TensorFlow计算与AI基础

一、概述

AI的现在和未来

纵观古今，很少有计算机技术能有较长的发展寿命，大部分昙花一现，比如：昔日的DOS、 windows3.2、 foxpro、 delphi、 80x86汇编，还有很多技术也在艰难地挣扎，比如：VB、 PB、 Sqlserver，甚至连微软的.NET也被来自全球强大的开源力量逼到了死角，以至于不得不开放源码，向LINUX投出橄榄枝，才得以继续发展壮大。比如下面这条关于微软的新闻：

早在2016年，当微软宣布SQL Server将很快在Linux上运行时，这一消息对用户和权威人士来说都是一个巨大的惊喜。在过去的一年中，微软对Linux（总之就是开源的操作系统）的支持已经十分重视，公司的使命似乎已经变成了只要哪里有用户，哪里就必须得有SQL Server工具。微软近期发布了SQL Server 2017的第一个候选版本，这将是第一个在Windows、Linux和Docker容器上运行的版本。仅Docker容器就已经吸引了超过100万的申请，因此毫无疑问，会有很多人对这款新版本非常感兴趣。尽管新版本有很多需要改进的地方，但SQL Server 2017支持Linux仍然是这个版本最重要的突破。

2013年深秋，一个秋高气爽的日子，此时的大数据在中国已经开始蠢蠢欲动，HADOOP作为离线计算的大数据平台开始悄然风靡，机器学习开始在中国播下种子，但这些仅局限于大公司和高校内部，中国的大地上并没有充满人工智能的气息， 我应机械工业出版社的邀请，写了《机器学习实践指南》（该书已于2014年出版，目前是第二版，购买网址：）。写完这本书后，我有一种强烈的预感：人工智能与机器学习未来肯定必火，而且它肯定会成为计算机技术发展史上最长寿的技术，可能长达几百年甚至几千年，以至于人类移民到了其它星球，人工智能仍会繁荣昌盛~，但是，我万万没想到，出书不到3年，人工智能已经在中国四处开花，硕果累累，其发展速度，超出我的想象，呈奥达公

本文将带领大家初探TensorFlow这个机器学习系统。欢迎各位朋友访问我的博客。机器学习博大精深，需要入门的朋友可以购买笔者的《机器学习实践指南》（购买网址，源码下载地址：http://pan.baidu.com/s/1gf9e5NH 密码：v12g ），从实践上机操作这一途径打开机器学习的大门。

TensorFlow概述

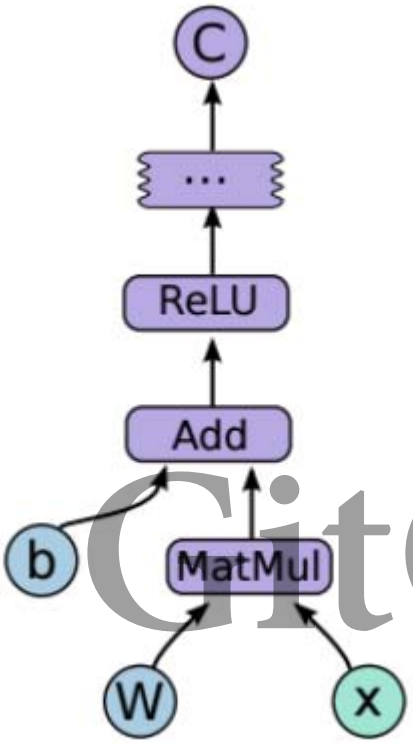
TensorFlow是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个CPU（或GPU），服务器，移动设备等等。TensorFlow 最初由Google大脑小组（隶属于Google机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但这个系统的通用性使其也可广泛用于其他计算领域。

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                    # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                        # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)              # Relu(Wx+b)
C = [...]                                           # Cost computed as a function
                                                    # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...        # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})           # Fetch cost, feeding x=input
  print step, result
```

下图是对上面这段程序的运算方式的解析：



观察上图，每个节点是运算，而箭头的起始端为圆圈的是数据。数据主要有以下几种类型：

- tf.constant：常量是在图执行过程是无法更改的。

简单加法

