Author:Flynn
E-Mail:zofon@qq.com

Machine Learning way

2017 年 6 月 23 日

This is
Open Source

目录

第一章 相关的概念 **5**

第二章 如何研究学习一个机器学习算法 **7**

2.1 引言 7

2.2 什么是研究机器学习算法 7

2.3 怎样研究学习机器学习算法 7

2.3.1 选择一个算法 7

2.3.2 确定一个问题 7

2.3.3 设计实验 8

2.3.4 进行试验并且报告你的结论 8

2.3.5 重复 8

2.4 研究学习算法不仅仅是学者才能做的 9

第三章 几种框架的比较 **11**

3.1 引言 11

3.2 网络和模型能力 11

3.2.1 Caffe 11

3.2.2 CNTK 11

3.2.3 TensorFlow 11

3.2.4 Theano 11

3.2.5 Torch 12

3.3 接口 12

3.3.1 Caffe 12

3.3.2 CNTK 12

3.3.3 TensorFlow 12

3.3.4 Theano 12

3.3.5 Torch 12

3.4 模型部署 12

3.5 性能 13

3.6 架构 13

3.7 跨平台 13

第四章 神经网络入门 **15**

4.1 生物学的大脑 15

第五章 深度学习实例 **17**
5.1 CNN 实例 17

第一章　相关的概念

在学习机器学习之前首先要了解一些相关的概念。

AI 人工智能是最早出现的，也是最大、最外侧的同心圆；

ML 机器学习

DL 深度学习，深度学习造成了前所未有的巨大的影响，是当今人工智能大爆炸的核心驱动。

五十年代，人工智能曾一度被极为看好。之后，人工智能的一些较小的子集发展了起来。先是机器学习，然后是深度学习。深度学习又是机器学习的子集。。

早在 1956 年夏天那次会议，人工智能的先驱们就梦想着用当时刚刚出现的计算机来构造复杂的、拥有与人类智慧同样本质特性的机器。这就是我们现在所说的“强人工智能”（General AI）。这个无所不能的机器，它有着我们所有的感知（甚至比人更多），我们所有的理性，可以像我们一样思考。

人们在电影里也总是看到这样的机器：友好的，像星球大战中的 C-3PO；邪恶的，如终结者。强人工智能现在还只存在于电影和科幻小说中，原因不难理解，我们还没法实现它们，至少目前还不行。

我们目前能实现的，一般被称为“弱人工智能”（Narrow AI）。弱人工智能是能够与人一样，甚至比人更好地执行特定任务的技术。例如，Pinterest 上的图像分类；或者 Facebook 的人脸识别。

这些是弱人工智能在实践中的例子。这些技术实现的是人类智能的一些具体的局部。但它们是如何实现的？这种智能是从何而来？这就带我们来到同心圆的里面一层，机器学习。

第二章　如何研究学习一个机器学习算法

中文:http://blog.jobbole.com/80658/ 英文:http://machinelearningmastery.com/how-to-investi

2.1　引言

机器学习算法都是一个个复杂的体系，需要通过研究来理解。学习算法的静态描述是一个好的开始，但是这并不足以使我们理解算法的行为，我们需要在动态中来理解算法。

机器学习算法的运行实验，会使你对于不同类型问题得出的实验结论，并对实验结论与算法参数两者的因果关系有一个直观认识。

在这篇文章中，你将会知道怎么研究学习一个机器学习算法。你将会学到 5 个简单步骤，你可以用来设计和完成你的第一个机器学习算法实验

你会发现机器学习实验不光是学者们的专利，你也可以；你也会知道实验是通往精通的必经之路，因为你可以从经验中学到因果关系的知识，这是其它地方学不到的。

2.2　什么是研究机器学习算法

当研究一个机器学习算法的时候，你的目标是找到可得到好结果的机器算法行为，这些结果是可以推广到多个问题或者多个类型的问题上。

你通过对算法状态做系统研究来研究学习机器学习算法。这项工作通过设计和运行可控实验来完成

一旦你完成了一项实验，你可以对结论作出解释和提交。这些结论会让你得以管窥在算法变化中因果关系。这就是算法行为和你获得的结论间的关系。

2.3　怎样研究学习机器学习算法

在这一部分，我们将学到 5 个简单的步骤，你可以通过它来研究学习一个机器算法

2.3.1　选择一个算法

选择一个你有疑问的算法

这个算法可能是你正在某个问题上应用的，或者你发现在其他环境中表现很好，将来你想使用

就实验的意图来说，使用现成的算法是有帮助的。这会给你一个底线：存在 bug 几率最低

自己实现一个算法可能是了解算法过程的一个好的方式，但是，实验期间，会引入额外的变量，比如 bug，和大量必须为算法所做的微观决策

2.3.2　确定一个问题

你必须有一个你试图寻找答案的研究问题。问题越明确，问题越有用

给出的示例问题包括以下几个方面：

KNN 算法中，作为样本空间中的一部分的 K 值在增大时有什么影响?

