目录

致谢

摘要

Abstract

1. 绪论
2. 研究背景与相关工作
3. 主要工作概述
4. 表达式摸板类矩阵库的实现
5. 矩阵运算库
6. Mkl
7. CUDA Toolkit

第二节 表达式摸板类的封装

1. 全连接网络的实现
2. word2vec的实现
3. 卷积神经网络的实现
4. 分布式网络的实现
5. 实验结果分析与对比
6. 总结和展望

参考文献

摘要

随着人工智能领域的发展与计算设备的快速更新，文本识别与图像识别的研究受到了越来越多的重视，而神经网络作为

Abstract

1. 绪论
2. 研究背景与相关工作

人工智能领域的研究一直具有十分重要的科学与社会意义。近年来，越来越多的学者致力于研究深度学习，并将其应用于语音识别、图像识别、自然语言处理等，并在各自的研究领域屡屡取得突破性进展。

深度学习是指利用计算机来创建深层次的人工神经网络计算模型，模拟生物神经网络的结构和功能，建立起复杂的模型进行学习与分析，从而令其可以按照人脑的机制识别图片、文字等信息。神经网络通常由大量的人工神经元联结构成，输入数据在其之间传导与计算，并且根据外界信息改变人工神经网络的内部结构，以此对输入和输出数据间复杂的关系进行建模。

1986年，Rumelhart和McCelland提出了误差反向传播算法，以此来对人工神经网络进行有监督的学习。自此，神经网络的研究与应用渐渐进入了人们的视野，一些科学家利用浅层的神经网络实现了汽车在公路上的自动驾驶[],之后又有科学家让计算机通过神经网络，学会了模仿人类的声音[]。慢慢的，人们发现人工神经网络具有一定的局限性，包括受限于当时硬件设备的计算能力，使训练神经网络耗时过长，并且不能处理规模过大的网络；同时网络加深层数后，由于当时神经元的激活函数普遍使用S型函数(Sigmoid)，误差在反向传播时随传播层数的增长呈指数级下降，所以通常多层神经网络只有最后几层学习到了有用的信息，导致其并未表现出比浅层神经网络更好的学习能力；另外，神经网络需要大量的数据进行学习，而当时的小数据集往往会出现过拟合现象，这些因素都导致了在整个90年代深度学习方面的研究进展很小。

1998年，Yann Lecun等人根据人眼感受野的工作原理，提出了首个进行了实际应用的卷积神经网络模型: LeNet-5。他将待识别的图片利用固定大小的卷积窗进行卷积以提取局部特征，并在卷积层后面添加了子采样层进行池化以增加神经网络对图像旋转、偏移的细微差别的鲁棒性。由于图片各个部位对应的卷积窗权值共享，所以极大的降低了参数的数量，加快训练速率，且不易发生过拟合。其卷积的特性在处理二维图像上对边缘信息特征的提取保证了其准确率。LeNet-5被广泛的应用在美国邮政编码与银行手写支票的识别。

2006年，Hinton等人为了克服深层网络误差传播的问题，提出了几种预训练的方法，包括受限玻耳兹曼机、栈式自编码、稀疏编码等方式，利用输入数据对神经网络的权值参数进行非监督的学习，使其预先学习到数据的分布信息，再利用标记数据进行监督学习以微调网络参数。利用这种方式训练出的多层人工神经网络具有非常好的表达。之后Hinton等人又提出了如Drop Out等在小数据集上抑制过拟合现象的方法，以及一些新的神经元激活函数如Rectified 函数等可以在误差反向传播时有效的传播梯度值。

在互联网时代，时刻都在产生大量的数据，而处理这些数据需要较高要求的硬件设备。近年来，图形处理单元(GPU)由于其优秀的计算能力以及并行度上的优势，已经逐渐代替CPU来进行大规模密集型科学计算。英伟达公司最新推出的Tesla K40 GPU，单精度浮点性能已达到4.29TFLOPS。Andrew Ng等人在论文中描述了他们将神经网络模型放到分布式GPU集群上以提高训练的迭代速度。至此深度学习的研究又进入了一个新的阶段。