```
>>> import tensorflow as tf
>>> a=tf.constant([1.0,2.0],name="a")
>>> b=tf.constant([2.0,3.0],name="b")
>>> result=a+b
>>> sess=tf.Session()
>>> sess.run(result)
array([ 3.,  5.], dtype=float32)
```

生成新的计算图

下面程序生成了新的计算图，并完成常量初始化，然后在新的计算图中完成加法计算。

```
import tensorflow as tf
g1=tf.Graph()
with g1.as_default():
   value=[1.,2.,3.,4.,5.,6.]
   init = tf.constant_initializer(value)
   x=tf.get_variable("x",initializer=init,shape=[2,3])
   y=tf.get_variable("y",shape=
[2,3],initializer=tf.ones_initializer())
   result=tf.add(x,y,name="myadd")

with tf.Session(graph=g1) as sess:
   tf.global_variables_initializer().run()
   with tf.variable_scope("",reuse=True):
       print(sess.run(tf.get_variable("x")))
       print(sess.run(tf.get_variable("y")))
   print(sess.run(result))
```


输出结果：

```
>>> import tensorflow as tf
>>> n1 = tf.constant(3.0, dtype=tf.float32)
>>> n2 = tf.constant(4.0)
>>> n3=tf.constant(66)
>>> print(n1,n2,n3)
(<tf.Tensor 'Const:0' shape=() dtype=float32>, <tf.Tensor
'Const_1:0' shape=() dtype=float32>, <tf.Tensor 'Const_2:0'
shape=() dtype=int32>)
>>> n4=tf.constant(77,dtype=tf.int32)
>>> print(n1,n2,n3,n4)
(<tf.Tensor 'Const:0' shape=() dtype=float32>, <tf.Tensor
'Const_1:0' shape=() dtype=float32>, <tf.Tensor 'Const_2:0'
shape=() dtype=int32>, <tf.Tensor 'Const_3:0' shape=()
dtype=int32>)
>>> sess = tf.Session()
>>> print(sess.run([n1, n2]))
[3.0, 4.0]
```

上述程序的要点如下：

1. tf.Tensor就是张量，而张量是一个对象，是不能直接通过print语句输出值，需要通过sess.run语句来输出。
2. 使用tf.constant函数，创建常数。可以指定常数类型，也可以隐式指定。
3. 下面的语句输出常数对象print(n1,n2,no3)
4. 创建session，并生成计算图，然后，调用run方法。输出n1和n2的计算，比如下面代码：

   print(sess.run([n1, n2]))

   print(sess.run([n1, n2]*(n1+n2)))[21. 28.]

使用计算图编写并运行算法

1）求1到11的能被3或5整除的数及其和

```
def cond(i,num_sum,numbers):
    return tf.less(i,12)

def body(i,num_sum,numbers):

add_num=tf.cond(tf.logical_or(tf.equal(tf.mod(i,3),0),tf.equal(tf
.mod(i,5),0)),lambda:i,lambda:0)
    num_sum=tf.add(num_sum,add_num)
    numbers=tf.concat([numbers,[add_num]],axis=0)
    return i+1,num_sum,numbers


i,num_sum,numbers= tf.while_loop(cond,body,loop_vars=
[i,num_sum,numbers],shape_invariants=
[i.get_shape(),num_sum.get_shape(),tf.TensorShape([None])])
mask = tf.greater(numbers,0)
res = tf.boolean_mask(numbers, mask)
sum_result=tf.reduce_sum(res)
with tf.Session() as sess:
    tf.global_variables_initializer().run()
    sess.run([i,num_sum,numbers])
    print sess.run(res)
    print sess.run(sum_result)
    print sess.run(num_sum)
```


运行结果如下：

```
[ 3  5  6  9 10]
33
[33]
```

2）求某范围内的素数，以及非素数的因式分解

```
    if isfind:
        print n, 'is a prime number'
"""
import tensorflow as tf

