: 计算协方差矩阵K_D、K_U、K_{DU}, 计算$K_U^{-1}=(K_U+10^{-7}\boldsymbol{I})^{-1}$;

Step 3: 初始化$\hat{F}_U$为零向量;

Step 4: 根据式(10)计算目标函数$\psi(\cdot)$在$\hat{F}_U$处的梯度$\nabla\psi(\hat{F}_U)$, $\nabla\nabla\psi(\hat{F}_U)$;

Step 5: 如果$\dfrac{\sqrt{(\nabla\psi(\hat{F}_U))^{\mathrm{T}}\nabla\psi(\hat{F}_U)}}{m}<\varepsilon$, 则转到Step 9, 否则转到Step 6, 本文实验中ε的取值为0.01;

Step 6: 确定下一步迭代的搜索方向为$\Delta F_U=-(\nabla\nabla\psi(\hat{F}_U))^{-1}\nabla\psi(\hat{F}_U)$;

Step 7: 利用二分法计算步长$\lambda_0=\arg\max\limits_{\lambda}\psi(\hat{F}_U+\lambda\Delta F_U)$, 结束条件是区间长度小于0.001时结束二分;

Step 8: 更新$\hat{F}_U$, $\hat{F}_U$=$\hat{F}_U$+$\lambda_0\Delta F_U$, 转到Step 4;

Step 9: μ=$\hat{F}_U$, Σ=$-(\nabla\nabla\psi(\hat{F}_U))^{-1}$.

2) 预测阶段.

输入: $U,\mu,\Sigma,K_U^{-1},x_*,\alpha,\beta$;

输出: x_*的真实标记是+1的概率.

Step 1: 计算协方差矩阵K_{*U}、K_{**};

Step 2: 根据式(13)计算$p(f_*|D,Y,x_*)$;

Step 3: 根据式(14)计算x_*的真实标记是+1的概率.

3 仿真实验

本节将在10个大规模二分类数据集上验证本文所提出算法的性能. 这10个数据集全部来自于UCI数据库, 其中: Mnist0数据集是在原始的Mnist多类数据集基础上以手写数字0所在类别为正类、其他类别为负类形成的二分类数据集; Covtype1、Connect1、SensIT Verhicle1和Shuttle1数据集是分别以原始的多类数据集中的第1类样本为正类、其他样本为负类形成的二分类数据集; Poker1和Poker2数据集是分别以原始的Poker-hand多类数据集的第1和第2类为正类、其他类别为负类形成的二分类数据集. 关于这些数据集的详细信息见表1.

表1 算法验证所用数据集

数据集	特征个数	正样本个数	负样本个数
Protein	357	11 310	13 077
Pendigits	16	1 143	9 849
Covtype1	54	211 840	369 172
Connect1	42	44 473	23 084
Mnist0	780	5 923	54 077
Skin	3	50 859	194 195
SensIT Vehicle1	100	22 841	75 687
Shuttle1	9	45 586	12 414
Poker2	10	43 3097	591 913
Poker1	10	513 702	511 308

本文是在文献[13]的变分高斯过程分类算法(KLSP)的基础上提出的一种新的大规模高斯过程分类算法, 因此, 下面将对本文所建算法(LASP)与KLSP算法进行对比和分析. 为了公平起见, 两个算法都采用$k(x,x')=\alpha\mathrm{e}^{-||x-x'||^2/\beta}$作为协方差函数, 其中, 协方差函数的参数$\alpha$和$\beta$的取值分别在集合$\{10^{-5},10^{-4},\cdots,10^{4}\}$和$\{3^{-4}d_u,3^{-3}d_u,\cdots,3^{4}d_u\}$($d_u$为诱导子集与训练集的样本之间的平均距离)中通过交叉验证法选取确定. 以下实验结果都是通过随机选取1/2的样本作为训练数据, 1/2的样本作为测试数据, 然后重复进行5次实验得到的平均结果, 运行设备为CPU主频3.2 GHz、内存8 GB的组装PC机.

表2给出了两个算法在10个数据集上当诱导集的样本个数取50、100、150、200四个不同值时的结果. 其中: Acc表示预测精度, Time表示计算时间(训练时间+预测时间).

