第 52 卷 第 5 期 厦门大学学报(自然科学版) Vol. 52 No. 5
2013 年 9 月 Journal of Xiamen University (Natural Science) Sep. 2013
doi:10.6043/j.issn.0438-0479.2013.05.006

# 基于粒子群算法与图形处理器加速的支持向量机参数优化方法

毛耀宗[1],陈 珂[2],江 弋[1],邹 权[1]*
(1.厦门大学信息科学与技术学院,福建 厦门 361005;2.广东石油化工学院计算机科学与技术系,广东 茂名 525000)

**摘要**:支持向量机(support vector machine,SVM)的参数选择对其性能有着重要的影响,使用穷举法优化参数需要大量的计算时间.为快速寻找最优参数组合,利用粒子群算法(particle swarm optimization,PSO)收敛速度快、简单易行等特点,将 SVM 参数作为粒子的解决方案.并利用图形处理器(graphics processing unit,GPU)并行化处理能力计算每个参数的分类准确率,从而提升了在一定的搜索空间内寻找最佳参数组合的计算速度.对 UCI 数据进行实验,对比结果显示,该方法能快速有效地获取优化结果.

**关键词**:支持向量机;粒子群算法;图形处理器;参数寻优
**中图分类号**:TP 302.7 **文献标志码**:A **文章编号**:0438-0479(2013)05-0609-04

支持向量机(support vector machine,SVM)[1]是一种具有严密理论基础的计算机学习的新方法,在解决小样本问题、非线性及高维模式识别等实际应用中表现出许多特有的优势.在机器学习、模式识别、智能计算等领域已经成为热点技术,受到国内外研究者的广泛关注.理论研究与实践表明,SVM 的性能和惩罚因子 $C$ 以及核函数的参数有着紧密的联系,寻找到好的参数组合对于 SVM 的推广与应用能起到极大的推动作用.

一般寻找参数的方法是在一个二维坐标轴上列出一个网格[2],网格上的点表示所有参数可能的取值,然后穷尽网格上的点,并计算其预测准确率,从中选取最佳的参数组合.这种方法虽然能找到最优的参数组合,但是基于穷举法的寻找过程要求的计算时间相对较大,这在另一方面限制了 SVM 的性能.

粒子群算法(particle swarm optimization,PSO)[3-4]在一定的搜索空间中通过多次迭代找寻最优解,每个粒子在迭代的过程中不断地调整自身的位置(或者方向)和速度,从而尽可能地获取最佳的解决方案.当迭代结束或者寻优结果落在可接受区间内,就用此局部最优解来替代全局最优.这一过程避免了穷尽所有可能的情况,不但能获取满意的结果而且减小了程序的计算量.

本文在 PSO 算法寻找 SVM 最优参数组合的过程中,引入图形处理器(graphics processing unit,GPU)计算[5],用来加速计算过程,实验结果表明这种方法能获得满意的预测准确率并且降低了程序寻优的时间.

## 1 基本原理介绍

### 1.1 SVM 基本原理

SVM 的基本思想就是将低维的线性不可分问题通过核函数映射到高维空间,使得问题成为线性可分[6-7].

对于基本的分类问题,计算公式如下:

$$\begin{cases}\min \dfrac{1}{2}\|w\|^2, \\ \text{s.t. } y_i[<w,x_i>+b]-1\geqslant 0, \\ \qquad (i=1,2\cdots n),\ n \text{ 为样本数},\end{cases} \tag{1}$$

其中,$\|w\|$ 为求解的目标函数,其值要求越小越好,$x$ 为样本点,$y$ 为样本的分类标记.当在处理分类问题过程中出现了离群点时,通过引入松弛变量来放宽对样本点的分类要求,由于这些点不能达到精确分类,对于分类器来说是一种损失,体现在数学式(2)中:

收稿日期:2013-05-03
基金项目:国家自然科学基金项目(61001013,61102136);福建省自然科学基金项目(2011J05158,2010J01351)
*通信作者:zouquan@xmu.edu.cn

$$\begin{cases}\min \dfrac{1}{2}\| w \|^2 + C\sum\limits_{i=1}^{n}\xi_i, \\ \text{s. t.} \quad y_i[< w, x_i > + b] \geqslant 1 - \xi_i, \\ \quad i = 1, 2, \cdots, n, \end{cases} \tag{2}$$

其中,$n$ 为样本数,$C$ 为惩罚因子,与松弛变量 $\xi_i$ 搭配后用来决定离群点所带来的损失,当 $C$ 的数值越大表示样本点的影响力越大,对目标函数的损失也越大. 最极端的情况是当 $C$ 的取值为无限大时,只要出现一个离群点,那么问题就会无解.

核函数的作用是,当处理线性不可分样本的时候,将低维空间的输入映射到高维空间,计算出经过核函数变化后在高维空间里向量的内积值,使得线性不可分问题成为线性可分. 本文所用的径向基核函数(radial basis fuction,RBF)公式如下:

$$k(x_i, y_i) = \exp(-\gamma \| x_i - y_i \|^2), \tag{3}$$

其中 $\gamma$ 即为 Libsvm 软件所用的 $g$ 参数(gamma),此参数反应了支持向量的相关程度,当 $\gamma$ 较小时,向量之间的相关程度低;当 $\gamma$ 较大时,向量之间的相关程度就高. 选取合适的参数 $C$ 和 $\gamma$ 的会影响 SVM 的性能.

### 1.2 PSO 基本原理

PSO 算法在每轮迭代过程中不断调整粒子的运行速度与方向,最终寻找到满意的解决方案. 算法首先初始化个数为 $n$ 的粒子,假设粒子在 $d$ 维的搜索空间中搜寻目标,每个粒子的位置可以用向量 $X_i = [X_{i1}, X_{i2}, \cdots, X_{id}]$ 表示,通过一个适应值计算函数 $f(x)$ 获取每个粒子个体最优方案 $x_{ip\_best} = [p_{i1}, p_{i2}, \cdots, p_{id}]$ 以及全局最优方案 $x_{g\_best} = [g_1, g_2, \cdots, g_d]$,之后进入迭代过程,每个粒子根据式(4)调整自身当前时刻的位置和速度:

$$\begin{cases} V_{i(t+1)} = wV_{it} + C_1 \cdot \text{rand}_1 \cdot (X_{ip\_best} - X_{it}) + \\ \quad C_2 \cdot \text{rand}_2 \cdot (X_{g\_best} - X_{it}), \\ X_{i(t+1)} = X_{it} + V_{i(t+1)}, \end{cases} \tag{4}$$

其中,$V_{it}$,$V_{i(t+1)}$ 分别表示粒子 $i$ 在 $t$ 时刻与 $t+1$ 时刻的速度 $V_i = [V_{i1}, V_{i2}, \cdots, V_{id}]$,$X_{it}$,$X_{i(t+1)}$ 分别表示粒子 $i$ 在 $t$ 时刻与 $t+1$ 时刻的解决方案 $X_i = [X_{i1}, X_{i2}, \cdots, X_{id}]$,实数 $C_1$,$C_2$ 为学习因子,$\text{rand}_1$,$\text{rand}_2$ 为范围在(0,1)的随机数,学习因子与随机数的乘积用来表征粒子对个体最优方案和全局最优方案的学习程度,通常 $C_1 = C_2 = 2.0$,惯性权重 $w$ 用来控制粒子 $t$ 时刻的速度对 $t+1$ 时刻速度的影响,当 $w$ 较大时算法的全局搜索能力较强,当 $w$ 较小时算法的局部搜索能力较强[8]. 在迭代过程中通过调整 $w$ 的大小平衡算法的搜索能力,通常采用的是公式(5)所示的线性迭代:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter}_{\text{current}}, \tag{5}$$

其中,$w_{\max}$,$w_{\min}$ 分别为惯性权重的最大值与最小值,$\text{iter}_{\text{current}}$,$\text{iter}_{\max}$ 分别为算法当前迭代次数与最大迭代次数. 随着迭代过程的进行,$w$ 的值由大到小变化,粒子由注重全局搜索能力逐渐转变为注重局部搜索能力.