SEARCH_RANGE=100
number=tf.constant(7,dtype=tf.int32)
x=tf.constant(2,dtype=tf.int32,name="x")
n=tf.Variable(2,dtype=tf.int32,name="n")
numbers=tf.Variable(tf.zeros([1,3], tf.int32))
non_prime=[]
prime=[]

def prime_cond(n,number,x,numbers):
    return tf.less(x,n)

def prime_body(n,number,x,numbers):
    num=tf.cond(tf.equal(tf.mod(n,x),0),lambda:
[n,x,tf.div(n,x)],lambda:[0,0,0])
    numbers=tf.concat([numbers,[num]],axis=0)
    return n,number,x+1,numbers

def is_primer(n,number,x,numbers):
    n,number,x,numbers=tf.while_loop(prime_cond,prime_body,\
                                     loop_vars=
[n,number,x,numbers],\
                                     shape_invariants=
[n.get_shape(),number.get_shape(),x.get_shape(),tf.TensorShape([N
one,3])])
    return n,number,x,numbers

init_assign = tf.global_variables_initializer()
res1=is_primer(n,number,x,numbers)
print "searching",
with tf.Session() as sess:
    sess.run(init_assign)
    for i in range(2, SEARCH_RANGE+1):
        print " "
```

```
            print "非素数：",non_p[0],"=",non_p[1],"*",non_p[2]
    print "素数：",
    print prime
```

运行结果如下：

```
searching . . . . . .
. . . . . .
. . . . . .
. . . . .
-----result-----
非素数：  4 = 2 * 2
非素数：  6 = 2 * 3
非素数：  6 = 3 * 2
非素数：  8 = 2 * 4
非素数：  8 = 4 * 2
非素数：  9 = 3 * 3
非素数：  10 = 2 * 5
非素数：  10 = 5 * 2
非素数：  12 = 2 * 6
非素数：  12 = 3 * 4
非素数：  12 = 4 * 3
非素数：  12 = 6 * 2
非素数：  14 = 2 * 7
非素数：  14 = 7 * 2
非素数：  15 = 3 * 5
非素数：  15 = 5 * 3
非素数：  16 = 2 * 8
非素数：  16 = 4 * 4
非素数：  16 = 8 * 2
非素数：  18 = 2 * 9
非素数：  18 = 3 * 6
非素数：  18 = 6 * 3
非素数：  18 = 9 * 2
非素数：  20 = 2 * 10
非素数：  20 = 4 * 5
```

```
非素数：  26 = 13 * 2
非素数：  27 = 3 * 9
非素数：  27 = 9 * 3
非素数：  28 = 2 * 14
非素数：  28 = 4 * 7
非素数：  28 = 7 * 4
非素数：  28 = 14 * 2
非素数：  30 = 2 * 15
非素数：  30 = 3 * 10
非素数：  30 = 5 * 6
非素数：  30 = 6 * 5
非素数：  30 = 10 * 3
非素数：  30 = 15 * 2
非素数：  32 = 2 * 16
非素数：  32 = 4 * 8
非素数：  32 = 8 * 4
非素数：  32 = 16 * 2
非素数：  33 = 3 * 11
非素数：  33 = 11 * 3
非素数：  34 = 2 * 17
非素数：  34 = 17 * 2
非素数：  35 = 5 * 7
非素数：  35 = 7 * 5
非素数：  36 = 2 * 18
非素数：  36 = 3 * 12
非素数：  36 = 4 * 9
非素数：  36 = 6 * 6
非素数：  36 = 9 * 4
非素数：  36 = 12 * 3
非素数：  36 = 18 * 2
非素数：  38 = 2 * 19
非素数：  38 = 19 * 2
非素数：  39 = 3 * 13
非素数：  39 = 13 * 3
非素数：  40 = 2 * 20
非素数：  40 = 4 * 10
非素数：  40 = 5 * 8
非素数：  40 = 8 * 5
```

