

TensorFlow 平台下的手写字符识别

张俊, 李鑫

(太原理工大学 信息工程学院, 山西 太原 030024)

摘要: 基于谷歌第二代人工智能学习系统 TensorFlow, 构建 BP 神经网络模型。将手写字符作为训练集输入神经网络, 训练过程中不断调整权值和阈值, 最终得到有较高识别精度的模型。体现了 TensorFlow 在提高建模、编程、分析效率中的作用。通过此开发流程介绍, 为进一步使用 TensorFlow 构建复杂神经网络提供了参考。

关键词: 人工智能; TensorFlow; BP 神经网络

中图分类号: TP18 文献标识码: A 文章编号: 1009-3044(2016)16-0199-03

DOI: 10.14004/j.cnki.ckt.2016.2359

Handwritten Character Recognition Based On TensorFlow Platform

ZHANG Jun, LI Xin

(College of Information Engineering, Taiyuan University of Technology, Taiyuan 030024, China)

Abstract: Based on Google's second generation of artificial intelligence learning system—TensorFlow, build a BP neural network model. Use handwritten characters as training set of inputs of neural network, constantly revise weight value and threshold value in the process of training, and get a higher identification precision of the model. It Embodies the TensorFlow's effect in improving the efficiency of modeling, programming and analysis. Through introducing the development process, It provides reference which use TensorFlow building complex neural networks.

Key words: artificial intelligence; TensorFlow; BP neural networks

1 概述

目前, 手写字符的识别方法已经有多种, 如支持向量机、BP 神经网络、KNN、朴素贝叶斯方法等。其中 BP 神经网络作为一种经典的模式识别工具, 应用广泛。将 BP 神经网络应用于手写字符识别, 具有识别速度快、分类能力强、有较好的容错性能和学习能力的优点。

TensorFlow 是一个采用数据流图, 用于数值计算的开源软件库。它通过构建有向图来描述所要执行的操作, 可以灵活的使用设备中的 CPU 或者 GPU 展开计算。TensorFlow 提供了构建神经网络的接口, 因此便于构建 BP 神经网络, 简化编程任务。与传统平台构建的识别模型相比, 提高了效率。

2 TensorFlow 深度学习平台

2.1 TensorFlow 平台特性

TensorFlow 使用灵活, 无论是个人 PC 还是大规模 GPU 计算集群, TensorFlow 都能够灵活的在这些平台运行, 使用 TensorFlow 表示的计算也可以在这些平台上方便地移植。目前, TensorFlow 已经被应用于机器学习系统, 以及和计算机科学相关的领域, 例如计算机视觉、语言识别、信息检索、机器人、地理信息抽取、自然语言理解和计算药物发现等。TensorFlow 灵活的特性也可以用来表示很多的算法, 比如推断算法和深度神经网络的训练等。

TensorFlow 采用数据流计算, 其表达的数据流计算由一个有向图表示, 这个图由一个节点集合组成。在一幅 TensorFlow 图中, 每个节点有一个或者多个输入和零个或者多个输出, 表示一种操作的实例化。图中的叶子节点通常为常量或者变量, 非叶子节点为一种操作, 箭头代表的是张量(常量、变量以及节点计算出的结果均可视为张量)的流动方向。

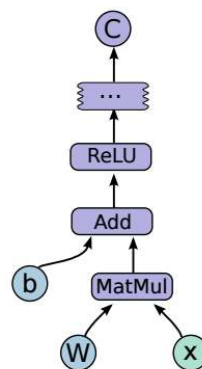


图1 TensorFlow 计算图

通过在 Ubuntu 系统上安装 TensorFlow 库、启用 GPU 支持即可使用 Python 语言构建计算图, 实现基于神经网络的手写字符识别。

收稿日期: 2016-03-25

基金项目: 太原理工大学国家级大学生创新创业项目资助(编号 201410112013); 山西省大学生创新创业项目资助(编号 2014053)

作者简介: 张俊, 男, 山西安泽县人, 本科生, 主要研究方向为自然计算与图像处理。

2.2 TensorFlow 平台搭建

TensorFlow 目前支持 Ubuntu 系统和 MAC OS 系统中安装,支持 C++ 和 Python 两种编程语言。考虑到 Python 语言的简洁性,本实验采用 Python 进行程序设计。

1) 使用 Virtualenv 创建隔离容器安装 TensorFlow, 不会改变不同 Python 项目的依赖关系, 便于进行项目的管理, 能使排查安装问题变得更容易。

```
$ sudo apt-get install python-pip python-dev python-virtualenv
```

2) 在 Virtualenv 环境下安装 TensorFlow 的 GPU 版本:

```
(tensorflow)$ pip install --upgrade https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.6.0-cp27-none-linux_x86_64.whl
```