在每轮迭代结束选取一部分粒子再次随机分配解决方案,这样能防止粒子群算法陷入局部最优,影响算法的性能. 由于 PSO 算法限定在一个有限的区域而不是对整个搜索空间进行计算,所以算法计算出的局部最优方案不一定是全局最优. 本文算法的适应值函数由程序调用 Libsvm 软件对测试集和预测集进行计算,并返回准确率来评价解决方案的效果.

### 1.3 GPU 计算简介

GPU 计算是指把 GPU 作为计算加速器或者协作处理器来承担密集型数据和并行数据的计算,从而为那些在 CPU 上运行的应用程序加速.

GPU 采用了大量的执行单元,这些执行单元不但能共享存储空间并且能进行并行处理,而不像 CPU 那样的单线程处理. 统一计算设备架构(compute unified device architecture, CUDA)[9] 是英伟达(NVIDIA)为自己的 GPU 编写的一套编译器及库文件,用于管理 GPU 上可以并行执行的多线程计算设备. 在主机端(host)执行 CPU 上的运算,当程序运行到需要处理大量数据且能并行执行的部分,主程序通过调用 kernel 函数进行 GPU 计算,在 GPU 部分尽量开发线程级并行,这些线程能够在硬件中被动态的调度和执行,当计算完毕后,再从设备端(device)把结果返回 CPU 继续执行.

## 2 基于 PSO 算法与 GPU 加速的 SVM 参数寻优算法

用一个数组将两个参数 $c$ 和 $g$ 存储在每个粒子对象当中,用来代表问题的解决方案,对应的 $c$ 和 $g$ 调整速度也以数组的形式存储,此外通过适应值函数计算出的个体最优解以及全局最优解也存储在粒子对象当中. 计算适应值的过程就是进行预测准确率的过程,通过调用 GPU 来训练样本并返回预测准确率. 在之后的迭代过程中,粒子根据个体最优解以及全局最优解调整自己的位置,获得新的 $c$ 和 $g$ 参数值并进行计算,直到迭代次数达到预先设定的最大值,程序返回优化值并结束退出.

本文采用交叉检验方法来获取预测准确率,$K$ 重交叉检验($K$-fold cross validation)[10]将训练数据均分成 $K$ 份,每份轮流作一次预测集,其余 $K$-1 份作为训练集,这样总共可以获取 $K$ 个训练模型以及 $K$ 次预测的准确率,最终返回这 $K$ 次分类准确率的平均值作为 $K$ 重交叉检验的性能指标.

算法具体步骤如下:

1) 设定粒子的数量 number_of_particle;学习因子 $C1$,$C2$;最大迭代次数 iter;搜索空间的范围 $x$_max,$x$_min;粒子更新速度的范围 $v$_max,$v$_min;

2) 初始化粒子群的位置 $x\_cg[c][g]$ 以及更新速度 $v\_cg[c][g]$;

3) 程序根据获取的 $c$,$g$ 值调用 Libsvm 对训练集进行训练,并计算模型预测的准确率;

4) 保存每个粒子的个体最优方案以及粒子群的全局最优方案;

5) 变量 count 计算当前迭代次数,如果 count>iter 执行步骤 11),否则利用式(5)计算此次迭代权重惯性 $w$ 的值;

6) 根据式(4)更新粒子的速度以及位置;

7) if(粒子的速度大于 $v$_max) $v_cg=v$_max;

8) if(粒子的速度小于 $v$_min) $v_cg=v$_min;

9) if(粒子的位置超出搜索空间) 重新在搜索空间内随机设定粒子的位置;

10) 返回步骤 3);

11) 算法结束.

