文章编号:1004-9037(2012)01-0085-06

基于CUDA 的GMM 模型快速训练方法

吴 奎 宋 彦 戴礼荣

(中国科学技术大学电子工程与信息科学系,合肥,230027)

摘要:由于能够很好地近似描述任何分布,混合高斯模型(GMM)在模式在识别领域得到了广泛的应用。GMM 模型参数通常使用迭代的期望最大化(EM)算法训练获得,当训练数据量非常庞大及模型混合数很大时,需要花费很长的训练时间。NVIDIA 公司推出的统一计算设备架构(Computed unified device architecture, CUDA)技术通过在图形处理单元(GPU)并发执行多个线程能够实现大规模并行快速计算。本文提出一种基于CUDA,适用于特大数据量的GMM 模型快速训练方法,包括用于模型初始化的K-means 算法的快速实现方法,以及用于模型参数估计的EM 算法的快速实现方法。文中还将这种训练方法应用到语种GMM 模型训练中。实验结果表明,与Intel DualCore Pentium Ⅳ 3.0 GHz CPU 的一个单核相比,在NVIDIA GTS250 GPU 上语种GMM 模型训练速度提高了26 倍左右。

关键词:混合高斯模型;语种识别;图形处理单元;统一计算设备架构

中图分类号:TP391 **文献标识码**:A

CUDA-Based Fast GMM Model Training Method and Its Application

Wu Kui, Song Yan, Dai Lirong

(Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, 230027, China)

Abstract: Due to its good property to provide an approximation to any distribution, Gaussian mixture model (GMM) is widely applied in the field of pattern recognition. Usually, the iterative expectation-maximization (EM) algorithm is applied to GMM parameter estimation. The computational complexity at model training procedure can become extremely high when large amounts of training data and large mixture number are engaged. The computed unified device architecture (CUDA) technology provided by NVIDIA Corporation can perform fast parallel computation by running thousands of threads simultaneously on graphic processing unit (GPU). A fast GMM model training implementation using CUDA is presented, which is especially applicable to large amounts of training data. The fast training implementation contains two parts, i.e., the K-means algorithm for model initialization and the EM algorithm for parameter estimation. Furthermore, the fast training method is applied to language GMMs training. Experimental results show that language model training using GPU is about 26 times faster on NVIDIA GTS250 than the traditional implementation on one of the single core of Intel DualCore Pentium Ⅳ 3.0 GHz CPU.

Key words: Caussian mixture model (GMM); language identification; graphic processing unit (GPU); computed unified device architecture (CUDA)

引 言

由于能够很好地近似描述任何分布,高斯混合模型(Gaussian mixture model, GMM)在模式识别领域得到了广泛的应用。GMM 模型参数通常使用迭代的期望最大化(Expectation maximization, EM)算法[1]训练获得。EM 算法是一个迭代算法,

收稿日期:2011-01-06;修订日期:2011-06-27

需要对模型初始化,一般采用 K-means 算法实现 EM 算法的初始化。当训练数据量非常庞大及模型混合数很大时,模型训练需要花费很长的时间。例如,在 GMM-UBM(Gaussian mixture model-universe background model)模型的语种识别系统[2]中,语种训练样本数非常庞大(如:NIST LRE2007 包含 14 个大语种,对应的移位差分倒谱(Shifted Delta Cepstra, SDC)[2]训练矢量特征总数为 68 281 155),模型混合高斯数多(一般为 2 048)计算量巨大。如果用一个CPU 的单核训练模型,那么训练时间就不得不成为一个需要考虑的因素,而并行化处理是加快训练一个有效途径。