3) 启用 gpu 支持

安装 Cuda 工具包 7.0 和 6.5 CUDNN V2, 可以使用 GPU 进行运算。

Cuda 工具包 7.0 安装:

```
sudo dpkg --i cuda-repo-ubuntu1504-7-5-local_7.5-18_amd64.deb
```

```
sudo apt-get update
```

```
sudo apt-get install cuda
```

6.5 CUDNN V2 安装:

```
tar xvfz cudnn-6.5-linux-x64-v2.tgz
```

```
sudo cp cudnn-6.5-linux-x64-v2/cudnn.h /usr/local/cuda/include
```

```
sudo cp cudnn-6.5-linux-x64-v2/libcudnn* /usr/local/cuda/lib64
```

配置环境变量:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
```

```
export CUDA_HOME=/usr/local/cuda
```

4) 安装 Numpy

```
apt-get install python-numpy
```

2.3 TensorFlow 平台开发流程

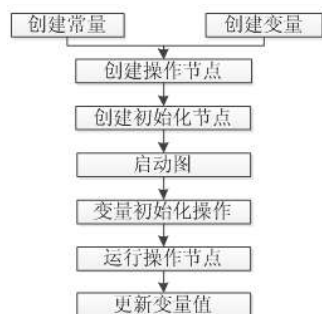


图2 TensorFlow 开发流程

创建图: TensorFlow 使用图来表示计算任务, 在执行计算操作之前需要将图构建完成。每一个非叶子节点都视为一种操作, 叶子节点则表示特殊的操作类型, 比如返回常量值或者变量值。创建图的最后阶段需要向图中添加一个初始化操作的节点, 其作用是将所有的变量进行初始化。

启动图: 图创建完成后, 才能启动图。启动图的第一步是创建一个 Session 对象, 如果无任何创建参数, 会话构造器将启动默认图。然后进行变量的初始化操作、运行操作节点、更新

变量值。

3 BP 神经网络设计

BP 神经网络是一种按误差逆传播算法训练的多层前馈网络, 是目前应用最广泛的神经网络模型之一。学习规则使用最速下降法, 通过反向传播来不断调整网络的权值和阈值, 使网络的误差平方和最小。由输入层、隐含层和输出层组成, 层与层之间采用全连接方式, 同层之间不存在相互连接。

3.1 BP 神经网络

三层 BP 神经网络模型如图 3 所示, 分为输入层、隐含层和输出层, 还包括的参数有:

w_{ij} : 输入层第 i 单元到第一隐含层第 j 单元的权重;

w_{jk} : 隐含层第 j 单元到输出层第 k 单元的权重;

b_j : 隐含层第 j 单元激活阈值;

b_k : 输出层第 k 单元激活阈值;

$f(X)$: 激活函数采用 S 型激活函数, 既 $f(X) = \frac{1}{1 + e^{-x}}$ 。

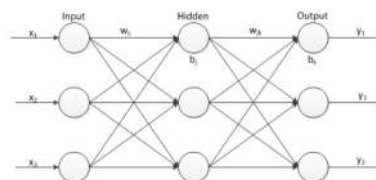


图3 BP 网络模型

误差计算: 方差代价函数为 $E(w, b)$, 其中 y_i 为计算输出结果, d_i 为期望输出结果。

$$E(w, b) = \frac{1}{2} \sum_{i=0}^{n-1} (y_i - d_i)^2 \quad (1)$$

权值阈值修正: 权值和阈值的修正依赖于误差信号的反向传输。根据梯度下降算法, 通过计算输出层误差, 可以调整隐含层和输出层之间的权值和阈值, 同样可以调整输入层和隐含层之间的权值和阈值。通过反复修正权值和阈值, 使代价函数 $E(w, b)$ 达到最小。

设节点 i 和节点 j 之间的权值为 w_{ij} , 节点 j 的阈值为 b_j , 每个节点的输出值为 x_i , 则调整过程为:

$$w_{ij} = w_{ij} - \eta_1 \frac{\partial E(w, b)}{\partial w_{ij}} = w_{ij} - \eta_1 \sigma_{ij} x_i \quad (2)$$

$$b_j = b_j - \eta_2 \frac{\partial E(w, b)}{\partial b_j} = b_j - \eta_2 \sigma_{ij} \quad (3)$$

3.2 TensorFlow 平台实现

本实验采用 MNIST 手写字符数据集, 手写字符为 28×28 像素的手写数字灰度图像。存储在 train-images-idx3-ubyte.gz 和 train-labels-idx1-ubyte.gz 文件中的 60000 幅手写字符数据, 55000 幅作为训练集, 5000 幅作为验证集。测试集 10000 幅图像的字符和标签存储在 t10k-images-idx3-ubyte.gz 和 t10k-labels-idx1-ubyte.gz 中。

输入层设计: 手写字符每一张图片的大小为 32×32 , 一维化后, 每一张图片作为输入时需要 784 个输入层神经元节点, 其中 None 表示输入图片的数目:

```
x = tf.placeholder("float", shape=[None, 784])
```

隐含层设计: previous_units 为前一层神经元节点数, hidden_units 当前层神经元节点数:

```

weights = tf.Variable(tf.truncated_normal
([previous_units, hidden_units], stddev=1.0 / math.sqrt(float
(IMAGE_PIXELS))),name='weights')
biases = tf.Variable(tf.zeros([hidden_units]),name='biases')
hidden = tf.nn.relu(tf.matmul(images, weights) + biases)
输出层设计:输出层为线性的 softmax 回归模型,用大小为
10 的一维张量表示 10 个不同的类别:
weights = tf.Variable(tf.truncated_normal
([hidden2_units, 10],stddev=1.0 / math.sqrt(float
(IMAGE_PIXELS))),name='weights')
biases = tf.Variable(tf.zeros([NUM_CLASSES]), name='biases')
logits = tf.matmul(hidden2, weights) + biases
计算损失:以交叉熵作为损失函数,训练过程与方差代价
函数类似,但是可以克服方差代价函数更新权重过慢的问题:
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits, onehot_labels,name=
'xentropy')
loss = tf.reduce_mean(cross_entropy, name='xentropy_mean')

```

3.3 模型训练与评估

TensorFlow 可以灵活访问整个计算图,它可以使用自动微分法找到对于各个变量损失的梯度值。TensorFlow 有大量内置的优化算法,采用最速下降法让交叉熵下降,步长为 0.01:

```

optimizer = tf.train.GradientDescentOptimizer(learning_rate)

```

optimizer 在运行时会使用梯度下降来更新参数。因此,整个模型的训练可以通过反复地运行 optimizer 来完成。

图 4 显示了模型学习到的图片上每个像素对于特定数字类的权值。红色代表负数权值,蓝色代表正数权值。

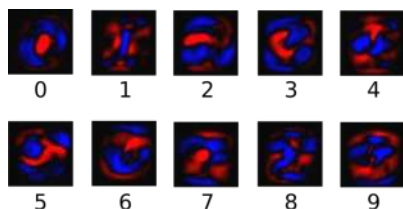


图 4 模型训练

找到所有预测正确标签的数量,得出模型预测的正确率

```

correct = tf.nn.in_top_k(logits, labels, 1)
return tf.reduce_sum(tf.cast(correct,tf.int32))

```

通过训练模型,最终得到如图 5 所示的结果:训练集的正确率为 99.06%;验证集数据正确率为 97.66%;测试集数据正确率为:97.64%。

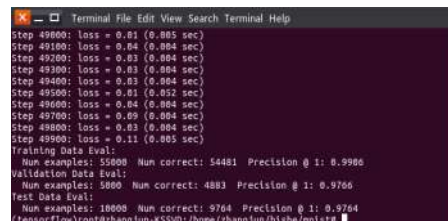


图 5 程序运行结果

4 总结

目前,神经网络被广泛应用于学术界和工业界,但在构建神经网络、优化参数、模型分析方面总是存在着一定的困难。TensorFlow 的出现使这种状况得以改善。TensorFlow 有着编程简单、优化算法集成度高、使用灵活的特性,利用 TensorFlow 将会使建模、编程、分析效率大大提高。实验以 BP 神经网络为例,介绍了 TensorFlow 平台下神经网络模型构建的一般方法和流程,并且得到了有较高识别精度的模型。对构建更复杂的神经网络具有一定的参考意义。

参考文献:

- [1] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015[J]. Software available from tensorflow.org.
- [2] 任翠池,杨淑莹,洪俊.基于 BP 神经网络的手写字符识别[J].天津理工大学学报,2006,22(4):80-82.
- [3] 张斌,赵玮烨,李积宪.基于 BP 神经网络的手写字符识别系统[J].兰州交通大学学报:自然科学版,2007,26(1).
- [4] 许宜申,顾济华,陶智,等.基于改进 BP 神经网络的手写字符识别[J].通信技术,2011,44(5):106-109.
- [5] 杨勇,谢刚生.基于 BP 神经网络的手写数字识别[J].华东地质学院学报,2003,26(4):383-386.
- [6] 金连文,徐秉铮.基于多级神经网络结构的手写体汉字识别[J].通信学报,1997,18(5).
- [7] 李平,蒋振刚.神经网络对手写字符识别特征的提取[J].长春光学精密机械学院学报,2000,23(2):22-24.
- [8] 朱大奇,史慧.人工神经网络原理及应用[M].北京:科学出版社,2006.