第3章

基于多核处理器的MATLAB程序加速

基于多核处理器的 MATLAB 程序加速的方法很多，主要包括自动进行的基于多核处理器的程序加速和 parfor 循环。本章在介绍 MATLAB 编程的 4 个主要组成部分（包括 MATLAB 矩阵、函数、语句和程序）的基础上，介绍基于多核处理器的 MATLAB 程序加速。

3.1 MATLAB 矩阵及运算符

MATLAB 是矩阵实验室（matrix laboratory）的缩写，所有的数据都以矩阵的形式存在。那么，MATLAB 矩阵是如何创建和操作的呢？本节讲述 MATLAB 矩阵的创建、MATLAB 矩阵的数据类型和 MATLAB 矩阵的操作。

3.1.1　MATLAB 矩阵的创建

MATLAB 的数据组织形式包括标量(scalar)、向量(vector)、矩阵(matrix)等多种形式。对于标量(scalar)而言,其相当一个只包含一个元素的数组,即 1×1 数组。

MATLAB 向量(vector)是指只有一行或者一列的 MATLAB 矩阵。

MATLAB 矩阵(matrix)是指拥有多行或者多列的 MATLAB 矩阵。MATLAB 的二维矩阵可以用方括号[]来创建。MATLAB 数组可以通过列举所有元素的方法来创建,而这些元素必须用方括号(square bracket)包裹起来。当用方括号[]来定义 MATLAB 的矩阵时,矩阵的列用逗号或者空格来分隔,矩阵的行用分号来分隔。MATLAB 矩阵的所有行必须具有相同的元素数。

很多 MATLAB 数组具有特殊的属性。除了列举所有元素外,可以通过 for 循环和 while 循环来创建这些数组。为了更高效地创建这些具有特殊属性的 MATLAB 数组,MATLAB 数组也可以用以下格式定义:init : increment : terminator。其中,init 表示初始值;increment 表示增量;terminator 表示终止值。MATLAB 矩阵的元素或者子矩阵可以通过圆括号(parenthese)来读写,而圆括号()内包含的是需要读写的 MATLAB 矩阵的元素的索引。索引的数据类型可以是整型或者布尔型。一些特殊的 MATLAB 矩阵可以通过函数来创建,这些函数包括 eye()函数、zeros()函数、ones()函数、rand()函数等。

几乎所有的 MATLAB 程序都从 MATLAB 数组的创建开始。因此,灵活运用高效的 MATLAB 数据创建的多种方法是加速 MATLAB 程序的基础。另外,MATLAB 矩阵是列优先的。例如:

```
m = zeros(2,5);

m(:) = 1:10

m =
     1     3     5     7     9
     2     4     6     8    10
```

MATLAB 矩阵的默认的数据类型为双精度型(double)。MATLAB 变量由赋值操作符=来定义。与 C/C++等语言不一样,因为 MATLAB 具有对不同类型数据

进行转换的内在机制，MATLAB 程序不需要对数据类型进行严格的定义。MATLAB 的变量可以不必进行数据类型的定义，而直接进行赋值。用于赋值的数量可以来自于常量、其他变量或者函数的输出。MATLAB 的变量的赋值以分号结束。当 MATLAB 的变量的赋值没有以分号结束的时候，MATLAB 在命令窗口显示变量的值。

在设备端（GPU 端），gpuArray 支持的数据类型有单精度型（single）、逻辑型（logical）、双精度型（double）、8 位整型（int8）、16 位整型（int16）、32 位整型（int32）、64 位整型（int64）、无符号 8 位整型（uint8）、无符号 16 位整型（uint16）、无符号 32 位整型（uint32）、无符号 64 位整型（uint64）等。

3.1.2 矩阵的性质的检验

在 MATLAB 程序编制的过程中，程序员经常需要对 MATLAB 矩阵的性质进行判断，包括对单个 MATLAB 矩阵的性质进行判断和对两个 MATLAB 矩阵的性质进行判断。

