Tensorflow预研

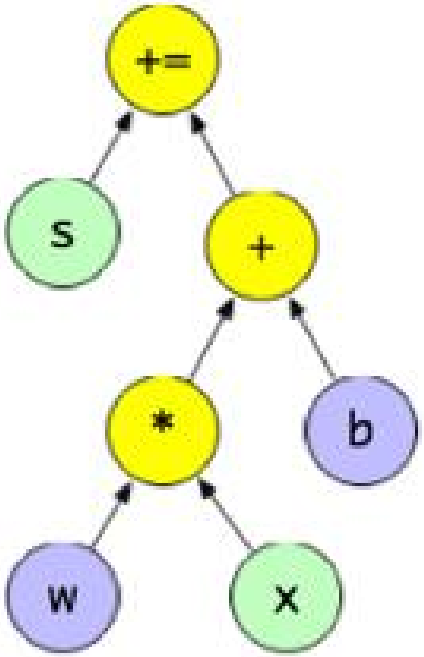
赵海臣

Tensorflow是什么

- Tensorflow是一个采用计算图的形式表述数值计算的编程系统，本身是一个开源软件库。Tensorflow计算图中每一个节点表示一次数学计算，每一条边表示计算之间的依赖关系。
- Tensor是计算图的基本数据结构，可以理解为多维数据，Flow表达了张量之间通过计算互相转化的过程。
- 它灵活的架构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。

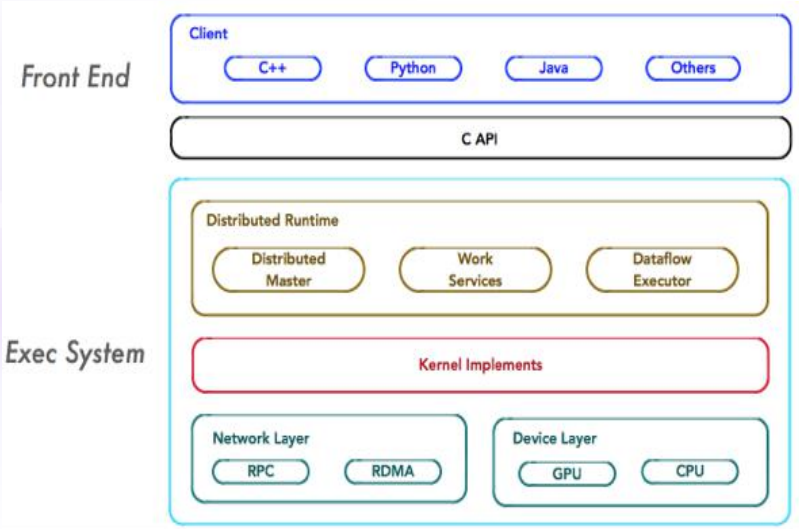
TensorFlow是什么

- Tensor张量
  - 是一个物理量，对高维的物理量进行“量纲分析”的一种工具。可以简单理解为：一维数组称为矢量， 二维数据为二阶张量，三维数组为三阶张量...
- 计算图
  - 用”结点”(nodes)和“线”(edges)的有向图来描述数学计算的图像。“节点”一般用来表示施加的数学操作，但也可以表示数据输入(feed in)的起点/输出(push out)的终点。“线”表示“节点”之间的输入/输出关系。这些数据“线”可以输运“size可动态调整”的多维数据数组，即“张量”(tensor)



TensorFlow架构

- TensorFlow的系统结构以C API为界，将整个系统分为“前端”和“后端”两个字系统
  - 前端系统(Front End)
    - 提供多语言编程环境，提供统一的编程模型支撑用户构造计算图
    - 通过Session的形式，连接TensorFlow后端的“运行时”，启动计算图的执行过程
  - 后端系统(Exec System)
    - 提供运行时环境，负责执行计算图