可编程图形处理器单元(Graphic processing unit, GPU)的出现,开辟了快速并行处理的新途径。目前,GPU 计算能力和发展速度都超过了传统 CPU。GPU 具有大量并行处理的流处理器(Stream Processers, SP),适合计算密集型的应用。在 NVIDIA 公司推出的统一计算设备架构(Computed unified device architecture, CUDA)编程环境中,使用简单且易实现的扩展C 语言就能编写GPU 并行程序,大大提高了GPU 的可编程性。Kumar N 等[3]用CUDA 实现了GMM 模型训练的EM 算法,然而这种实现方法需要计算并存放多个阶数与高斯混合数、特征维数和训练样本数成正比的矩阵。由于显存容量有限,高斯混合数、特征维数和训练样本数的大小受到限制,不能满足训练样本数庞大的情形。K-means 算法在GPU 上已有多种实现方法[4-5],尽管如文献[5]中所报道的方法对类别数和维数没有限制,但同样受到训练样本数的限制。

本文采取分批处理策略,提出了一种改进的基于CUDA 的GMM 模型快速训练方法,包括用于模型初始化的 K-means 算法和模型参数估计的 EM 算法的快速实现方法。这种GMM 模型快速训练方法不受训练样本数的限制,且高斯混合数取值范围可以很大(2 到 4 096),能够满足大部分实际要求,如语种 GMM 模型训练。在实现过程中,简化了文献[3]中繁琐的协方差矩阵更新过程。实验结果表明,在训练语种 GMM 模型时,使用这种方法,模型训练速度是传统CPU 单核的 26 倍左右。

1 统一计算设备架构

NVIDIA 公司的CUDA 是一种新型的硬件和软件相结合的架构。CUDA 将GPU 视为并行计算设备,在计算时自动完成资源的分配和管理,而无需映射到复杂的图形API 中,并且使用简单的扩展 C 即可编写GPU 并行程序,极大地提高了GPU 的可用性和编程性。线程是CUDA 中的最小的执行单位,并以线程块的形式组织起来,执行相同的指令,但处理不同的数据。执行相同程序的线程块组成一个网格。所有线程执行的相同的程序称为一个 kernel,一个 kernel 对应一个网格。

线程可以访问多种形式的存储空间。每个线程都有自己的快速读写的本地寄存器和本地存储单元。同一个线程块中的线程可以访问属于该线程块的 16KB 共享内存。线程访问共享内存没有产生 Bank conflict[7]时,可达到与寄存器同样的速度。线程还可以访问容量较大的全局内存,因为没有缓存,读取全局内存的延时很大。CUDA 提供的 Access coalescing 机制[7]可以加快对全局内存的访问。寄存器和共享内存是GPU 上的有限资源,需要合理的使用以让尽可能多的块和线程同时工作。

CUDA 可以直接集成在 VS2005 或者其他的编译器中。GPU 称为设备,CPU 称为宿主。宿主主要负责启动kernel 和传输数据,设备负责计算。设备上的 kernel 的执行和宿主是异步的,这样 GPU 在工作时,CPU 还可以做其他任务,当然 GPU 和 CPU 是可以同步的。文献[7]详细介绍了CUDA。

2 GMM 模型训练在GPU 上的实现

在EM 和K-means 算法中,训练样本在计算过程中是相互独立,适合GPU 的并行计算。由于EM 算法较为复杂,而 K-means 算法相对简单,所以着重介绍EM 算法的快速实现方法,然后再简要地介绍 K-means 算法的快速实现方法。

2.1 EM 算法的矩阵表示

首先用矩阵的形式描述EM 算法,以更好的介绍EM 算法在GPU 上的实现过程。混合数为M 的 GMM 模型的密度函数表示为

$$p(\boldsymbol{x}|\Theta)=\sum_{i=1}^{i=M}\alpha_i p_i(\boldsymbol{x}|\boldsymbol{\theta}_i)$$

其中$\Theta=(\alpha_1,\alpha_2,\cdots,\alpha_M,\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\cdots,\boldsymbol{\theta}_M)$是参数集合,$\boldsymbol{\theta}_i=(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$,$\alpha_i,\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i$ 分别是第 i 个高斯分量的权重、均值向量和协方差矩阵,且 $\sum_{i=1}^{M}\alpha_i=1$,第 i 个高斯的密度函数为

$$p(\boldsymbol{x}|\boldsymbol{\theta}_i)=\frac{1}{2\pi^{\frac{D}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}}\mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)}$$