MATLAB 对单个 MATLAB 矩阵的性质进行判断的函数包括 unique()函数、isempty()函数、isinite()函数、isinf()函数、isnan()函数、isprime()函数、issorted()函数、ismember()函数、histcounts()函数、prod()函数等。unique()函数的功能是 MATLAB 矩阵的每个元素只出现一次；isempty()函数的功能是检验 MATLAB 矩阵是否为空集；isinite()函数的功能是检验 MATLAB 矩阵的每一个元素是否不是 Inf 和 NaN；isinf()函数的功能是检验 MATLAB 矩阵的每一个元素是否为 Inf；isnan()函数的功能是检验 MATLAB 矩阵的每一个元素是否为 NaN；isprime()函数的功能是检验 MATLAB 矩阵的每一个元素是否为素数；issorted()函数的功能是检验 MATLAB 矩阵是否经过排序；ismember()函数的功能是检验一个 MATLAB 矩阵的元素是否被另一个 MATLAB 矩阵包含；histcounts()函数的功能是计算输入 MATLAB 矩阵中每一个数值区间的元素数量。MATLAB 的 prod()函数的功能是计算输入 MATLAB 矩阵中元素的乘积。

MATLAB 对多个 MATLAB 矩阵的性质进行判断的函数包括 intersect()函数、setdiff()函数、setxor()函数、union()函数等。MATLAB 的 intersect()函数的功能是两个 MATLAB 矩阵的交集。MATLAB 的 setdiff()函数的功能是两个

MATLAB矩阵的不同元素的集合。MATLAB的setxor()函数的功能是两个MATLAB矩阵的交集以外的元素的集合。MATLAB的union()函数的功能是两个MATLAB矩阵的元素的集合，每个元素只出现一次。

3.1.3　MATLAB矩阵的操作

MATLAB矩阵包含三种操作：矩阵操作(matrix operation)、数值操作(arithmetic operation)和关系操作(relational operation)。MATLAB所有的算术运算都是通过这两种操作进行的。其中，数组操作通过单个标量的方式进行，而矩阵操作(matrix operation)通过线性代数的规则进行。因此，MATLAB数值操作(arithmetic operation)支持高维数值。MATLAB矩阵操作按照线性代数的规则进行操作，矩阵操作不能在多维矩阵上操作。

1. 矩阵操作

MATLAB矩阵的第一种操作是矩阵操作。MATLAB矩阵操作包括五种操作：矩阵乘、矩阵左除、矩阵右除、矩阵乘方和矩阵转置。矩阵乘的操作符是 $*$，矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 的线性代数乘是 $\boldsymbol{C}=\boldsymbol{A}*\boldsymbol{B}$，且矩阵 $\boldsymbol{A}$ 的列数必须和矩阵 $\boldsymbol{B}$ 的行数相等。矩阵左除的操作符是\，等式 $\boldsymbol{Ax}=\boldsymbol{B}$ 的解是 $\boldsymbol{x}=\boldsymbol{A}\backslash\boldsymbol{B}$，矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 必须拥有相同数目的行数。矩阵右除的操作符是/，等式 $\boldsymbol{xA}=\boldsymbol{B}$ 的解是 $\boldsymbol{x}=\boldsymbol{B}/\boldsymbol{A}$，矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 必须拥有相同数目的行数。矩阵乘方的操作符是^，矩阵 $\boldsymbol{A}$ 的 b 次幂是 $\boldsymbol{A}\ \hat{}\ b$，而对于 b 的其他值，计算包括特征值(eigenvalue)和特征向量(eigenvector)。矩阵转置的操作符是′，矩阵 $\boldsymbol{A}$ 的线性代数转置是 $\boldsymbol{A}'$；对于复数矩阵，矩阵 $\boldsymbol{A}$ 的复数共轭转置是 $\boldsymbol{A}'$。