表 2　LASP和KLSP算法在UCI数据集上的性能比较

数据集	算法	诱导集样本个数							
		50		100		150		200	
		Acc	Time	Acc	Time	Acc	Time	Acc	Time
Protein	LASP	66.51	0.6	70.02	0.9	71.31	1.2	72.65	1.4
	KLSP	66.50	4.5	70.09	5.4	71.49	6.3	72.93	8.0
Pendigits	LASP	99.75	0.7	99.83	0.9	99.90	1.1	99.91	1.3
	KLSP	99.56	12.9	99.76	10.4	99.86	9.6	99.87	10.2
Covtype1	LASP	67.46	31.5	71.62	42.0	73.40	52.4	75.05	62.4
	KLSP	65.10	45.6	67.51	59.4	71.05	78.0	73.03	99.2
Connect1	LASP	77.25	2.5	79.07	3.5	80.38	4.7	81.08	5.6
	KLSP	74.69	11.6	77.62	15.66	79.07	20.2	80.80	22.1
Mnist0	LASP	98.35	1.3	98.81	1.8	98.89	2.6	98.98	3.0
	KLSP	98.35	4.4	98.80	7.2	97.61	11.7	98.94	14.0
Skin	LASP	98.18	13.6	98.34	16.4	98.57	20.2	98.61	23.2
	KLSP	95.65	19.6	96.43	24.4	96.64	32.5	96.39	41.7
SensIT Vehicle1	LASP	87.01	5.0	88.19	6.6	88.81	8.0	89.28	9.4
	KLSP	86.48	21.1	86.42	20.8	86.07	22.3	85.01	25.5
Shuttle1	LASP	98.31	2.8	99.06	3.5	99.27	4.3	99.43	5.6
	KLSP	96.42	4.9	96.74	6.8	97.18	9.9	97.36	13.2
Poker2	LASP	59.42	29.6	59.67	36.5	59.97	46.7	60.23	57.4
	KLSP	58.41	102.5	57.89	154.4	58.76	204.2	59.02	254.5
Poker1	LASP	60.12	40.0	60.31	54.3	60.99	69.0	61.37	79.9
	KLSP	59.09	100.8	57.60	134.7	59.13	192.	58.54	237.6

由表2可以看出,正如所期望的,本文算法LASP的计算时间在所有数据集上都低于KLSP算法,最高降低了约10倍(如在Pendigits数据集上,本文算法的计算时间约1 s,而KLSP算法约10 s),最低也降低了约1倍;此外,令人惊奇的是,本文算法LASP的预测精度在Covtype1、Connect1、Skin、SensIT Vehicle1、Shuttle1、Poker1等7个数据集上竟然比KLSP算法高出了最少1个百分点,有的甚至能达到3个百分点.

为了进一步明确两种算法的预测精度之间究竟有无明显差距,本文利用5%显著性水平的t检验方法对两种算法在各个数据集上的预测精度进行了分析,如表3所示. 表3中所用符号的定义如下:如果t检验结果显示算法A的预测精度在统计意义上高于算法B的预测精度,则记作$A > B$;如果在统计意义上两种算法的预测精度是一样的,则记作$A = B$. 表3的结果表明,除在Protein、Pendigits和Mnist0三个数据集上两种算法的预测精度在统计意义上是一样的之外,在其他所有数据集上本文算法的预测精度从统计上看均高于KLSP算法.

表 3　LASP和KLSP算法的预测精度的t检验比较结果

数据集	诱导集样本个数			
	50	100	150	200
Protein	LASP = KLSP	LASP = KLSP	LASP = KLSP	LASP = KLSP
Pendigits	LASP > KLSP	LASP > KLSP	LASP = KLSP	LASP > KLSP
Covtype1	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Connect1	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Mnist0	LASP = KLSP	LASP = KLSP	LASP = KLSP	LASP > KLSP
Skin	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
SensIT Vehicle1	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Shuttle1	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Poker2	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Poker1	LASP > KLSP	LASP > KLSP	LASP > KLSP	LASP > KLSP
Total	LASP > KLSP (32)　LASP = KLSP (8)　KLSP > LASP (0)			

考虑到有些实验数据集正负类样本的不平衡性较大,Acc指标并不能客观反映各个算法在这类数据集上的性能,本文同时在表4中给出了各个算法在这些数据集上的AUC、F-score和G-mean三个指标值. 由表4可以看出,在这3个指标上,LASP算法也明显取得了比KLSP算法更好的结果.