```
非素数：  45 = 9 * 5
非素数：  45 = 15 * 3
非素数：  46 = 2 * 23
非素数：  46 = 23 * 2
非素数：  48 = 2 * 24
非素数：  48 = 3 * 16
非素数：  48 = 4 * 12
非素数：  48 = 6 * 8
非素数：  48 = 8 * 6
非素数：  48 = 12 * 4
非素数：  48 = 16 * 3
非素数：  48 = 24 * 2
非素数：  49 = 7 * 7
非素数：  50 = 2 * 25
非素数：  50 = 5 * 10
非素数：  50 = 10 * 5
非素数：  50 = 25 * 2
非素数：  51 = 3 * 17
非素数：  51 = 17 * 3
非素数：  52 = 2 * 26
非素数：  52 = 4 * 13
非素数：  52 = 13 * 4
非素数：  52 = 26 * 2
非素数：  54 = 2 * 27
非素数：  54 = 3 * 18
非素数：  54 = 6 * 9
非素数：  54 = 9 * 6
非素数：  54 = 18 * 3
非素数：  54 = 27 * 2
非素数：  55 = 5 * 11
非素数：  55 = 11 * 5
非素数：  56 = 2 * 28
非素数：  56 = 4 * 14
非素数：  56 = 7 * 8
非素数：  56 = 8 * 7
非素数：  56 = 14 * 4
非素数：  56 = 28 * 2
非素数：  57 = 3 * 19
```

```
非素数：  62 = 31 * 2
非素数：  63 = 3 * 21
非素数：  63 = 7 * 9
非素数：  63 = 9 * 7
非素数：  63 = 21 * 3
非素数：  64 = 2 * 32
非素数：  64 = 4 * 16
非素数：  64 = 8 * 8
非素数：  64 = 16 * 4
非素数：  64 = 32 * 2
非素数：  65 = 5 * 13
非素数：  65 = 13 * 5
非素数：  66 = 2 * 33
非素数：  66 = 3 * 22
非素数：  66 = 6 * 11
非素数：  66 = 11 * 6
非素数：  66 = 22 * 3
非素数：  66 = 33 * 2
非素数：  68 = 2 * 34
非素数：  68 = 4 * 17
非素数：  68 = 17 * 4
非素数：  68 = 34 * 2
非素数：  69 = 3 * 23
非素数：  69 = 23 * 3
非素数：  70 = 2 * 35
非素数：  70 = 5 * 14
非素数：  70 = 7 * 10
非素数：  70 = 10 * 7
非素数：  70 = 14 * 5
非素数：  70 = 35 * 2
非素数：  72 = 2 * 36
非素数：  72 = 3 * 24
非素数：  72 = 4 * 18
非素数：  72 = 6 * 12
非素数：  72 = 8 * 9
非素数：  72 = 9 * 8
非素数：  72 = 12 * 6
非素数：  72 = 18 * 4
```

```
非素数：  78 = 2 * 39
非素数：  78 = 3 * 26
非素数：  78 = 6 * 13
非素数：  78 = 13 * 6
非素数：  78 = 26 * 3
非素数：  78 = 39 * 2
非素数：  80 = 2 * 40
非素数：  80 = 4 * 20
非素数：  80 = 5 * 16
非素数：  80 = 8 * 10
非素数：  80 = 10 * 8
非素数：  80 = 16 * 5
非素数：  80 = 20 * 4
非素数：  80 = 40 * 2
非素数：  81 = 3 * 27
非素数：  81 = 9 * 9
非素数：  81 = 27 * 3
非素数：  82 = 2 * 41
非素数：  82 = 41 * 2
非素数：  84 = 2 * 42
非素数：  84 = 3 * 28
非素数：  84 = 4 * 21
非素数：  84 = 6 * 14
非素数：  84 = 7 * 12
非素数：  84 = 12 * 7
非素数：  84 = 14 * 6
非素数：  84 = 21 * 4
非素数：  84 = 28 * 3
非素数：  84 = 42 * 2
非素数：  85 = 5 * 17
非素数：  85 = 17 * 5
非素数：  86 = 2 * 43
非素数：  86 = 43 * 2
非素数：  87 = 3 * 29
非素数：  87 = 29 * 3
非素数：  88 = 2 * 44
非素数：  88 = 4 * 22
非素数：  88 = 8 * 11
```