也就是说，如果使用矩阵右除(matrix right division)操作符除两个矩阵，这两个矩阵必须拥有相同的列数。如果使用矩阵乘操作符 $*$ 对两个矩阵进行相乘，那么这两个矩阵必须拥有相同的内部尺寸(inner dimension)。这就是说，第一个矩阵的列数必须等于第二个矩阵的行数。

2. 数值操作

MATLAB矩阵的第二种操作是数值操作。在MATLAB数值操作中，在第一个输入中的每个元素都与一个类似的位于第二个输入的元素相匹配。如果有一个

操作数是标量，MATLAB 数值操作对数组中每一个元素进行相同的运算：

$$a + \boldsymbol{B} = \boldsymbol{A} + \boldsymbol{B},$$

理论上来说，当其中一个数组 a 是标量时，MATLAB 数值运算将这个标量 a 扩展成与 $\boldsymbol{B}$ 同尺寸的数组 $\boldsymbol{A}$。

与 MATLAB 矩阵运算不同的是，MATLAB 数值运算都以句点．开头。MATLAB 数值运算包括加(+)、减(-)、乘(.*)、乘方(.^)、右除(./)、左除(.\)、转置(.')等。其中，加(+)表示两个数组通过单个标量(scalar)的方式进行相加；减(-)表示两个数组通过单个标量(scalar)的方式进行相减；乘(.*)表示两个数组通过单个标量(scalar)的方式进行相乘；乘方(.^)表示数组通过单个标量(scalar)的方式进行乘方；右除(./)表示 $\boldsymbol{A}./\boldsymbol{B}=A(i,j)/B(i,j)$；左除(.\)表示 $\boldsymbol{A}.\backslash\boldsymbol{B}=B(i,j)/A(i,j)$；转置(.')$\boldsymbol{A}.'$表示将数组 $\boldsymbol{A}$ 进行转置。

sigmoid 函数是一个具有 S 形曲线(sigmoid 曲线)的函数。sigmoid 函数通常由如下公式表达：

$$S(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

根据以上公式，计算 sigmoid()函数值的代码为：

```
function X = sigm(P)
    X = 1./(1 + exp(-P));
end
```

如果操作数都是向量或者矩阵，MATLAB 数组操作要求参与的数组具有相同的大小。如果操作数的维度相同而大小不一致时，需要使用 repmat()函数或 repelem()函数将尺寸较小的数组扩大到大小较大的数组的大小，或者将两个数组扩大到相同大小，再进行 MATLAB 数组运算。如果操作数的维度不一致时，需要使用 reshape()函数将两个数组的维度数转换到一致，再进行 MATLAB 数组运算。

然而，MATLAB 的矩阵变换函数，如 repmat()函数、repelem()函数和 reshape()函数，有着较大的计算开销和内存开销。如何在不使用 MATLAB 的矩阵变换函数的情况下，将不同大小的数组进行 MATLAB 数组操作呢？答案是 bsxfun()函数、cellfun()函数和 arrayfun()函数。

MATLAB 的数组运算是指数组元素之间的运算。因此，参与 MATLAB 的数组运算的数组的大小通常是一致的。当参与 MATLAB 的数组运算的数组的大小不

一致时，需要采用 bsxfun()函数。

以下以深度学习工具箱(DeepLearnToolbox)的归一化函数和 softmax 函数为例，说明 bsxfun()函数的用途：

1) 归一化函数

归一化就是要把需要处理的数据经过处理后(通过某种算法)限制在需要的一定范围内。首先，归一化是为了后面数据处理的方便，其次是保证程序运行时收敛加快。归一化的具体作用是归纳统一样本的统计分布性。归一化在 0～1 是统计的概率分布，归一化在某个区间上是统计的坐标分布。归一化有同一、统一和合一的意思。

简而言之，归一化的目的是使得没有可比性的数据变得具有可比性，同时又保持相比较的两个数据之间的相对关系，如大小关系；或是为了作图，原来很难在一张图上作出来，归一化后就可以很方便地给出图上的相对位置等。