随着人工神经网络的逐渐成熟，人们将其应用到图像识别、文本学习等领域，并取得了不错的成果。MNIST是一个手写数字数据集，其中包含共60,000个训练样本以及10,000个测试样本，每张图片大小为28\*28的0-255表示的灰度图。在2013年，Li Wan等人使用一种名为DropConnect的神经网络结构，在此数据集上达到了99.79%的分类正确率。CIFAR-10是一个彩色图片数据集，其中包含了10个物体种类共60,000张32\*32的RGB三色图片，其中50,000张训练样本, 10,000张测试样本。在2014年，Min Lin等人提出一种名为”Network In Network”的神经网络结构，在此数据集上达到了91.2%的分类正确率。Word2vec 是谷歌公司发行的一个用神经网络计算连续分布式词汇表达的工具。用word2vec在英文维基数据集上进行训练，并在单词相似度推理任务中可以达到61%的正确率。

为了进行深度学习方面的研究，搭建一个快速、准确、灵活的神经网络平台至关重要。Theano是一个Python的机器学习库，用户可以使用它进行神经网络方面的研究与应用。支持GPU, 开发快捷是Theano的优点。Matlab拥有自己的Neural Network Toolbox，其支持监督学习的前向传递、动态的网络等，同时也支持非监督学习的自组织映射等。使用这个工具箱，用户可以设计、训练神经网络，其最大的特点是具有可视化以及仿真功能。Cuda-convnet 是Hinton等人开发的卷积神经网络 C++/CUDA实现。其主要用于GPU，用户可自定义自己的网络结构，它的优点是训练速度较快。CAFFE平台与Cuda-convnet类似，也是用C++实现的卷积神经网络，不过其在CPU上也有很好的表现。Accord.NET是用C#实现的机器学习库，它支持深度置信网以及受限玻耳兹曼机。

1. 主要工作概述

本文的工作主要包括以下几个方面：

1. 表达式模板矩阵库：

利用表达式模板的方式，封装了CPU与GPU的矩阵运算库。在编译的时候解析表达式，并直接展开成对应矩阵运算函数，避免了开辟和使用临时变量，提高了效率。

1. 全连接网络：

为了使用户便于搭建自己的神经网络结构，将神经网络模型抽象成层(layer)与连接(connection)类的组成。其中，层的子类包括输入层、隐层与输出层；对于全连接网络，连接的子类包括全连接与稀疏连接。由层与连接构成的有向无环图形成一个完整的神经网络。

1. word2vec网络:

word2vec网络在输出层采用哈夫曼编码或负采样(negative sampling) 。

1. 卷积神经网络:

由于卷积神经网络在层与层之间进行卷积与池化的操作，故为连接添加子类卷积连接与池化连接。

1. 分布式网络：

利用消息传递接口(MPI)可以在服务器间进行进程级别的通信。若神经网络的规模较大，可以将模型切分并放置在多台服务器上；若训练数据量较多，可以将完整模型放在多台服务器上，对每个模型分别利用部分数据训练，若干次迭代后进行模型的平均。分布式的网络可以处理大规模的网络和数据，并且可以加速训练的过程。

本文的组织结构如下：

* 第二章介绍了表达式模板矩阵库的实现。主要介绍了CPU与GPU的矩阵运算库，以及运用表达式模板的方法进行封装。
* 第三章主要介绍了全连接网络的实现。其中包括层与连接的实现，以及其各子类的实现。
* 第四章介绍了word2vec网络的结构与实现。
* 第五章简要介绍了卷积神经网络的实现，包括卷积连接与池化连接等。
* 第六章介绍了分布式网络的实现。包括多机间的通信与网络模型的切分等。
* 第七章分析了我们的实验结果，将其与现有的平台进行了对比。
* 第八章进行了简要的总结，并对以后工作进行了展望。分布式网络的实现

1. 表达式模板矩阵库的实现

本章主要介绍了CPU与GPU上的矩阵运算库，并叙述了利用表达式模板的方法对其进行封装。

第一节 矩阵运算库

1. Intel® Math Kernel Library

Intel® Math Kernel Library 是英特尔公司发行的科学计算库。其提高了软件应用程序在进行大规模数学运算时的效率。英特尔MKL库包含了基础线性代数程序集(BLAS, Basic Linear Algebra Subprograms)、Fortran线性代数包(LAPACK, Linear Algebra PACKage)，快速傅里叶变换，量化数学函数以及随机数生成等接口。