```
非素数： 91 = 13 * 7
非素数： 92 = 2 * 46
非素数： 92 = 4 * 23
非素数： 92 = 23 * 4
非素数： 92 = 46 * 2
非素数： 93 = 3 * 31
非素数： 93 = 31 * 3
非素数： 94 = 2 * 47
非素数： 94 = 47 * 2
非素数： 95 = 5 * 19
非素数： 95 = 19 * 5
非素数： 96 = 2 * 48
非素数： 96 = 3 * 32
非素数： 96 = 4 * 24
非素数： 96 = 6 * 16
非素数： 96 = 8 * 12
非素数： 96 = 12 * 8
非素数： 96 = 16 * 6
非素数： 96 = 24 * 4
非素数： 96 = 32 * 3
非素数： 96 = 48 * 2
非素数： 98 = 2 * 49
非素数： 98 = 7 * 14
非素数： 98 = 14 * 7
非素数： 98 = 49 * 2
非素数： 99 = 3 * 33
非素数： 99 = 9 * 11
非素数： 99 = 11 * 9
非素数： 99 = 33 * 3
非素数： 100 = 2 * 50
非素数： 100 = 4 * 25
非素数： 100 = 5 * 20
非素数： 100 = 10 * 10
非素数： 100 = 20 * 5
非素数： 100 = 25 * 4
非素数： 100 = 50 * 2
素数： [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
```

是机器学习算法和数据存取技术的结合，利用机器学习提供的统计分析、知识发现等手段分析海量数据，同时利用数据存取机制实现数据的高效读写。

提示：在此普及一下两个概念： 1. 数据挖掘是识别出巨量数据中有效的、新颖的、潜在有用的、最终可理解的模式的非平凡过程。 2. 数据分析是指用适当的统计方法对收集来的大量第一手资料和第二手资料进行分析，以求最大化地开发数据资料的功能，发挥数据的作用，它是为了提取有用信息和形成结论而对数据加以详细研究和概括总结的过程”无论是数据分析还是数据挖掘，都是帮助人们收集、分析数据，使之成为信息，并作出判断。

人工智能与模式识别

人工智能不是一门编程技术，也不是数据结构与算法，人工智能是编程、数据结构、算法与数学的结合 。

我们开始来谈模式识别，它是起源于工程领域，而机器学习起源于计算机科学，这两个不同学科的结合带来了模式识别领域的调整和发展。模式识别研究主要集中在两个方面：一是研究生物体（包括人）是如何感知对象的，属于认识科学的范畴；二是在给定的任务下，如何用计算机实现模式识别的理论和方法，这些是机器学习的长项，也是机器学习研究的内容之一。

模式识别的应用领域广泛，包括计算机视觉、医学图像分析、光学文字识别、自然语言处理、语音识别、手写识别、生物特征识别、文件分类、搜索引擎等，而这些领域也正是机器学习的大展身手的舞台。

人工智能与数学

人工智能和机器学习的入门门槛较高，尤其对研究者的数学理解能力有较高要求，你肯定有话说，我懂数据结构、计算机算法，我做软件开发很多年了，我都是系统架构师了，都是项目管理师了，各种高端~

但我告诉你们，机器学习是一个全新的领域，也是一个全新的高度，理解机器学习算法往往要从理解它涉及的数学公式和数学知识开始，所有的AI从业者都必须通过攻克数学

靠代数学不能有效解决的问题。微积分学建立在代数学、三角学和解析几何学的基础上，包括微分学、积分学两大分支，包括连续、极限、多元函数的微积分、高斯定理等内容。

微积分在天文学、力学、化学、生物学、工程学、经济学、计算机科学等领域有着越来越广泛的应用，比如：在医疗领域，微积分能计算血管最优支角，将血流最大化；在经济学中，微积分可以通过计算边际成本和边际利润来确定最大收益；微积分可用于寻找方程的近似值；通过微积分解微分方程，计算相关的应用，比如，宇宙飞船利用欧拉方法来求得零重力环境下的近似曲线等。

在机器学习和数据分析领域，微积分是很多算法的理论基础，如：多层感知器神经网络算法。多层感知器是一种前馈人工神经网络模型，算法分为两个阶段：正向传播信号、反向传播误差。

正向传播信号阶段是对样本的学习阶段，输入的信息从输入层传入，经各个隐层计算后传至输出层，计算每个单元的实际值，向各层各单元分摊产生的误差；反向传播误差阶段通过网络输出与目标输出的误差对网络进行修改审查，将正向输出的误差再传播回各层进行权重值调整，直到误差最小化或达到规定的计算次数。微积分理论在多层感知器模型中运用较多。

2) 线性代数

线性代数是高等数学中的一门成熟的基础学科，它内容广泛，不但包含行列式、矩阵、线性方程组等初等部分，而且包括线性空间、欧式空间、酉空间、线性变换和线性函数、λ-矩阵、矩阵特征值等更深入的理论，线性代数在数学、物理学、社会科学、工程学等领域也有广泛的应用。

线性代数理论是计算技术的基础，在机器学习、数据分析、数学建模领域有着重要的地位，这些领域往往需要应用线性方程组、矩阵、行列式等理论，并通过计算机完成计算。

3) 概率论

概率论是研究随机性或不确定性现象的数学，用来模拟实验在同一环境下会产生不同结果的情况

统计学是收集、分析、表述和解释数据的科学，作为数据分析的一种有效工具，统计方法已广泛应用于社会科学和自然科学的各个领域。统计学与概率论联系紧密，前者以后者为理论基础。统计学主要分为描述统计学和推断统计学。描述统计学描绘或总结观察量的集中和离散情形，基础的数学描述包括了平均数和标准差等；推断统计学将资料中的数据模型化，计算它的机率并且做出对于母群体的推论，主要包括假设检定、对于数字特征量的估计、对于未来观察的预测、相关性预测、回归、变异数分析、时间序列、数据挖掘等。