在 SVM 算法中，选择不同的核函数在二分类问题上有什么影响?

在二分类问题中，逻辑回归上的不同参数的缩放有什么影响?

在随机森林模型中，在训练集上增加任意属性对在分类准确性上有什么影响?

针对算法，设计你想回答的问题。仔细考虑，然后列出 5 个逐渐演变的问题，并且深入推敲那个最精确的

2.3.3 设计实验

从你的问题中挑选出关键元素然后组成你的实验内容。例如，拿上面的示例问题为例：“二元分类问题中逻辑回归上的不同的参数缩放有什么影响?”

你从这个问题中挑出来用来设计实验的元素是：

属性缩放法：你可以采用像正态化、标准化，将某一属性提升至乘方、取对数等方法

逻辑回归：你想使用哪种已经实现的逻辑回归。

二元分类问题：存在数值属性不同的二分类问题标准。需要准备多种问题，其中一些问题的规模是相同的（像电离层），然而其他一些问题的属性有不同的缩放值（像糖尿病问题）。

性能：类似分类准确性的模型性能分数是需要的

花时间仔细挑选你问题中的组成元素以便为你的问题给出最佳解答。

2.3.4 进行试验并且报告你的结论

完成你的实验

如果算法是随机的，你需要多次重复实验操作并且记录一个平均数和标准偏差

如果你试图寻找在不同实验（比如带有不同的参数）之间结果的差异，你可能想要使用一种统计工具来标明差异是否统计上显著的（就像学生的 t 检验）

一些工具像 R 和 scikit-learn/SciPy 完成这些类型的实验，但是你需要把它们组合在一起，并且为实验写脚本。其他工具像 Weka 带有图形用户界面，你所使用的工具不要影响问题和你实验设计的严密

总结你的实验结论。你可能想使用图表。单独呈现结果是不够的，他们只是数字。你必须将数字和问题联系起来，并且通过你的实验设计提取出它们的意义

对实验问题来说，实验结果又暗示着什么呢?

保持怀疑的态度。你的结论上有留什么样的漏洞和局限呢。不要逃避这一部分。知道局限性和知道实验结果一样重要

2.3.5 重复

重复操作

继续研究你选择的算法。你甚至想要重复带有不同参数或者不同的测试数据集的同一个实验。你可能想要处理你试验中的局限性

不要只停留在一个算法上，开始建立知识体系和对算法的直觉

通过使用一些简单工具，提出好的问题，保持严谨和怀疑的态度，你对机器算法行为的理解很快就会到达世界级的水平

2.4 研究学习算法不仅仅是学者才能做的

你也可以学习研究机器学习算法。

你不需要一个很高的学位，你不需要用研究的方式训练，你也不需要成为一名学者

对每个拥有计算机和浓厚兴趣的人来说，机器学习算法的系统研究学习是开放的。事实上，如果你主修机器学习，你一定会适应机器学习算法的系统研究。知识根本不会自己出来，你需要靠自己的经验去得到

当谈论你的发现的适用性时，你需要保持怀疑和谨慎

你不一定提出独一无二的问题。通过研究一般的问题，你也将会收获很多，例如根据一些一般的标准数据集总结出一个参数的普遍影响。你保不住会发现某些具有最优方法的常例的局限性甚至反例。

第三章　几种框架的比较

3.1　引言

人工智能无疑是计算机世界的前沿领域，而深度学习无疑又是人工智能的研究热点，那么现在都有哪些开源的深度学习工具，他们各自的优缺点又是什么呢？最近 zer0n 和 bamos 在 GitHub 上发表了一篇文章，对 Caffe、CNTK、TensorFlow、Theano 和 Torch 等深度学习工具从网络、模型能力、接口、部署、性能、架构、生态系统、跨平台等方面做了比较 [?]。

http://www.infoq.com/cn/news/2016/01/evaluation-comparison-deep-learn

几种深度学习框架的分析与比较（添加标注）

3.2　网络和模型能力

3.2.1　Caffe

*** 可能是第一个主流的工业级深度学习工具，它开始于 2013 年底，具有出色的卷积神经网络实现。在计算机视觉领域 Caffe 依然是最流行的工具包，它有很多扩展，但是由于一些遗留的架构问题，它对递归网络和语言建模的支持很差。此外，在 Caffe 中图层需要使用 C++ 定义，而网络则使用 Protobuf 定义。