在本文所述平台中，主要在CPU模式下运用英特尔MKL库的基础线性代数程序集以及向量数学函数进行矩阵运算。

1. CUDA Toolkit

CUDA™ 是 NVIDIA® 公司的并行计算架构。The NVIDIA® CUDA® Toolkit 是其公司发行的基于CUDA架构的开发包，它为开发利用GPU加速的程序的开发者们提供了一个详尽的基于C/C++的开发环境。CUDA Toolkit包括NVIDIA GPU 的编译器，数学函数库以及一些用于调试和优化程序的工具。

在本文所述平台中，主要在GPU模式下运用NVIDIA CUDA Toolkit中的cuBlas库进行稠密矩阵的运算，cuSparse库进行稀疏矩阵的运算以及利用cuRand库作为GPU上的随机数生成器。

第二节　表达式模板的封装

1. 全连接网络的实现

本章介绍了全连接网络的基本结构以及在平台中具体的实现方法。

1. 全连接网络结构与训练算法

以MNIST数据集为例，构建4层神经网络如图2。MNIST数据集是由10种(0-9)每张大小为28\*28的手写数字组成，故该网络输入层节点数为784维，输出节点为10维。另外此网络包含2个隐含层，隐含层节点数为1024维。层与层之间采用全连接。输出层代价函数采用均方差函数，激活函数采用SoftMax激活函数。

求解神经网络分为两步，首先前向传播得到输出结果，之后根据输出值与标记样本计算代价函数，利用批量梯度下降或其他算法求解神经网络。

当前向传播时，每次将 个 维的输入数据合为一批，组成一个 维的输入矩阵，得到输入层的激活值。之后对于除输入层外的每一层进行循环，将上一层得到的激活值与层与层间的权值矩阵做矩阵乘法，得到本层神经元的输出值。对于隐含层，用激活函数(对于本任务，通常使用Sigmoid函数或Rectified函数) 激活本层神经元的输出值，作为本层神经元的激活值。对于输出层，由于是多分类问题，所以需要用SoftMax函数作用在本层神经元的输出值，得到各个分类的概率分布。输出概率最高的神经元可被认为是本样本的输出值。对于二分类问题，可使用单一输出的神经元，输出层的激活函数可以选Sigmoid函数。

在得到神经网络的输出值后，我们可以计算整体的代价函数。之后我们使用批量梯度下降法来求解神经网络，目的是为了得到合适的权重值与偏置值，使得代价函数最小。因为代价函数(本任务中使用均方差函数)是一个非凸函数，故梯度下降法很可能收敛到局部最优解。但是在实际应用中，梯度下降法通常能非常快速的得到一个令人满意的结果。

在梯度下降法中，每一次迭代都要对权重值与偏置值求导，故计算偏导数是求解过程中的关键步骤。在神经网络中，通常使用误差反向传播算法来计算偏导数。反向传播算法的思路如下：给定一个样例 ,我们首先进行“前向传导”运算，利用前向传导公式，得到 直到输出层 的激活值。对输出层（第 \textstyle n_l 层），计算其残差值： \begin{align}
\delta^{(n_l)}
= - (y - a^{(n_l)}) \bullet f'(z^{(n_l)})
\end{align}，对于 \textstyle l = n_l-1, n_l-2, n_l-3, \ldots, 2 的各层，计算其残差值： \begin{align}
\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \bullet f'(z^{(l)})
\end{align}。最后计算最终需要的偏导数值： \begin{align}
\nabla_{W^{(l)}} J(W,b;x,y) &= \delta^{(l+1)} (a^{(l)})^T, \\
\nabla_{b^{(l)}} J(W,b;x,y) &= \delta^{(l+1)}.
\end{align}

神经网络训练算法流程如以下伪代码所示：

**算法：手部的区域生长**

01 **输入**：样本集\textstyle \{ (x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)}) \} ，网络权重初始化范围，梯度下降学习速率alpha。