无论是描述统计学还是推断统计学都是数据分析技术的基础。通过描述统计学方法，数据分析专家能对数据资料进行图像化处理，将资料摘要变为图表，分析数据分布特征。此外，还可以分析数据资料，以了解各变量内的观察值集中与分散的情况等。通过推断统计学方法，对数据未知特征做出以概率形式表述的推断，在随机抽样的基础上推论有关总体数量特征。

5）离散数学

离散数学是数学的几个分支的总称，研究基于离散空间而不是连续的数学结构，其研究内容非常广泛，主要包括数理逻辑、集合论、信息论、数论、组合数学、图论、抽象代数、理论计算机科学、拓扑学、运筹学、博弈论、决策论等。

离散数学广泛应用于机器学习、算法设计、信息安全、数据分析等领域，比如：数理逻辑和集合论是专家系统的基础，专家系统是一类具有专门知识和经验的计算机智能程序系统，一般采用人工智能中的知识表示和知识推理技术，模拟通常由领域专家才能解决的复杂问题；信息论、数论、抽象代数用于信息安全领域；与信息论密切相关的编码理论可用来设计高效可靠的数据传输和数据储存方法；数论在密码学和密码分析中有广泛应用，现代密码学的DES、RSA等算法技术（包括因子分解、离散对数、素数测试等）依赖于数论、抽象代数理论基础；运筹学、博弈论、决策论为解决很多经济、金融和其他数据分析领域的问题提供了实用方法，这些问题包括资源合理分配、风险防控、决策评估、商品供求分析等。

以上是机器学习需要的核心数学知识，但不是全部知识。随着今后人类对机器学习的深入研究，将有更多的数学分支进入机器学习领域。因此，仅掌握大学数学知识是不够的，还需要向更高层次进军，对于非数学专业毕业的朋友来说，还应该学习其他数学分支理论，比如说泛函分析、复变函数、偏微分方程、抽象代数、约束优化、模糊数学、

它们都是机器学习所涉及的经典数学书，可以考虑将它们和《设计模式》、《算法导论》、《深入理解计算机系统》等经典算法书放在一起，作为案头必备书。

历数这么多，只是人工智能需要懂得的数学的冰山一角。危机是来了，可是程序员们，为了AI，还是拼一把，好好学习数学吧~

一位MIT的牛人在BLOG中曾提到，数学似乎总是不够的，为了解决和研究工程中的一些问题，不得不在工作后，重新回到图书馆捧起了数学教科书。他深深感到，从大学到工作，课堂上学的和自学的数学其实不算少了，可是在机器学习领域总是发现需要补充新的数学知识。看来，要精通机器学习知识，必须在数学领域学习、学习、再学习，这一切都是很艰苦的。要学好机器学习必须做好艰苦奋斗的准备，坚持对数学知识的追求。

人工智能与AI框架

1）tensorflow

一个非常优秀的AI框架，注意，它不仅仅是一个深度学习库，它是一个AI架构。**让人惊喜的是：tensorflow已经可以跑在spark上了。**

tensorflow on spark:TensorflowOnSpark 支持使用Spark/Hadoop集群分布式的运行Tensorflow，号称支持所有的Tensorflow操作。TensorFlowOnSpark提供两种不同的模式来提取训练和推理数据：

1. TensorFlow QueueRunners ： TensorFlowOnSpark利用TensorFlow的file readers和QueueRunners直接从HDFS文件中读取数据。Spark不涉及访问数据。

2. Spark Feeding ：Spark RDD数据被传输到每个Spark执行器里，随后的数据将通过feed_dict传入TensorFlow图。

2）caffe

caffe是一个清晰，可读性高，快速的深度学习框架。
caffe on spark: CaffeOnSpark被设计成为一个Spark深度学习包，CaffeOnSpark API支持dataframes，以便易于连接准备使用Spark应用程序的训练数据集，以及提取模型的预测

TensorFlow从业者的发展路线

第0步，为什么说是第0步，因为这一步只是需要熟悉了争，不需要精通。需要熟悉的内容如下：

> 学习hadoop搭建与基础、zookeeper、mapreduce、hbase、hive、flume、kafka、spark dataframe、spark sql、spark streaming、scala基础。

为什么学TensorFlow需要学习hadoop和spark?因为 tensorflow on spark的数据来源就是HDFS和Spark RDD，而且TensorFlow本身也支持直接访问HDFS

第一步，学习数据结构与算法，用程序调用TensorFlow库。

第二步，使用大数据存取平台，存取数据，调用TensorFlow库。

第三步，学会调节参数，努力学习数学，调用TensorFlow库。

第四步，使用TensorFlow实现机器学习算法，理解其库的内部实现方式，必要时，直接编写AI算法和现有AI库一起工作。

第五步，更深入学习数学，把TensorFlow作为一个数值计算框架，迎接AI技术的不断发展和挑战。

四、TensorFlow与神经网络

权值、隐藏层、前向传播

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
```

```
    print sess.run(y,feed_dict={x:[[5.2,2.9],[3.9,1.1],[3.9,5.2],
[6.1,9.2]]})
```