在使用机器学习算法的数据预处理阶段，归一化也是非常重要的一个步骤。例如，在应用支持向量机(Support Vector Machine，SVM)之前，训练数据的缩放是非常重要的。对于神经网络和支持向量机而言，缩放都是非常重要的。缩放的最主要优点是能够避免大数值区间的属性过分支配了小数值区间的属性。另一个优点能避免计算过程中数值复杂度。因为关键值通常依赖特征向量的内积(inner product)，例如，线性核和多项式核，属性的大数值可能会导致数值问题。推荐将每个属性线性缩放到区间[－1，＋1]或者[0，1]。

当然，必须使用同样的方法缩放训练数据和测试数据。例如，假设把训练数据的第一个属性从[－10，＋10]缩放到[－1，＋1]，那么如果测试数据的第一个属性属于区间[－11，＋8]，我们必须将测试数据转换成[－1.1，＋0.8]。

该函数归一化输入数组使它的范数或者数值在一定的范围内。在深度学习工具箱 DeepLearnToolbox 的卷积神经网络(Convolutional Neural Network，CNN)中，训练数据进行归一化的代码为：

```
function x = normalize(x, mu, sigma)
    x = bsxfun(@minus,x,mu);
    x = bsxfun(@rdivide,x,sigma);
end
```

其中，输入是需要归一化的训练集合，以及相关的平均值和标准差；输出是归一化以

后的训练集合。相应的操作由 bsxfun()函数完成。

2) softmax 函数

在数学中，softmax 函数又称为归一化指数函数(normalized exponential function)。softmax 函数的功能是将含有 K 个元素的向量压缩到 0～1，并保证 K 个元素的和为 1。softmax 函数的公式是：

$$\delta_i = \frac{e^i}{\sum_{k=1}^{K} e^k}$$

softmax 函数可以用于表示分类概率分布(categorical probability distribution)。在深度学习工具箱 DeepLearnToolbox 的卷积神经网络中，计算 softmax()函数值的代码为：

```
function mu = softmax(eta)
    tmp = exp(3 * eta);
    denom = sum(tmp, 2);
    mu = bsxfun(@rdivide, tmp, denom);
end
```

softmax 函数的除操作由 bsxfun()函数完成。bsxfun()函数不是简单等同于 repmat()加上 MATLAB 的数组运算。MATLAB 的 bsxfun()函数最大的优点是能够自动进行单点扩张(singleton expansion)。单点扩张是指，当两个操作数的维度相同而有且仅有一个维度的尺寸不一致时，bsxfun()函数能够沿着这个尺寸不一致的维度进行“复制”，将尺寸较小的数组扩大到尺寸较大的数组的尺寸，再进行 MATLAB 数组运算。bsxfun()函数的复制过程是虚拟的和并行计算的，所以速度比 repmat()函数快很多。

3. 关系操作

MATLAB 矩阵的第三种操作是关系操作(relational operation)。MATLAB 关系运算是一种特殊的数组运算，用于比较(小于、大于、等于)两个数组。关系运算将两个数组对应的元素逐一比较，运算结果是布尔变量。

关系运算符(relational operator)的作用是对两个矩阵进行关系运算，这些运算包括小于、大于、不等于。关系运算的结果是逻辑矩阵(logical matrix)，其每一个元素反映出两个矩阵中对应位置运算结果是真(true)或假(false)。