02 **输出**：神经网络权重以及网络偏置。样本输出值。

03 初始化网络权重。网络偏置置零。

**第一部分：神经网络训练算法**

04 对于 – 每批m个数据 做

05 前向传播

06 误差反向传播

07 结束 – 循环

**第二部分：前向传播算法**

08 对于 – 每层 神经元 做

09 如果 是输入层 做

10

11 否则 如果 是隐含层 做

12

13

14 如果 是输出层 做

15

16

17 结束 – 如果

18 结束 – 循环

19 对于 – 每个 输出层神经元

20 寻找激活值最大的神经元作为该样本输出

21 结束 – 循环

**第三部分：误差反向传播**

22 对于 – 每层 神经元 做

23 如果 是输出层 做

24 ;

25 否则 如果 是隐含层 做

26

27

28

29

30

31 结束 – 如果

32 结束 – 循环

1. 全连接网络的设计模式

为了使用户便于搭建自己的神经网络，将神经网络模型抽象成层(layer)类与连接(connection)类的组成。

1. 层类

第四章 word2vec的实现

本章介绍了word2vec实验的基本网络结构以及在平台中具体的实现方法。

第一节　人工神经网络词汇表达模型

现阶段人工神经网络词汇表达模型多使用词向量表达，又称分布式词汇表达(Distributed word representation)，指使用低维、稠密的实数向量来表达词汇。词汇的语义信息通过向量各维的数值进行表达。随着互联网上文本数据的爆发性增长，以及深度学习技术的发展，分布式词汇表达广泛被应用在许多的文本挖掘任务中。许多研究者也提出了有效、高效的分布式词汇表达模型和训练算法。

词向量表达通常使用无监督机器学习算法产生，多数算法使用上下文相关的语言模型，以人工神经网络或其变种作为概率模型。无监督机器学习不需要人工对训练数据进行标注，可以充分利用互联网上海量的语料数据进行训练。训练所得的词向量可用于机器翻译、文本挖掘、文档分类等多种领域。

自2003 年Bengio 等人发表人工神经网络概率语言模型以来，出现了众多基于人工神经网络训练词向量表达的方法。2005 年，Morin 和Bengio 发表了基于层次结构的Softmax 模型[2]，使用树状层次结构代替原有的平铺结构，将算法的时间复杂度从O(n)降为O(())，大大提高了算法的效率。2010 年，Mikolov 等人发表了Word2Vec 项目，提出了CBOW（Continuous Bag-of-word，连续词袋）模型，和Skip-Gram 模型（如图2）。Word2vec 项目的两种模型对人工神经网络进行了进一步简化，并且吸收了 Morin 的树状层次结构，使得训练效率进一步提高。Mikolov 还提出了一种语义类比问题，用于评价词向量表达的质量。2013 年的NIPS 会议上，Mikolov 等人发表了Word2Vec 项目的改进版本，加入了负样本概率模型，再一次提高了模型的训练效率。