将所有训练样本集 X 表示成矩阵形式 $\boldsymbol{X}=[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N]^{\mathrm{T}}$,样本数为 N,维数为 D,其中 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{iD})^{\mathrm{T}}$,$i=1,2,\cdots,N$。

上述形式的GMM模型的EM算法迭代公式[6]如下

$$\alpha_l^{\text{new}}=\frac{1}{N}\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)\tag{1}$$

$$\mu_l^{\text{new}}=\frac{\sum_{i=1}^{N}\boldsymbol{x}_i p(l\mid\boldsymbol{x}_i,\Theta^g)}{\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)}\tag{2}$$

$$\boldsymbol{\Sigma}_l^{\text{new}}=\frac{\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)(\boldsymbol{x}_i-\boldsymbol{u}_i)^{\mathrm{T}}(\boldsymbol{x}_i-\boldsymbol{u}_i)}{\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)}\tag{3}$$

式中 $\Theta^g=(\alpha_1^g,\cdots,\alpha_M^g,\theta_1^g,\cdots,\theta_M^g)$ 是已知的当前模型参数估计值,$p(l\mid\boldsymbol{x}_i,\Theta^g)=\dfrac{\alpha_l^g\,p(\boldsymbol{x}_i\mid\theta_l^g)}{\sum_{k=1}^{M}\alpha_k^g\,p(\boldsymbol{x}_i\mid\theta_k^g)}$,称为高斯分量 l 对训练样本 $\boldsymbol{x}_i$ 的占有率。实际应用中,协方差矩阵通常取对角阵,所以式(3)又可写为

$$\boldsymbol{\Sigma}_l^{\text{new}}=\operatorname{diag}\left(\frac{\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}}{\sum_{i=1}^{N}p(l\mid\boldsymbol{x}_i,\Theta^g)}-\mu_l^{\text{new}}(\mu_l^{\text{new}})^{\mathrm{T}}\right)\tag{4}$$

将 M 个高斯分量的参数表示成矩阵形式权重矩阵 $\boldsymbol{W}=(\alpha_1,\alpha_2,\cdots\alpha_M)$;均值矩阵 $\mathbf{MEAN}=[\mathbf{u}_1{}^{\mathrm{T}},\cdots\boldsymbol{u}_M{}^{\mathrm{T}}]^{\mathrm{T}}$;协方差矩阵 $\boldsymbol{\Sigma}=[\boldsymbol{\Sigma}_1{}^{\mathrm{T}},\cdots,\boldsymbol{\Sigma}_M{}^{\mathrm{T}}]^{\mathrm{T}}$,$\Sigma_i=(\sigma_{i1}^2,\sigma_{i2}^2\cdots,\sigma_{iD}^2)$ 是第 i 个高斯分量的对角协方差矩阵中对角线上的元素构成的向量。估计模型的参数,也就是去估计上面三个矩阵。定义三个统计累积量矩阵:

权重累积量矩阵 $\boldsymbol{W}_{\text{acc}}=(\alpha_{1,\text{acc}},\alpha_{2,\text{acc}},\cdots,\alpha_{M,\text{acc}})$,均值累积量矩阵 $\mathbf{MEAN}_{\text{acc}}=[\boldsymbol{u}_{1,\text{acc}}{}^{\mathrm{T}},\cdots,\boldsymbol{u}_{M,\text{acc}}{}^{\mathrm{T}}]^{\mathrm{T}}$,方差累积量矩阵 $\boldsymbol{\Sigma}_{\text{acc}}=[\boldsymbol{\Sigma}_{1,\text{acc}}{}^{\mathrm{T}},\cdots,\boldsymbol{\Sigma}_{M,\text{acc}}{}^{\mathrm{T}}]^{\mathrm{T}}$ 及 $N\times M$ 阶占有率矩阵 $\boldsymbol{O}$,其中 $a_{l,\text{acc}}=\sum_{n=1}^{N}p(l\mid\boldsymbol{x}_n,\Theta^g)$,$\mu_{l,\text{acc}}=(\mu_{l1,acc},\cdots,\mu_{lD,acc})=\sum_{n=1}^{N}p(l\mid\boldsymbol{x}_n,\Theta^g)\boldsymbol{x}_n$