MATLAB 关系运算包括 6 个操作符：小于、小于或等于、大于、大于或等于、等于、不等于。

- MATLAB“等于”关系运算的操作符是==，数组 ***A*** 与数组 ***B*** 通过单个标量(scalar)的方式进行等于的比较可以表示为：***C***=***A***==***B***。如果 $A(i,j)=B(i,j)$，则 $C(i,j)$的值为 1。
- MATLAB“不等于”关系运算的操作符是~=，数组 ***A*** 与数组 ***B*** 通过单个标量的方式进行不等于的比较可以表示为：***C***=***A***~=***B***。如果 $A(i,j)=B(i,j)$不成立，则 $C(i,j)$的值为 1。
- MATLAB“小于”关系运算的操作符是<，数组 ***A*** 与数组 ***B*** 通过单个标量的方式进行小于的比较可以表示为：***C***=***A***<***B***。如果 $A(i,j)<B(i,j)$，则 $C(i,j)$的值为 1。
- MATLAB“小于或等于”关系运算的操作符是<=，数组 ***A*** 与数组 ***B*** 通过单个标量的方式进行小于或等于的比较可以表示为：***C***=***A***<=***B***。如果 $A(i,j)<=B(i,j)$，则 $C(i,j)$的值为 1。
- MATLAB“大于”关系运算的操作符是>，数组 ***A*** 与数组 ***B*** 通过单个标量的方式进行大于的比较可以表示为：***C***=***A***>***B***。如果 $A(i,j)>B(i,j)$，则 $C(i,j)$的值为 1。
- MATLAB“大于或等于”关系运算的操作符是>=，数组 ***A*** 与数组 ***B*** 通过单个标量的方式进行大于的比较可以表示为：***C***=***A***>=***B***。如果 $A(i,j)>=B(i,j)$，则 $C(i,j)$的值为 1。

MATLAB 与(AND)逻辑运算的操作符是 &，逻辑运算 A 与逻辑运算 B 通过与逻辑运算形成的逻辑表达式可以表示为：$A\&B$。MATLAB 或(OR)逻辑运算的操作符是|，逻辑运算 A 与逻辑运算 B 通过或逻辑运算形成的逻辑表达式可以表示为：$A|B$。MATLAB 异或(XOR)逻辑运算的操作符是 xor，逻辑运算 A 与逻辑运算 B 通过异或逻辑运算形成的逻辑表达式可以表示为：A xor B。

在设备端(GPU 端)，MATLAB 也支持关系操作。例如，在深度学习工具箱 DeepLearnToolbox 的卷积神经网络中，sigmrnd()函数的作用是产生符合 sigmoid 方程的随机数，代码如下：

```
function X = sigmrnd(P)
    X = double(1./(1 + exp( - P)) > rand(size(P)));
end
```

其中，符合 sigmoid 方程的随机数是由 MATLAB 大于关系运算产生的。

综上所述，MATLAB 主机端（CPU 端）和设备端（GPU 端）都包含 3 种矩阵操作：矩阵操作、数值操作和关系操作。通过算术运算符（arithmetic operator）、关系运算符和逻辑运算符（logical operator）的组合，形成了表达式（expression）。

3.2 MATLAB 函数

MATLAB 函数来源于内置函数、自定义函数、MEX 函数等。除了编程简便以外，MATLAB 的一大优点就是提供多种多样的内置函数，并通过工具箱的形式组织起来。

3.2.1 MATLAB 函数的定义

1. 内置函数

MATLAB 包含大量内置函数，并通过工具箱的形式组织起来。如何在 GPU 上运行这些内置函数呢？

许多 MATLAB 内置函数的输入可以是设备端（GPU 端）的 gpuArray 数组。在设备端（GPU 端）执行 MATLAB 内置函数的时候，主机端（CPU 端）和设备端（GPU 端）的 MATLAB 矩阵可以混合使用。当主机端（CPU 端）和设备端（GPU 端）的 MATLAB 矩阵混合使用的时候，主机端（CPU 端）的 MATLAB 矩阵将自动复制到设备端（GPU 端）。

当 MATLAB 的内置函数运行完成以后，输出的数组是设备端（GPU 端）的 gpuArray。如果输出的结果是标量，这些标量将储存在主机端（CPU 端）；当输出的数组是复数的时候，输出的结果存储在设备端（GPU 端）。MATLAB 的 whos 命令的作用是：显示当前内存的变量信息，包括设备端（GPU 端）的 gpuArray 数组。