h是隐藏层，w1,w2是权值。

基于反向传播的权值调整

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 10 09:35:04 2017

@author: myhaspl
"""
import tensorflow as tf
import numpy as np

batch_size=10
w1=tf.Variable(tf.random_normal([2,6],stddev=1,seed=1))
w2=tf.Variable(tf.random_normal([6,1],stddev=1,seed=1))
b=tf.Variable(tf.zeros([6]),tf.float32)

x=tf.placeholder(tf.float32,shape=(None,2),name="x")
y=tf.placeholder(tf.float32,shape=(None,1),name="y")


h=tf.nn.relu(tf.matmul(x,w1)+b)
yo=tf.nn.relu(tf.matmul(h,w2))

#损失函数计算差异平均值
cross_entropy=tf.reduce_mean(tf.abs(y-yo))
#反向传播
train_step=tf.train.AdamOptimizer().minimize(cross_entropy)

#生成200个随机样本
```

```
        #设定训练轮数
        TRAINCOUNT=5000
        for i in range(TRAINCOUNT):
            #每次递进选择一组
            start=(i*batch_size) % DATASIZE
            end=min(start+batch_size,DATASIZE)
            #开始训练
            sess.run(train_step,feed_dict=
    {x:x_[start:end],y:y_[start:end]})
            if i%1000==0:
                total_cross_entropy=sess.run(cross_entropy,feed_dict=
    {x:x_[start:end],y:y_[start:end]})
                print("%d 次训练之后，损失:%g"%
    (i+1,total_cross_entropy))
        print sess.run(w1)
        print sess.run(w2)
        print sess.run(b)