$\Sigma_{l,acc}=(\sigma_{l1,acc}^2,\cdots,\sigma_{lD,acc}^2)=\sum_{n=1}^{N}p(l\mid\boldsymbol{x}_n,\Theta^g)\boldsymbol{x}_n{}^2$,$\boldsymbol{x}_n{}^2=(x_{n1}{}^2,\cdots x_{nD}{}^2)^{\mathrm{T}}$,$O_{(n,l)}=p(l\mid\boldsymbol{x}_n,\Theta^g)$,$n=1,\cdots,N$;$l=1,\cdots,M$ 写成矩阵形式

$$\boldsymbol{W}_{\text{acc}}=(1,1,\cdots,1)\cdot\boldsymbol{O}\tag{5}$$

$$\mathbf{MEAN}_{\text{acc}}=\boldsymbol{O}^{\mathrm{T}}\boldsymbol{X}\tag{6}$$

$$\boldsymbol{\Sigma}_{\text{acc}}=\boldsymbol{O}^{\mathrm{T}}\boldsymbol{X}^2\tag{7}$$

式中 $\boldsymbol{X}^2=(\boldsymbol{x}_1{}^2,\cdots,\boldsymbol{x}_N{}^2)^{\mathrm{T}}$

根据式(1,2,4)可以得到参数矩阵更新公式

$$\alpha_l^{\text{new}}=\frac{\alpha_{l,\text{acc}}}{N}\tag{8}$$

$$\boldsymbol{u}_l^{\text{new}}=\frac{\boldsymbol{u}_{l,\text{acc}}}{N\alpha_l^{\text{new}}}\tag{9}$$

$$\boldsymbol{\Sigma}_l^{\text{new}}=\frac{\boldsymbol{\Sigma}_{l,\text{acc}}}{N\alpha_l^{\text{new}}}-(\boldsymbol{u}_l^{\text{new}})^2\tag{10}$$

式中 $(\boldsymbol{u}_l^{\text{new}})^2=((u_{l1}^{\text{new}})^2,\cdots,(u_{lD}^{\text{new}})^2)$,$l=1,\cdots,M$。这里在更新协方差矩阵时,充分考虑了协方差是对角阵这一条件。这样,EM算法的过程可以表示式(5-6)的矩阵运算及式(8-10)的除法运算。

2.2　EM算法在GPU上的实现

本文在GPU上实现EM算法的基本流程基于式(5-10),先计算 $\boldsymbol{O}$,然后依次进行计算,其中关键是计算 $\boldsymbol{O}$。权重和均值的更新与[3]相似,但[3]在更新协方差矩阵 $\boldsymbol{\Sigma}$ 时需要计算多个阶数较大的矩阵,过程繁琐,而在本文的方法中,只需计算 $\boldsymbol{\Sigma}_{\text{acc}}$ 后按式(10)更新 $\boldsymbol{\Sigma}$,简单明了。

计算 $\boldsymbol{O}$ 时,要对每一个训练样本 x_i 计算占有率 $p(l\mid\boldsymbol{x}_i,\Theta^g)$,$l=1,2,\cdots,M$。文献[3]中把每一个这样的值存放在GPU的显存中,也就是存储一个 $N\times M$ 阶矩阵。那么,N 和 M 的大小都要受到显存容量的限制。例如GTS250型号的GPU显存有512MB,若 M 取2 048,每个数据用32位浮点数表示,那么 N 最多只能取65 536,显然这不适用于训练样本数为几千万的情形。若将几千万个训练样本分成若干等份,每份样本数为 n,每次GPU只处理一份,$\boldsymbol{O}$ 的阶数减小为 $n\times M$,选取适当的 n(根据显存大小及 M 的值确定,实验表明 n 取16的倍数,GPU计算效率最高)就可以消除 N 的限制,并且 M 可以取得很大(2到4 096),这就是本文采取的分批处理策略。因为采取分批处理,所以3个累积量矩阵在每一批训练样本处理完之后累加一次,公式如下