表 4 LASP和KLSP算法在类不平衡数据集上的性能比较

数据集	算法	诱导集样本个数											
		50			100			150			200		
		AUC	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean	AUC	F-score	G-mean
Pendigits	LASP	1.000	0.987	0.989	1.000	0.993	0.994	1.000	0.993	0.994	1.000	0.995	0.995
	KLSP	1.000	0.976	0.985	1.000	0.990	0.991	1.000	0.989	0.991	1.000	0.992	0.994
Mnist0	LASP	0.981	0.914	0.942	0.986	0.938	0.959	0.990	0.944	0.967	0.990	0.949	0.969
	KLSP	0.981	0.881	0.945	0.986	0.923	0.960	0.985	0.916	0.965	0.988	0.935	0.968
Skin	LASP	0.998	0.957	0.988	0.999	0.965	0.990	0.999	0.964	0.990	0.999	0.970	0.992
	KLSP	0.995	0.899	0.968	0.996	0.917	0.976	0.996	0.916	0.976	0.996	0.925	0.978
SensIT Vehicle1	LASP	0.892	0.719	0.807	0.909	0.740	0.822	0.920	0.757	0.835	0.927	0.767	0.843
	KLSP	0.870	0.715	0.815	0.863	0.718	0.817	0.859	0.716	0.817	0.861	0.708	0.816
Shuttle1	LASP	0.999	0.990	0.970	1.000	0.995	0.985	1.000	0.996	0.988	1.000	0.996	0.989
	KLSP	0.990	0.979	0.932	0.992	0.980	0.940	0.995	0.982	0.941	0.993	0.982	0.945

4 结 论

本文基于拉普拉斯方法建立了一种可以有效处理大规模数据的高斯过程分类算法. 在10个UCI 数据集上的实验结果表明,本文算法不仅计算时间较短,而且可以取得较高的预测精度. 当然,该算法也有需要改进的地方,例如:在训练时每一次都需要用到所有的训练数据,计算复杂度仍然偏高,下一步将考虑利用随机梯度法建立一种训练效率更高的算法;此外,考虑到目前基于交叉验证法选取协方差函数参数的策略不仅效率低,而且不能得到最优参数,如何利用模型的边缘似然自动选择协方差函数参数也是下一步需要考虑的问题.

参考文献(References)

[1] Rasmussen C, Williams K. Gaussian process for machine learning[M]. Cambridge: MIT Press, 2006: 33-77.

[2] Ranganathan A, Yang M, Ho J. Online sparse Gaussian process regression and its applications[J]. IEEE Trans on Image Processing, 2011, 20(2): 391-404.

[3] Kim H, Ghahramani Z. Bayesian Gaussian process classification with the EM-EP algorithm[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2006, 28(12): 1948-1959.

[4] Chu W, Sindhwani V, Ghahramani Z, et al. Relational learning with Gaussian processes[C]. Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2007, 19: 289-296.

[5] Lawrence N, Jordan M. Semi-supervised learning via Gaussian processes[C]. Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2004: 753-760.

[6] Guiver J, Snelson E. Learning to rank with softrank and gaussian processes[C]. Proc of the 31st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM Press, 2008: 259-266.

[7] Bonilla E, Chai K, Williams C. Multi-task Gaussian process prediction[C]. Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2007: 153-160.

[8] He J, Gu H, Wang Z. Bayesian multi-instance multi-label learning using Gaussian process prior[J]. Machine Learning, 2012, 88(1/2): 273-295.

[9] Titsias M. Variational learning of inducing variables in sparse Gaussian processes[C]. Proc of the 12th Int Conf on Artificial Intelligence and Statistics. Clearwater Beach: JMLR, 2009, 5: 567-574.

[10] Quiñonero-Candela J, Rasmussen C. A unifying view of sparse approximate Gaussian process regression[J]. J of Machine Learning Research, 2005, 6: 1939-1959.

[11] Lawrence N, Seeger M, Herbrich R. Fast sparse Gaussian process methods: The informative vector machine[C]. Proc of the 16th Annual Conf on Neural Information Processing Systems. Cambridge: MIT Press, 2003: 609-616.

[12] Naish-Guzman A, Holden S. The generalized FITC approximation[C]. Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2007: 1057-1064.

[13] Hensman J, Matthews A, Ghahramani Z. Scalable variational Gaussian process classification[C]. Proc of the 18th Int Conf on Artificial Intelligence and Statistics. San Diego: AISTATS, 2015: 351-360.

[14] Williams C, Barber D. Bayesian classification with Gaussian processes[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1998, 20(12): 1342–1351.

[15] Opper M, Archambeau C. The variational Gaussian approximation revisited[J]. Neural Computation, 2009, 21(3): 786-792.

[16] Chen M. Kmeans clustering[DB/OL]. (2009-07-01) [2016-04-12]. http://www.mathworks.com/matlabcentral/fileexch ange/24616-kmeans-clustering.

（责任编辑：李君玲）