## 3　实验数据分析

本实验采用 UCI 中的数据,所有实验都是在安装在 CUDA(compute unified device architecture)运行环境的 64 位 Linux 操作系统下运行,程序用 Java 语言编写,并且为随机数设定种子 random(16)方便实验结果的重现.为了便于实验结果的数据比对,还添加一组 Libsvm 自带的 Grid.py 程序的实验结果作为对比.

对 UCI 数据集中 abalone_scale.txt 数据进行分类实验,采用 5 重交叉检验,实验中的参数设定如下:粒子数=20,迭代次数 iter=10,学习因子 $c1=c2=2.0$,搜索空间范围为二维坐标轴上 $x=(-10,-9,-8,\cdots,8,9,10)$,$y=(-10,-9,-8,\cdots,8,9,10)$ 上的整数点,总共有 $20\times20=400$ 个可行解,粒子最大更新速度 $v$_max=10,最小更新速度 $v$_min=−10,最大惯性权重 $w$_max=1.2,最小惯性权重 $w$_min=0.2,表 1 记录了网格搜索算法、PSO 算法、经 GPU 加速的 PSO 算法的实验数据.

表 1　3 种算法在 UCI 数据集上作 5 重交叉检验结果比对

Tab.1　The five corss validation result of three algorithms on UCI data set

| 算法 | 准确率/% | $t$/s | $c$ | $g$ |
|---|---|---|---|---|
| Grid.py | 27.8190 | 3859 | 512 | 0.125 |
| PSO | 27.9148 | 2646 | 64 | 0.500 |
| PSO+GPU | 27.9148 | 2273 | 64 | 0.500 |

把实验结果 $c=512$,$g=0.125$ 用 Libsvm 进行 5 重交叉检验,确实获得27.819 0%的准确率,把 $c=64$,$g=0.5$ 用 Libsvm 同样进行 5 重交叉检验结果为 27.914 8%,由于数据本身的问题,该分类的准确率不能达到很高的标准.

从表 1 可以看出在准确率基本不变的情况下,相同的训练数据 3 种算法运行的时间不同,基于网格的参数搜索所需计算时间最长,而基于 PSO 优化算法的计算时间有所缩短,而经过 PSO 优化以及 GPU 加速的算法所用时间最短.

对于 UCI 数据集中 pendigits.txt,当不采用 5 重交叉检验而是直接用 UCI 提供的测试集 pendigits.txt 做检验来获取最佳的 $c$,$g$ 参数组合,实验结果如表 2 所示.

表 2　3 种算法对 pendigits.txt 的预测结果比对

Tab.2　The prediction result of three algorithms on pendigits.txt

| 算法 | 准确率/% | $t$/s | $c$ | $g$ |
|---|---|---|---|---|
| Grid.py | 98.0560 | 3524 | 128 | $3.05175\times10^{-5}$ |
| PSO | 95.5117 | 3007 | 2.0 | $9.765\times10^{-4}$ |
| PSO+GPU | 95.5117 | 2522 | 2.0 | $9.765\times10^{-4}$ |

由表 2 数据显示,PSO 与 PSO+GPU 算法的计算速度相比 Grid.py 方法均有所提升,而在准确率方面虽然已经达到了95.5517%,但是相比 Grid.py 方法获取的 98.0560%低,这是因为 PSO 算法只能在一定区域内搜索,而不能获取全局最优解,因此在准确率方面可能达不到 Grid.py 方法的水平.

PSO 优化方法和 PSO+GPU 优化方法在两个数据集上相对于 Grid 方法的加速比如表 3 所示.

从表 3 的数据看出,PSO+GPU 在两个数据集上的加速比均高于 PSO 方法,由此可见,在 PSO 算法的基础上经过 GPU 加速确实能够提高程序的计算速度.