3.2.2　CNTK

由深度学习热潮的发起演讲人创建，目前已经发展成一个通用的、平台独立的深度学习系统。在 CNTK 中，网络会被指定为向量运算的符号图，运算的组合会形成层。CNTK 通过细粒度的构件块让用户不需要使用低层次的语言就能创建新的、复杂的层类型。

3.2.3　TensorFlow

**** 是一个理想的 RNN（递归神经网络）API 和实现，TensorFlow 使用了向量运算的符号图方法，使得新网络的指定变得相当容易，但 TensorFlow 并不支持双向 RNN 和 3D 卷积，同时公共版本的图定义也不支持循环和条件控制，这使得 RNN 的实现并不理想，因为必须要使用 Python 循环且无法进行图编译优化。

3.2.4　Theano

***** 支持大部分先进的网络，现在的很多研究想法都来源于 Theano，它引领了符号图在编程网络中使用的趋势。Theano 的符号 API 支持循环控制，让 RNN 的实现更加容易且高效。

3.2.5 Torch

***** 对卷积网络的支持非常好。在 TensorFlow 和 Theano 中时域卷积可以通过 conv2d 来实现，但这样做有点取巧；Torch 通过时域卷积的本地接口使得它的使用非常直观。Torch 通过很多非官方的扩展支持大量的 RNN，同时网络的定义方法也有很多种。但 Torch 本质上是以图层的方式定义网络的，这种粗粒度的方式使得它对新图层类型的扩展缺乏足够的支持。与 Caffe 相比，在 Torch 中定义新图层非常容易，不需要使用 C++ 编程，图层和网络定义方式之间的区别最小。

3.3 接口

3.3.1 Caffe

*** 支持 pycaffe 接口，但这仅仅是用来辅助命令行接口的，而即便是使用 pycaffe 也必须使用 protobuf 定义模型。

3.3.2 CNTK

** */2 CNTK 的使用方式与 Caffe 相似，也是通过指定配置文件并运行命令行，但 CNTK 没有 Python 或者任何其他高级语言的接口。

3.3.3 TensorFlow

**** */2 TensorFlow 支持 Python 和 C++ 两种类型的接口。用户可以在一个相对丰富的高层环境中做实验并在需要本地代码或低延迟的环境中部署模型。

3.3.4 Theano

**** Theano 支持 Python 接口。

3.3.5 Torch

**** Torch 运行在 LuaJIT 上，`C++`、`C#` 以及 Java 等工业语言，相比速度非常快，用户能够编写任意类型的计算，不需要担心性能，唯一的问题就是 Lua 并不是主流的语言。

3.4 模型部署

Caffe 是基于 C++ 的，因此可以在多种设备上编译，具有跨平台性，在部署方面是最佳选择。

CNTK 与 Caffe 一样也是基于 C++ 并且跨平台的，大部分情况下部署非常简单。但是它不支持 ARM 架构，这限制了它在移动设备上的能力。

TensorFlow 支持 C++ 接口，同时由于它使用了 Eigen 而不是 BLAS 类库，所以能够基于 ARM 架构编译和优化。TensorFlow 的用户能够将训练好的模型部署到多种设备上，不需要实现单独的模型解码器或者加载 Python/LuaJIT 解释器。但是 TensorFlow 并不支持 Windows，因此其模型无法部署到 Windows 设备上。

Theano 缺少底层的接口，并且其 Python 解释器也很低效，对工业用户而言缺少吸引力。虽然对大的模型其 Python 开销并不大，但它的限制摆在那，唯一的亮点就是它跨平台，模型能够部署到 Windows 环境上。

Torch 的模型运行需要 LuaJIT 的支持，虽然这样做对性能的影响并不大，但却对集成造成了很大的障碍，使得它的吸引力不如 Caffe/CNTK/TensorFlow 等直接支持 C++ 的框架。

3.5 性能

在单 GPU 的场景下，所有这些工具集都调用了 cuDNN，因此只要外层的计算或者内存分配差异不大其表现都差不多。本文的性能测试是基于 Soumith@FB 的 ConvNets 基准测试来做的。

Caffe 简单快速。

CNTK 简单快速，在多 GPU 方面，CNTK 相较于其他的深度学习工具包表现更好，它实现了 1-bit SGD 和自适应的 minibatching。

TensorFlow 仅使用了 cuDNN v2，但即使如此它的性能依然要比同样使用 cuDNN v2 的 Torch 要慢 1.5 倍，并且在批大小为 128 时训练 GoogleNet 还出现了内存溢出的问题。