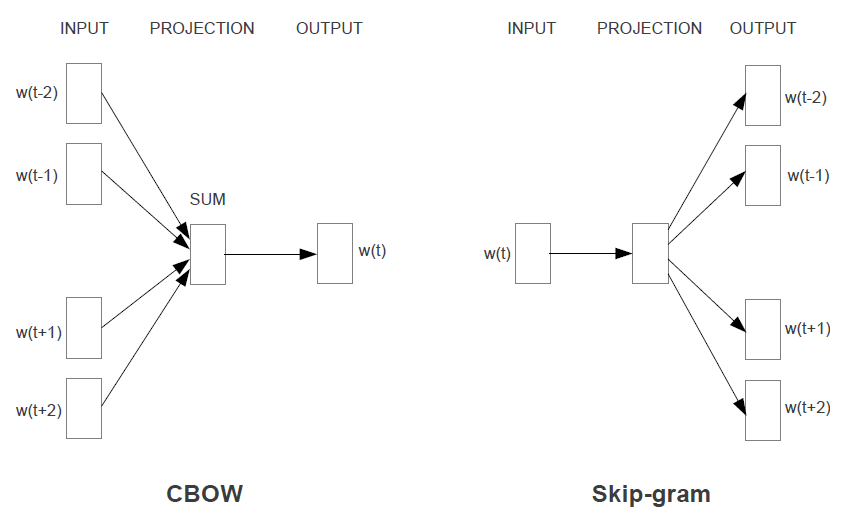


图2 CBOW模型和Skip-gram模型

第二节 word2vec的基本结构与评测方法

word2vec可用来计算连续分布式词汇表达。它利用文本语料库作为输入数据，格式为词表大小的1-of-v向量表示词汇，输出的模型为低维的词汇表达向量。

word2vec使用两种词汇模型：CBOW模型和Skip-gram模型。如图3所示，将输入数据语料库中的文本使用固定大小的滑动窗口进行截取，选取其中词汇，在CBOW模型中，将窗口中心词汇作为标记信息，其余词汇作为训练样本进行输入。相反的，Skip-gram模型中使用中心词汇来预测其上下文中其他词汇。

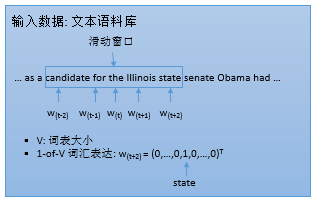


图3 滑动窗口取词

如图4，CBOW模型是一个两层神经网络结构的模型。输入层维度是词表大小，输入数据为n-of-v的词汇向量，表示窗口中在中心词上下文的n个词。隐含层维度为词汇表达向量空间的维度。输入层与隐含层之间连接的权值矩阵即词汇向量表达矩阵。输出层为1-of-v的预测中心词汇。Skip-gram模型

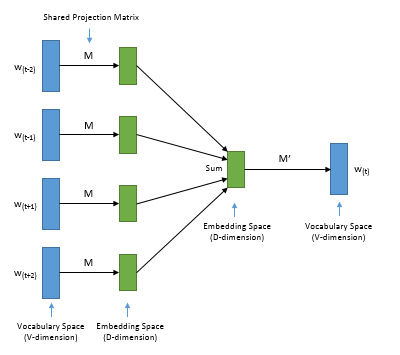


图4 CBOW模型网络结构

第五章 卷积神经网络的实现

本章介绍了卷积神经网络的基本结构以及在平台中具体的实现方法。

1. 卷积神经网络的基本结构

在 2006 年深度置信网的训练算法提出以前，基于生物学研究成果，由Fukushima、LeCun等人提出并发展的卷积神经网络是为数不多的，可以在避免过拟合情况下，进行多隐层训练的神经网络，具有结构简单、训练参数少和适应性强等特点。特别是在图像识别领域，可以直接对二维图像（而无需一维化）进行建模，对平移、比例缩放、倾斜或者共他形式的变形具有高度不变性。

LeCun在1998 年提出了首个进行了实际应用的神经网络模型：LeNet-5。卷积神经网络由于其权值共享、参数自由度较小的特点，深层训练时，不易发生过拟合。特别是其卷积的特性使得其在处理二维图像上有着很强的优势。

下面简单介绍LeNet-5的基本结构：

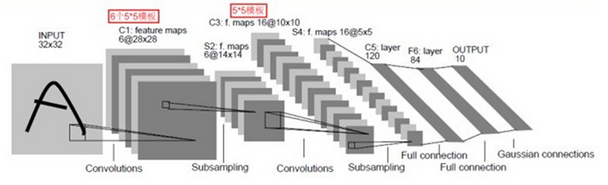


图2 LeNet-5的基本结构

如图2所示，LeNet-5共有7层，它将MNIST数据库的每张图片进行填充(padding)后作为输入。MNIST手写数据库的图片大小为28\*28，填充操作将每张图片外围填补2层零值，即原图变为32\*32大小。填充操作的目的是希望图片边缘的明显特征能够出现在最高层特征监测子感受野(卷积窗口)的中心。

第一层为卷积层，卷积模板大小为5\*5，步长为1，共使用6个可训练的卷积模板。使用卷积模板对输入图像进行卷积过后，加一个偏置b，得到第一个卷积层。第一个卷积层有6张卷积过后的特征图片，大小为28\*28。卷积层的目的是提取图片特征。

第二层为池化层，池化窗口为2\*2，步长为2。池化操作即对池化窗口内的像素点取最大值或平均值作为下一层图片对应位置的值，称之为最大值池化或平均值池化。池化后的图片大小为14\*14。池化后再经过激活函数，即得到池化层。池化层的目的是增加神经网络对图片细微旋转和偏移的鲁棒性。一次卷积和池化的完整操作见图3的。