$$\boldsymbol{W}_{\text{acc}}^{k}=\boldsymbol{W}_{\text{acc}}^{k-1}+(1,1,\cdots,1)\cdot\boldsymbol{O}_k\tag{11}$$

$$\mathbf{MEAN}_{\text{acc}}^{k}=\mathbf{MEAN}_{\text{acc}}^{k-1}+\boldsymbol{O}_k^{\mathrm{T}}\boldsymbol{X}_k\tag{12}$$

$$\boldsymbol{\Sigma}_{\text{acc}}^{k}=\boldsymbol{\Sigma}_{\text{acc}}^{k-1}+\boldsymbol{O}_k^{\mathrm{T}}\boldsymbol{X}_k^2\tag{13}$$

式中 $\boldsymbol{X}_k=(\boldsymbol{x}_{(k-1)n+1},\boldsymbol{x}_{(k-1)n+2},\cdots,\boldsymbol{x}_{kn})^{\mathrm{T}}$,$\boldsymbol{O}_k$ 分别是第 k 批训练样本矩阵及其占有率矩阵,$\boldsymbol{X}_k^2=(\boldsymbol{x}_{(k-1)n+1}^2,\boldsymbol{x}_{(k-1)n+2}^2,\cdots,\boldsymbol{x}_{kn}^2)^{\mathrm{T}}$。基于上面的分析,在GPU上实现适GMM模型训练的EM算法不是很复杂。设每批训练样本数为 n(本文取32 768)。定义5个主要的顺序执行的kernel:

(1)kernel 1 对第k批训练样本,计算$n\times M$阶占有率矩阵$\boldsymbol{O}_k$;

在GPU上实现时,每个线程只处理一个样本,即一个线程只计算$\boldsymbol{O}$矩阵中的一行,需要n个线程。假设每个线程块有B个线程(B一般取16的倍数,本文取256),那么需要$\frac{n}{B}$个线程块。M次循环计算得到整个矩阵,每次循环计算该批所有训练样本对某个高斯的占有率。

(2)kernel 2 计算$\boldsymbol{W}_{\mathrm{acc}}^k$;

(3)kernel 3 计算$\mathbf{MEAN}_{\mathrm{acc}}^k$

(4)kernel 4 计算$\boldsymbol{\Sigma}_{\mathrm{acc}}^k$;

(5)kernel 5 更新参数矩阵。

步骤(2~4)都是典型的矩阵线性运算,可以通过CUDA的CUBLAS库中高效的矩阵函数CublasSegmm实现。在实际模型训练中,为防止无限次的迭代,需要预先设定模型似然值L变化量改进门限δ和最大迭代次数K,如果L变化量小于δ或迭代次数大于K,算法结束,将最后的模型参数送给CPU,否则,继续下一次迭代。

2.3 K-means 算法在GPU上的实现

文献[5]中介绍了一种很好的在GPU上实现K-means算法的方法,它的基本方法是:CPU一次性将所有训练样本和初始类中心加载到GPU显存中去,GPU的每个线程计算一个样本或多个样本的类别号传送给CPU。CPU根据类别号更新类中心,并将新的类中心传给GPU,继续下一次迭代。

同样,由于GPU显存大小的限制,这种方法也只能处理较少的训练样本。因此,同上述EM算法的实现一样,也必须采用分批处理的策略。GPU每次只处理n个训练样本,每个线程处理一个样本,并将类别号传给CPU,计算类别号的方法与文献[5]相同。CPU根据每批训练样本的类别号更新各类的权重、均值和方差累积量,这与EM算法的实现很相似,最后将新的类中心传给GPU,继续下一次迭代。和EM算法一样,K-means算法也需预先设定类内距离准则函数值J_e变化量改进门限δ及最大迭代次数K。