2. 自定义函数

MATLAB 自定义函数通过以下格式定义：

```
function [y1,...,yN] = myfun(x1,...,xM)
```

其中，myfun 是 MATLAB 的自定义函数的名称；(x1,…,xM)是输入参数；[y1,…,yN]是输出参数。

MATLAB 自定义函数的定义部分必须在代码的第一行。MATLAB 自定义函数存储在一个以.m 为扩展名的文本文件中，文本文件的文件名必须与 MATLAB 自定义函数的名称一致。MATLAB 自定义函数名称必须由一个字母开始，文件名可以包含字母、数字或者下画线。MATLAB 自定义函数可以包含多个子函数或者嵌套函数。另外，MATLAB 自定义函数必须以关键字 end 结束。

3. MEX 函数

MEX 函数是一种可由 C 语言/CUDA 编写并编译的、可以在 MATLAB 环境中调用的函数。MEX 函数将在 6.2 节和 6.3 节中详述。

3.2.2　MATLAB 函数的执行

总的来说，MATLAB 函数可以直接执行，也可以通过 arrayfun()函数执行。

MATLAB 的 arrayfun()函数可以用于执行主机端(CPU 端)和设备端(GPU 端)的元素级函数。如果 arrayfun()函数的输入为设备端(GPU 端)数组，则输出也为设备端(GPU 端)数组。

程序员利用支持 gpuArray 的函数将主机端(CPU 端)的代码转换为设备端(GPU 端)的代码，获得了运行速度的提升。为了进一步提升 MATLAB 代码的运行速度，程序员应该利用 arrayfun()函数运行设备端(GPU 端)的元素级函数。

因为 MATLAB 的 arrayfun()函数将元素级函数转换成相应的 CUDA 核，程序员可以通过 MATLAB 的 arrayfun()函数提高设备端(GPU 端)的元素级函数的运行速度。

3.3　语句与代码

MATLAB 语句主要包括分支结构和循环结构。

3.3.1 分支结构

MATLAB 条件语句(conditional statement)是在执行时选择执行路径的语句。MATLAB 的条件语句包括：if 语句、switch 语句和 try-catch 语句。

MATLAB 最简单的条件语句是 if 语句，可以通过以下格式进行定义：

```
if 表达式
    一组语句 1
else
    一组语句 2
end
```

MATLAB 的 if 语句对表达式进行判断，当表达式结果是非空或者仅仅含有非零元素时候，表达式为真，则执行一组语句 1；否则，表达式为假，执行 else 包含的一组语句 2。另外，else 和 elseif 所包含的一组语句是可选的。

第二种的条件语句是 switch 语句。当表达式的值是一组已知的数值时，switch 语句能够实现不同执行语句之间的选择功能。switch 语句可以通过以下格式进行定义：

```
switch s 表达式
   case c 表达式
      一组语句
     …
   otherwise
      一组语句
end
```

MATLAB 的 switch 语句对 s 表达式和 c 表达式的值进行判断，如果 s 表达式与 c 表达式的值相等，则执行 c 表达式对应的一组语句。总的来说，当程序员拥有一系列确定的数值的时候，switch 语句比 if 语句简洁明了。然而，因为要求表达式有确定的数值，switch 语句比 if 语句的应用范围窄得多。另外，if 语句和 switch 语句都需要以 end 结尾。

第三种条件语句是 try-catch 语句。try 语句执行代码，catch 语句收集错误信息。try-catch 语句可以通过以下格式进行定义：

```
try
    一组语句 1
catch 错误信息
    一组语句 2
end
```

通过 try-catch 语句，程序员可以覆盖程序中的可能错误，而不会导致程序的中断。MATLAB 首先执行 try 语句包含的一组语句 1，语句产生一个错误，MATLAB 立即对比 try 产生的错误信息与 catch 提供的错误信息；如果一致，MATLAB 将控制权转移到相应 catch 语句，执行一组语句 2，进行错误处理，而不会导致程序中断。如果 try 语句没有产生错误，MATLAB 将执行 try-catch 语句后面的语句。