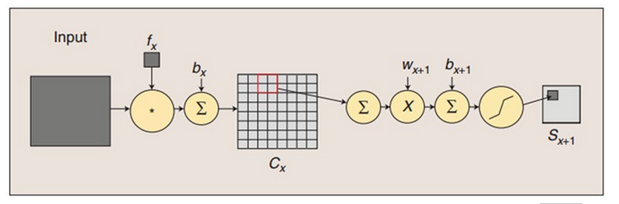


图3 卷积与池化操作

第三层依然是卷积层，拥有16个大小为5\*5,步长为1的卷积模板，卷积后得到16张大小为10\*10的特征图片。本层得到的特征图片表示的是上一层提取到特征图片的不同组合(这里类比人的视觉系统，底层的结构构成上层更抽象的结构，例如边缘构成形状)。第四层是池化层，池化窗口

为2\*2，步长为2。第五、六、七层均为全连接层，分别拥有120, 84, 10个神经元，其中第七层为输出层。

1. 卷积神经网络的实现

由于图像的存储与输入格式采用一维数组的方式，在进行卷积操作时，寻址的开销与计算的开销非常昂贵。为了避免这种情况，考虑到任务的实际情况进行相应优化。

对于输入图像，由于有多个相同大小的卷积模板要对其进行卷积，所以考虑事先将输入图像按照卷积相乘时所对应的位置将其展开。我们将这种操作叫做im2col。当对输入图像进行im2col操作后，得到展开后的图像，在做前向传播时即可直接与权值矩阵相乘，得到原图与权值矩阵进行卷积后的效果。在反向传播时，直接将误差信号与权值矩阵相乘，得到被展开的误差信号，此时做im2col的反向操作，将误差信号完整的传播到下一层神经元上。

具体的流程如下：

1. 卷积连接中的前向传播。
2. 前层神经元的输入：使用I.N×I.C×I.H×I.W长度的向量表示输入图像，I.N表示批处理图像时一次训练的样本数量， I.C表示前层卷积特征图像的个数（对于输入层即输入图像通道数），I.H表示输入特征图像的高度， I.W表示输入特征图像的宽度。
3. 卷积连接的权值矩阵：F.N×F.C×F.S×F.S，F.N表示本层卷积操作所需卷积模板的个数(即生成特征图像个数)， F.C=I.C，表示前层卷积特征图像的个数，F.S×F.S是卷积窗口的大小。
4. 后层神经元的输出：O.N×O.C×O.H×O.W，O.N=I.N，表示批处理图像时一次训练的样本数量，O.C=F.N，表示输出特征图像的个数（也即本层卷积连接的卷积模板个数），O.H×O.W表示输出图像的大小。
5. 依次对每张输入图片进行采样展开(im2col)：如图4，令大小为F.S×F.S的卷积窗口在大小为I.H×I.W，通道数为I.C的输入图像上滑动，将每次采样得到的F.C×F.S×F.S个数值依次填入Col\_data矩阵的一列。采样完毕后，可得到将输入图像按照卷积顺序展开的完整Col\_data矩阵。



图4 Im2col操作

1. 将权值矩阵二维化，使其行向量表示一个卷积模板的所有权重，行数表示本层卷积模板的数量。
2. 如图5，计算后层神经元的输出，将权值矩阵与展开后的输入图像相乘，得到输出的特征图像。



图5 前向传播

二、卷积连接中的误差反向传播

1. 误差反向传播时的误差信号维度与输入图像相同。

2. 首先传播误差信号，按照公式，如图6，计算前层神经元采样展开后的误差输出。



图6 反向传播误差信号

3. 对于计算所得前层误差输出，需进行反采样展开(col2im)得到前层神经元对应的结果再输出。如图7所示。



图7 col2im

4. 利用后层神经元误差信号与前层神经元前向传播时的激活值对权重进行更新。首先对前层神经元激活值进行采样展开，得到展开后矩阵。如图4。

5. 如图8，根据公式 ，对权值矩阵进行更新。



图8 反向传播更新权重