2.4 数据存取格式和程序优化

存储器访问方式很大程度上决定了GPU的计算效率,例如,以Coalescing和无Bank conflict的方式分别访问显存和共享内存能最大化利用存储器带宽。因此,在GPU上实现任何算法时,必须考虑这些因素。

在EM和K-means算法实现中,训练样本和模型参数都存放在容量较大的显存里。为了能让线程读取训练样本实现Coalescing,以达到很快的访问速度,训练样本矩阵须按列存取,这在文献[3]有详细介绍。EM算法中,计算$\boldsymbol{O}$矩阵时,同一个Blocks里所有的线程需要同时读取当前高斯分量的参数(权重、均值向量、协方差矩阵)计算加权概率密度值$\alpha_l^g p(\boldsymbol{x}_i|\boldsymbol{\theta}_l^g)$,会在显存中产生访问冲突,线程须排队读取,严重降低线程的计算效率。由于共享内存的广播机制[7],本文在实现时,先用一部分线程将当前高斯分量的参数读入到共享内存中去,线程在同时访问这些共享内存时就不会产生Bank conflict,可同时获得参数,大大减少读取时间,从而提到了线程的计算效率。虽然共享内存大小只有16 KB,要将当前高斯分量的参数都读入到共享内存中,训练样本的维数必然受到限制。若每个数据用32位浮点数表示,那么维数最大可取2 047,能够满足大多数实际应用(本文使用的语种的SDC特征矢量的维数为56),可以说这种限制可不予考虑。$\boldsymbol{O}$矩阵也按列存放,这样线程在写$\boldsymbol{O}$矩阵时能实现coalescing。在K-means算法中,也先将当前类中心读入到共享内存中。

GPU的kernel执行与CPU是异步的,CPU在GPU计算期间里可以处理其他任务,例如可以读取存放在硬盘中的下一批训练样本数据。

分批处理的训练样本数n的选择需要认真考虑。它受到显存大小、高斯混合数、维数等的限制。n太大,显存不够;太小会导致许多线程空闲。在本文的两个算法中n都取16的倍数,GPU计算效率最高。用EM算法训练2 048混合高斯数、56维的GMM-UBM模型时,本文取n为32 768(显存为512 MB)。K-means算法中,因为没有阶数很大的矩阵,n可取大一点,本文取65 536。当总训练样本数不是n的倍数时,补充一些样本,并加以特殊的标识,使最后一批训练样本数变为n。GPU在计算时直接丢弃这些补充的样本。这样所有的矩阵和向量所需要的显存空间只需分配一次。

对于EM算法,在计算占有率矩阵$\boldsymbol{O}$时,必须要计算

$$\alpha_l^g\, p(\boldsymbol{x}_i|\boldsymbol{\theta}_l^g) = \alpha_l^g \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{\sigma_{l1}^{g\,2}\cdots\sigma_{lD}^{g\,2}}}\exp\left(-\frac{1}{2}\sum_{k=1}^{D}\frac{(x_{ik}-\mu_{lk}^g)^2}{\sigma_{lk}^{g\,2}}\right),$$

这个值一般很小,为了保证计算精度,求其对数值,即

$$\log[\alpha_l^g\, p(\boldsymbol{x}_i|\boldsymbol{\theta}_l^g)] = \log\alpha_l^g - \log\sqrt{\sigma_{l1}^{g\,2}\cdots\sigma_{lD}^{g\,2}} - D\log\sqrt{2\pi} - \exp\left(-\frac{1}{2}\sum_{k=1}^{D}\frac{(x_{ik}-\mu_{lk}^g)^2}{\sigma_{lk}^{g\,2}}\right)$$

在每次迭代中,上式右边前三项构成的表达式的值不变,可事先计算好放在显存中。计算$\boldsymbol{O}$矩阵时,将其读到共享内存中去,然后再运算。上式右边最后一项有D次除法,可将所有的方差事先取倒数,变除法为乘法,减少计算量。实验表明这种方法比直接计算要快1.5倍。本文要与之比较的CPU实现的EM算法也采用这种方法。由于GPU的计算精度有限,在K-means算法中,本文中利用Kahan求和公式[8]计算距离提高运算精度。