3.3.2　循环结构

MATLAB 循环语句包括 for 循环和 while 循环。MATLAB 还拥有终止和继续循环的语句：break 语句和 continue 语句。

MATLAB 的 for 循环的基本功能是将一段语句执行指定的次数。for 循环可以通过以下格式进行定义：

```
for 执行的次数
    一组语句
end
```

for 循环执行的次数可以通过以下方式定义：执行次数索引或者列向量。定义执行次数索引的方法和定义普通索引的方法一致：起始值:终止值。其中，起始值表示索引值的起始值，终止值表示索引值的结束值。for 语句从索引值等于起始值开始执行，索引值每次增加 1，直到索引值大于终止值为止。当步长值大于 1 时，执行次数索引可以通过如下方法定义：起始值:步长:终止值。其中，起始值表示索引值的起始值，步长表示每次循环索引增加的数值，终止值表示索引值的结束值。for 语句从索引值等于起始值开始执行，索引值每次增加步长值，直到索引值大于终止值为止。

for 循环执行的次数还可以通过列向量来定义。在 for 循环的每一个循环中，创建给定矩阵 valArray 的一个列向量 index：在第 1 个循环中，index 的值是矩阵 valArray 的第 1 列，在第 2 个循环中，index 的值是矩阵 valArray 的第 2 列，以此

类推。

MATLAB 循环语句还包括 while 循环。while 循环可以通过以下格式进行定义：

```
while 表达式
    一组语句
end
```

当表达式的结果非空而且不为零的时候，表达式为真；反之，表达式为假。当 while 后面的表达式的值为真的时候，while 所包含的一组语句将不断地被执行。

MATLAB 还拥有终止和继续循环的语句：break 语句和 continue 语句。break 语句的作用是终止执行 while 循环。continue 语句的作用是不再执行当前循环余下的语句，继续 for 循环和 while 循环的下一个循环。

3.4 MATLAB 代码

与 MATLAB 函数相同，MATLAB 代码(script)由一系列 MATLAB 命令所构成。但是，MATLAB 函数的执行速度高于 MATLAB 代码的执行速度。

代码执行：解释与编译。

3.5 MATLAB 并行设置

MATLAB 能够在一定程度上自动进行基于多核处理器的 MATLAB 程序加速。对于 MATLAB 的 R2016b 版本而言，程序员应该打开“Parallel Computing Toolbox 预设项”对话框，如图 3.1 所示，确保选择 Automatically create a parallel pool (if one doesn't already exist) when parallel keywords (e.g., parfor) are executed. 项。