        #生成测试样本，仅进行前向传播验证：
        testyo=sess.run(yo,feed_dict={x:[[0.6,0.2],[0.3,0.9]]})
        myout=[int(testout>0.5) for testout in testyo]
        print myout
```

运行结果如下：


```
1 次训练之后，损失:0.673294
1001 次训练之后，损失:0.287716
2001 次训练之后，损失:0.200945
3001 次训练之后，损失:0.172289
4001 次训练之后，损失:0.156476
。。。。。。
[0, 1]
```

更多的机器学习算法与神经网络解析请阅读《机器学习实践指南》第二版。

1. 编程好的牛人，应加油补数学。
2. 数学好的牛人，应加油补编程。

附录：TensorFlow 与AI从业者的数学课程

数学基础课程

数学分析、高等代数、解析几何、常微分方程、统计初步、信息技术应用、近世代数、概率论、数据结构、复变函数、微分几何、实变函数、数学模型、拓扑学、偏微分方程、几何基础，还有一些选修课，比如数值分析、数值代数、运筹学、组合数学、小波分析、模糊数学、数学软件

数学提升课程

- 黎曼几何引论
- 李群与对称空间
- 微分拓扑
- 同调论
- 群论
- 交换代数
- 泛函分析
- 实分析
- 李群与李代数的表示
- 常微分方程定性理论
- 偏微分方程
- 动力系统
- 测度与积分
- 高等概率论
- 高等统计学
- 随机过程论

- 代数几何初步
- 组合数学
- 李代数
- 模形式与数论
- 黎曼曲面论
- 复分析
- 非线性泛函分析基础
- 经典力学的数学方法
- 黎曼几何
- 流形的几何与拓扑
- 纽结理论
- 复流形
- 群表示论
- 设计与编码理论
- 代数数论
- 几何分析
- 调和分析
- 复动力系统
- 多复变
- 常微分方程选讲
- 光滑遍历论
- 密码学
- 代数几何
- 动力系统
- 高等时间序列分析
- 随机分析
- 极限定理
- 抽样调查
- 非参数统计
- 生存分析与可靠性
- 序贯分析
- 随机点过程
- 多元统计分析

- 计算流体力学
- 差分方法
- 有限元方法
- 软件工程
- 控制系统
- 矩阵扰动分析
- 计算机代数
- 计算机图形学
- 运筹学
- 平行计算
- 混沌计算
- 神经网络基础
- 模型与

- 图象数据库
- 数字压缩方法
- 图象处理
- 图象处理的数学方法
- 小波分析
- 计算机安全性
- 程序设计语言原理
- 计算机代数算法
- 计算机数学
- 程序语言研究
- 数字信号处理
- 理论计算机科学基础
- 自动推理
- 数理逻辑高级课程

- 拓扑专题课程
- 代数专题课程
- 分析专题课程
- 方程专题课程
- 近代数学物理选讲
- 偏微分方程选讲
- 随机过程选讲
- 高等统计选讲
- 随机过程选讲II
- 高等统计选讲II
- 现代信息处理选讲
- 信息科学技术选讲
- 计算机理论选讲
- 人工智能选讲
- 软件理论与方法选讲
- 程序理论选讲
- 网络新技术选讲
- 金融数学专题课程

- 微分几何
- 拓扑学
- 低维流形
- 调和分析
- 微分动力系统
- 向量场分支理论
- 小波分析
- 偏微分方程
- 复分析
- 非线性分析
- 几何分析
- 有限群论

- 动力系统与遍历论
- 算子代数与共形场
- 无穷维李群表示和算子代数
- 代数与优化
- 有限元方法
- 科学与工程计算
- 矩阵计算及其应用
- PDE
- 软件方法与高级语言
- 信号与信息处理
- 计算机科学中的逻辑
- 吴方法
- 极限定理
- 多元统计软件
- 抽样调查
- 随机过程
- 图模型分析
- 定性数据统计分析
- 粒子系统
- 现代统计方法
- 生存分析与可靠性
- 智能计算
- 生物信息
- 时间序列谱分析
- 金融与时间序列
- 稳健分析
- 删失与失效数据分析
- 精算与风险分析
- 保险与决策
- 精算理论与方法