3　实验结果和分析

所有的测试都基于NIST07语种数据库(14个大语种,22个方言),采用56维的SDC特征,特征提取时使用了声道长度规整(Vocal tract length normalization,VTLN)[9]和因子分析(Factor analysis,FA)[10]技术,总训练特征矢量数为68 281 155,模型采用2 048 Gs,所有数据都用32 bit的单精度浮点数表示。EM和K-means算法最大迭代次数K设为25,改进门限δ设为0.000 1。

CPU实现的EM和K-means算法用C编写,在VS2005中编译,并在编译时使用VS2005自带的SSE2浮点运算扩展指令。GPU实现代码在CUDA Toolkit2.2中编译。PC机配置Intel DualCore 3.0 GHz Pentium Ⅳ CPU,512 MB内存和GTS250型号的NVIDIA显卡(128个SP,512 MB显存)。

表1比较了在1 449 920和68 281 155个训练样本时,K-means和EM算法一次迭代时的GPU和CPU单核运算时间。为了验证GPU运算结果的准确性,在GPU和CPU运算时,K-means算法的初始类中心相同,EM算法的初值模型也相同。对于这两个算法,GPU相对CPU提升的速度均为26倍左右,且不依赖训练样本数,这是因为在GPU上实现这两个算法时,对训练样本都进行分批处理,总训练样本数不影响GPU的性能。可以推测,如果进行完整的语种GMM训练(先K-means算法训练UBM初值模型,然后EM算法训练UBM模型及各语种GMM模型)GPU运算速度也应该是CPU单核的26倍左右。GPU和CPU运算的结果之间的误差也很小(以CPU为正确结果),68 281 155个训练样本时,在EM算法中,权重、均值和方差的最大相对误差只有0.000 2%。在K-means中,这些误差最大也只有0.002 2%左右,造成误差的原因是GPU只支持单精度运算,计算精度有限。

表1　一次迭代时GPU和CPU运算时间

训练样本数	算法	CPU时间	GPU时间	速度提升
1 449 920	EM	584.3 s	22.06 s	26.49×
	K-means	410.86 s	15.7 s	26.16×
68 281 155	EM	440.72 min	17.01 min	25.9×
	K-means	334.72 min	11.45 min	29.23×

为了能用CPU很快的得到语种GMM模型(单CPU训练模型速度太慢)去验证GPU得到的语种模型的准确性,本文中使用DTM(Distributed task machine)[11]分布式计算技术加快基于CPU实现的语种模型训练速度。DTM技术将一个网络中的所有计算资源(计算机CPU核)综合起来,完成独立的计算任务。使用DTM训练UBM模型时,先将训练样本分成若干份(根据网络中可用CPU核的个数设定),每一颗核计算一份训练样本,保存对应的累计统计量,最后综合所有核的累计统计量更新模型;在训练各语种模型时,每颗核负责训练一个或多个语种模型。

本实验中,DTM使用20颗核。表2给出了GPU和DTM在完整的语种GMM训练过程中每一步(K-means算法训练UBM初值模型、EM算法训练UBM模型及各语种GMM模型)的运算时间。它们只在用K-means算法训练UBM初值模型时使用相同的类中心。从表中看GPU花费的时间都比DTM少(DTM训练各语种GMM模型的时间8 h是一个大概值,因为某些核承担多个语种的训练任务,所以比UBM模型训练时间长)。GPU得到的UBM初值模型和DTM的结果的相对误差比较小(权重0.048 4%,均值0.118 3%,方差0.066 8%);GPU得到的各语种GMM模型平均相对误差中,权重0.35%,均值0.63%,方差0.48%。造成这种误差较大是由于用GPU得到的UBM模型初值与DTM的结果不同,但从最后的模型识别性能看,这种误差并不导致语种识别性能有大的差别,可以忽略不计。表3给出了两种方法得到的语种GMM模型的语种识别性能