对于 for 循环而言，因为 MATLAB 无法判断循环之间的独立性，因此不能自动进行基于多核处理器的 for 循环的加速。

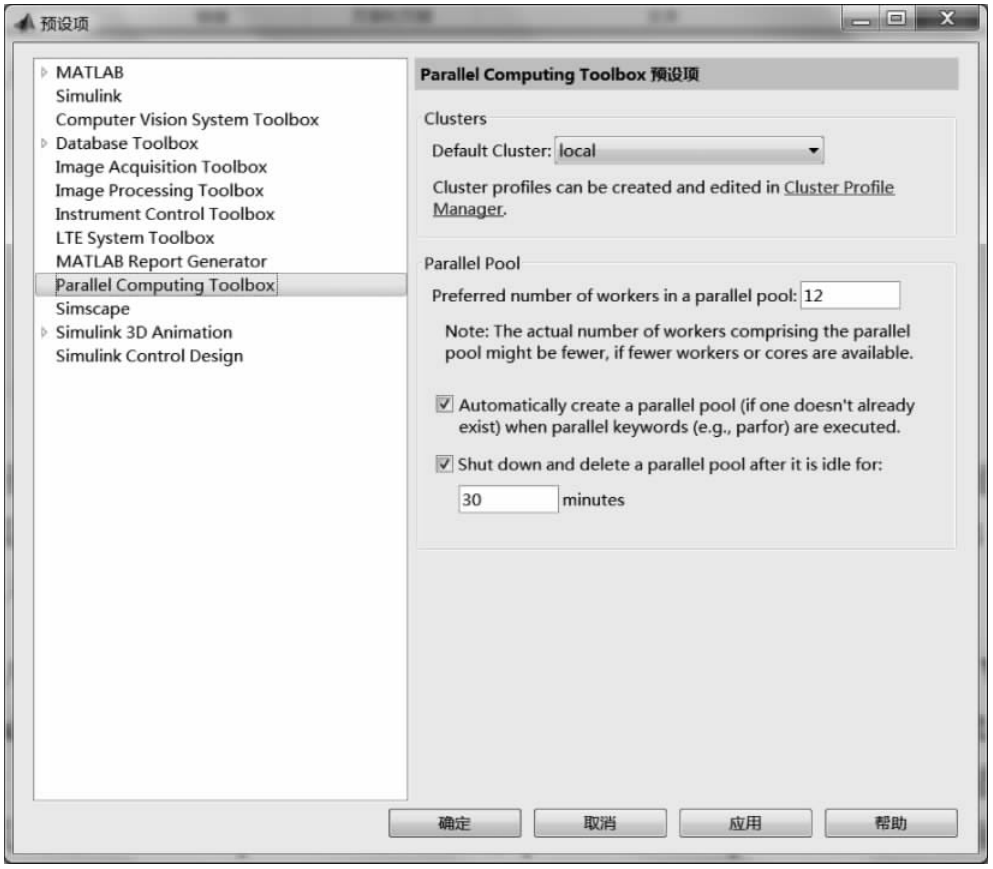

图 3.1　MATLAB 并行设置界面

3.6　基于并行 for 循环(parfor 循环)的 MATLAB 程序加速

当 for 循环的每一个循环都相互独立时,程序员可以使用 parfor 循环加速 for 循环。parfor 循环是并行 for 循环(Parallel for-Loops)的简写。通过 MATLAB 的 parfor 循环,程序员可以利用多核处理器或者集群加速 MATLAB 程序。

MATLAB 的 parfor 循环内的变量包含以下类型:循环变量(loop variable)、切片变量(sliced variable)、广播变量(broadcast variable)、规约变量(reduction

variable)和临时变量(temporary variable)。parfor 的创建非常简单：将每一个循环都相互独立的 for 循环的关键字 for 改成 parfor，即可创建 parfor 循环。parfor 循环不能嵌套：parfor 循环不能包含另一个 parfor 循环，而 parfor 循环可以包含 for 循环。为了提高 parfor 循环的性能，程序员需要仔细考虑在 parfor 循环内部还是 parfor 循环的外部创建矩阵。

在 MATLAB 编程的过程中，程序员经常遇到的情况是将嵌套 for 循环转换为 parfor 循环。对于双层嵌套的 for 循环，程序员可以采用两种方式转换为 parfor 循环：将外层循环转换为 parfor 循环，将内层循环转换为 parfor 循环。将外层循环转换为 parfor 循环优于将内层循环转换为 parfor 循环：将外层循环转换为 parfor 循环产生单个 parfor 循环，而将内层循环转换为 parfor 循环产生多个 parfor 循环。因此，将外层循环转换为 parfor 循环的延迟(overhead)大于将内层循环转换为 parfor 循环。

例如，在卷积神经网络前向传播(forward propagation)的卷积层包含有卷积操作。在进行卷积操作以前，程序员需要将所有的卷积核旋转 90°。因为 MATLAB 的 rot90()函数只能够运行在二维的维度，对于多个卷积核而言，程序员需要使用 for 循环完成所有的旋转操作：

```
for filterNum = 1 : numFilters
    Wc_rotated(:, :, filterNum) = rot90(Wc(:, :, filterNum), 2);
end
```

因为每一卷积核的操作都是相互独立的，程序员只需要简单地将 for 循环变换成 parfor 循环而获得加速：

```
parfor filterNum = 1 : numFilters
    Wc_rotated(:, :, filterNum) = rot90(Wc(:, :, filterNum), 2);
end
```