Theano 在大型网络上的性能与 Torch7 不相上下。但它的主要问题是启动时间特别长，因为它需要将 C/CUDA 代码编译成二进制，而 TensorFlow 并没有这个问题。此外，Theano 的导入也会消耗时间，并且在导入之后无法摆脱预配置的设备（例如 GPU0）。

Torch 非常好，没有 TensorFlow 和 Theano 的问题。

3.6 架构

Caffe 的架构在现在看来算是平均水准，它的主要痛点是图层需要使用 C++ 定义，而模型需要使用 protobuf 定义。另外，如果想要支持 CPU 和 GPU，用户还必须实现额外的函数，例如 Forward_gpu 和 Backward_gpu；对于自定义的层类型，还必须为其分配一个 int 类型的 id，并将其添加到 proto 文件中。

TensorFlow 的架构清晰，采用了模块化设计，支持多种前端和执行平台。

Theano 的架构比较变态，它的整个代码库都是 Python 的，就连 C/CUDA 代码也要被打包为 Python 字符串，这使得它难以导航、调试、重构和维护。

Torch7 和 nn 类库拥有清晰的设计和模块化的接口。

3.7 跨平台

Caffe、CNTK 和 Theano 都能在所有的系统上运行，而 TensorFlow 和 Torch 则不支持 Windows。

第四章　神经网络入门

文章参考:http://blog.csdn.net/zzwu/article/details/574931

4.1　生物学的大脑

大脑是一块灰色的、像奶冻一样的东西。它并不像脑中的 CPU 那样，利用单个或少数几个处理单元来进行工作。如果你有一具新鲜地保存到福尔马林中的尸体，用一把锯子小心地将它的头骨锯开，搬掉头盖骨后，你就能看到熟悉的脑组织皱纹。大脑的外层象一个大核桃那样，全部都是起皱的 [图 0 左]，这一层组织就称皮层 (Cortex)。如果你再小心地用手指把整个大脑从头颅中端出来，再去拿一把外科医生用的手术刀，将大脑切成片，那么你将看到大脑有两层: 灰色的外层 (这就是“灰质”一词的来源，但没有经过福尔马林固定的新鲜大脑实际是粉红色的。如图 4.1) 和白色的内层。灰色层只有几毫米厚，其中紧密地压缩着几十亿个被称作 neuron (神经细胞、神经元) 的微小细胞。白色层在皮层灰质的下面，占据了皮层的大部分空间，是由神经细胞相互之间的无数连接线组成（但没有神经细胞本身，正如印刷电路板的背面，只有元件的连线，而没有元件本身那样，译注)。皮层象核桃一样起皱，这可以把一个很大的表面区域塞进到一个较小的空间里。这与光滑的皮层相比能容纳更多的神经细胞。人的大脑大约含有 10G（即 100 亿）个这样的微小处理单元; 一只蚂蚁的大脑大约也有 250,000 个。

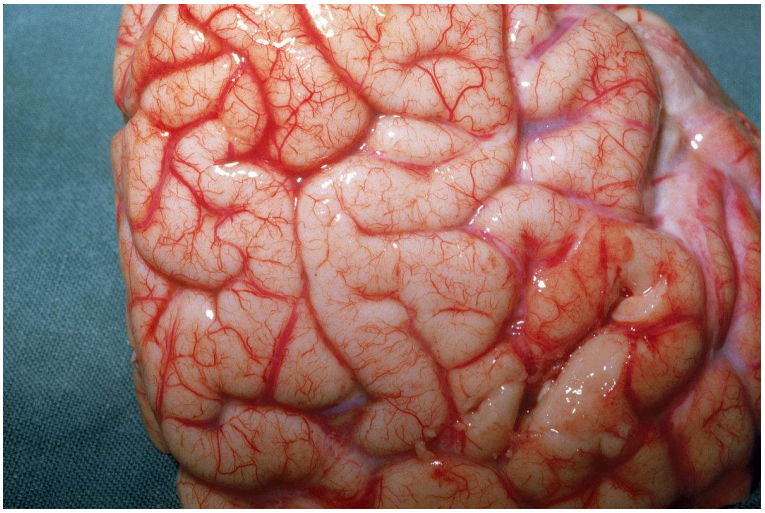

图 4.1: 大脑皮层解剖图

这是一个神经网络游戏的网站，很有意思。http://playground.tensorflowjiaocheng.com/#JJML

第五章　深度学习实例

5.1　CNN 实例

CNN（Convolutional Neural Network）是深度学习算法在图像处理领域的一个应用。利用卷积神经网络将一幅图像的内容与另一幅图像的风格相结合。

https://github.com/jcjohnson/neural-style