(EER)。GPU 得到的语种GMM 模型的识别性能与 DTM 得到的模型相当,这也说明GPU 训练得到的语种GMM 模型是准确的。

表2 DTM 和GPU 模型训练时间

	DTM/h	GPU/h
UBM 初值	6.0	4.7
UBM	5.87	4.317
所有语种GMM	8.0	3.63

表3 DTM 和GPU 得到的语种GMM 模型性能

测试时长/s	EER	
	DTM	GPU
3	19.51%	19.50%
10	7.23%	7.23%
30	2.45%	2.46%

4 结束语

本文提出了一种改进的基于CUDA 的GMM 模型快速训练方法,包括用于模型初始化的 K-means 算法和模型参数估计的EM 算法的快速实现方法。这种GMM 模型快速训练方法采用分批处理策略,不受训练样本数的限制,且高斯混合数取值范围很大。实验结果表明,在训练语种GMM 模型时,这种基于GPU 的模型训练速度比基于单核CPU 的快26 倍左右,大大缩短模型训练时间,而且得到的语种模型的识别性能与CPU 得到的相当。相对文中提到的DTM 技术,本文提出的方法只需要较廉价的NVIDIA 显卡,不用购买昂贵的多核高性能服务器,节省资金成本。

参考文献:

[1] Dempster A P, Laird N M, Rubin D B. Maximum-likelihood from incomplete data via the EM algorithm[J]. Royal Statist Soc Ser B, 1977,39(1):1-38.

[2] Torres-Carrasquillo P A, Singer E, Kohler M A, et al. Approaches to language identification using Gaussian mixture models and shifted delta cepstral features[C]//Proc ICSLP 2002. Colorado, USA: [s. n.], 2002: 89-92.

[3] Kumar N, Satoor S, Buck I. Fast parallel expectation maximization for Gaussian mixture models on gpus using cuda[C]//Proc 11th IEEE International Conference on High Performance Computing and Communications. Washington DC, USA: IEEE Computer Society Press, 2009: 103-109.

[4] Bai Hongtao, He Lili, Ouyang Dantong, et al. K-means on commodity GPUs with CUDA[C]//Proc the 2009 WRI World Congress on Computer Science and Information Engineering. Washington DC, USA: IEEE Computer Society Press, 2009: 651-655.

[5] Zechner M, Granitzer M. Accelerating K-means on the graphics processor via CUDA[C]//Proc the 2009 First International Conference on Intensive Applications and Services. Washington DC, USA: IEEE Computer Society Press, 2009: 7-15.

[6] Bilmes J A. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models[R]. ISCI Technical Report TR-97-021, USA: ISCI,1998.

[7] NVIDIA CUDA programming guide-version 2.2 available[EB/OL] http://www.nvidia.com/object/cuda_develop.html.

[8] Goldberg D. What every computer scientist should know about floating point arithmetic[J]. ACM Comput Surv, 1991, 23 (1): 5-48.

[9] Wong E, Sridharan S. Methods to improve Gaussian mixture model based language identification system [C]//Proc ICSLP 2002. Colorado, USA:[s. n.], 2002: 93-96.

[10] 付强, 宋彦, 戴礼荣. 因子分析在基于GMM 的自动语种识别中的应用[J]. 中文信息学报, 2009, 23(24): 77-81.
Fu Qiang, Song Yan, Dai Lirong. Factor analysis in GMM-Based language identification[J]. Journal of Chinese Information Processing, 2009,23(4):77-81.

[11] Pandey N, Sharma G K. Startup comparison for message passing libraries with DTM on linux clusters [J]. The Journal of Supercomputing, 2007,39 (1): 59-72.

作者简介:吴奎(1989-),男,硕士研究生,研究方向:语音信号处理,E-mail:wukui@mail.ustc.edu.cn;宋彦(1972-),男,博士,研究方向:音视频内容分析与检索;戴礼荣(1962-),男,博士,教授,研究方向:数字信号处理和模式识别。