表 3 相对于 Grid. py 方法的计算时间加速比

Tab. 3 The speed-up ratio of the experiment compares with the Grid. py method

| 算法(算法加速比) | 鲍鱼 | 手写体数字 |
|---|---|---|
| PSO(s) | 2646 | 3007 |
| PSO 加速比 | 1. 458 | 1. 171 |
| PSO+GPU(s) | 2273 | 2522 |
| PSO+GPU 加速比 | 1. 697 | 1. 397 |
| Grid. py(s) | 2859 | 3524 |

注:Grid 在鲍鱼、手写体数字上测试时间分别为 2859,3524 s.

## 4 结 论

SVM 的参数选择对于 SVM 的性能至关重要,但是其参数的选取又必须经过不断的试探和舍取,找到一个行之有效的参数选择方法将有助于 SVM 的推广应用. 本文利用粒子群优化算法,在一定的搜索空间内尽可能获取满意解,并利用 GPU 加速算法的计算,提高了算法的收敛速度. 通过 UCI 数据进行测试,该方法能够快速有效的获取最优参数组合.

参考文献:

[1] Cortes C, Vapnik V. Support vector network[J]. Machine Learning, 1995, 20: 273-297.

[2] 王佳,徐蔚鸿. 基于动量粒子群的混合核 SVM 参数优化方法[J]. 计算机应用, 2011, 31(2): 501-503.

[3] Kennedy J, Eberhart R C. Particle swarm optim ization[J]. Proc IEEE Int Conf Neural Networks, 1995, 4: 1942-1948.

[4] Eberhart R, Kennedy J. A new optim izer using particle swarm theory[C]// Proc of the 16th International Symposium on MicroMachine and Human Science. Nagoya, Japan: IEEE, 1995: 39-43.

[5] Liao Q, Wang J B, Webster Y, et al. GPU accelerated support vector machines for mining high-throughput screening data[J]. Journal of Chemical Information and Modeling, 2009, 49(12): 2718-2725.

[6] 邓乃扬,田英杰. 数据挖掘中的新方法——支持向量机[M]. 北京:科学出版社, 2004

[7] 崔伟东,周志华,李星. 支持向量机研究[J]. 计算机工程与应用, 2011, 37(1): 58-61.

[8] 李太白. 基于混沌粒子群的 SVM 参数优化算法[J]. 重庆文理学院学报:自然科学版, 2011, 30(4): 81-84.

[9] 科可(美),胡文美. 大规模并行处理器编程实战[M]. 北京:清华大学出版社, 2010.

[10] 邵信光,杨慧中,陈刚. 基于粒子群优化算法的支持向量机参数选择及其应用[J]. 控制理论与应用, 2006, 23(5): 740-743.

# Parameter Optimization of SVM Based on Particle Swarm Optimization Algorithm and GPU Acceleration

MAO Yao-zong[1], CHEN Ke[2], JIANG Yi[1], ZOU Quan[1]*

(1. School of Information Science and Engineering, Xiamen University, Xiamen 361005, China;
2. Department of Computer Science and Technology, Guangdong University of Petrochemical Technology, Maoming 525000, China)

**Abstract**: The parameter selection of the support vector machine (SVM) has significant impact on its performance. Brute-force method for finding the optimal parameters is time consuming. In this paper, we are making use of particle swarm optimization (PSO)′s character of fast convergence speed and could be implemented easily, adding the SVM parameters as the solution of the particles. And exploit the computing capability of the graphics processing unit(GPU) to calculate the classification accuracy of each parameter, thus enhancing the computing speed for finding the best parameter combination in a constraint solutions space. The comparison results on UCI data show that this method can obtain the optimal results quickly and efficiently.

**Key words**: support vector machine(SVM); particle swarm optimization(PSO); graphic processing unit(GPU); parameter optimization