TensorFlow基本使用

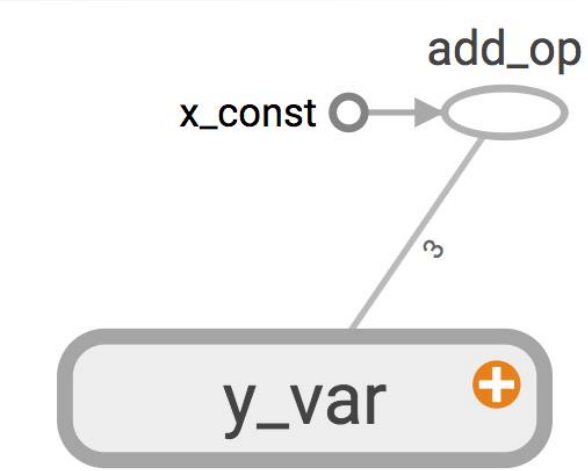
- 使用tensor表示数据
- 使用变量(Variable)输入训练数据，维护状态
- 使用计算图(computational graph)来表示计算任务
- 在会话(Session)的上下文(context)中执行计算图

TensorFlow基本使用——Tensor

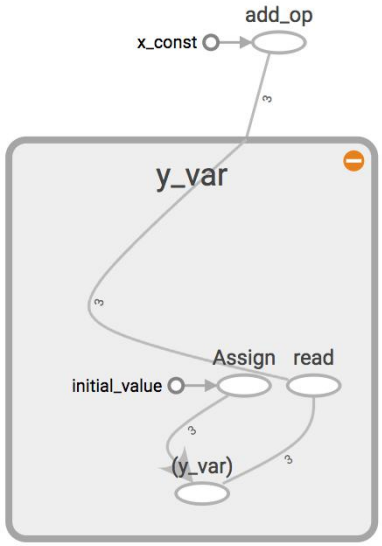
- Tensor是TensorFlow中的核心单元，TensorFlow程序使用tensor数据结构来代表所有的数据，计算图中，操作间传递的数据都是tensor。你可以把TensorFlow tensor看作是一个n维的数组或列表，如同矩阵一样
- 导入tensorflow
  - import tensorflow as tf
- 使用Tensor
  - node1 = tf.constant(3.0, tf.float32)
  - node2 = tf.constant([1.0, 2.0])
  - node3 = tf.constant([[1,2],[3,4],[5,6]], name='node3')

TensorFlow基本使用——计算图

- 计算图是用图中节点呈现一系列操作的图表。包括：
  - 构建计算图
  - 运行计算图
- 构建简单的计算图，每个节点将零个或多个tensor作为输入，产生一个tensor作为输出。一个典型的节点为常量，他将被tensorflow内部存储起来：
  - import tensorflow as tf
  - input1 = tf.constant([1.0,2.0,3.0], name=”x_const”)
  - input2 = tf.Variable([3.0,4.0,5.0], name='y_var')
  - output = tf.add(input1, input2, name=”add_op”)



TensorFlow基本使用——计算图



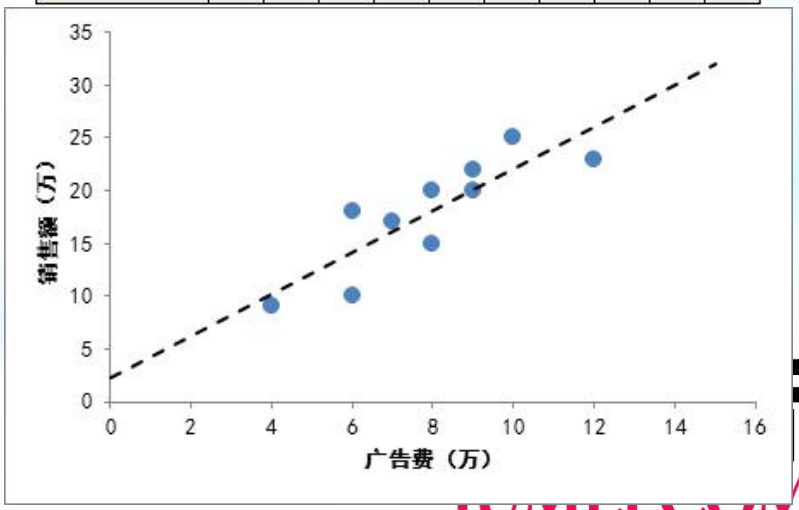
- 计算图是用图中节点呈现一系列操作的图表。包括：
  - 构建计算图
  - 运行计算图
- 构建的计算图必须在tensorflow的session中才能运行：
  - import tensorflow as tf
  - # 构建计算图
  - input1 = tf.constant([1.0,2.0,3.0], name=”x_const”)
  - input2 = tf.Variable([3.0,4.0,5.0], name='y_var')
  - output = tf.add(input1, input2, name=”add_op”)
  - # Session, 运行计算图前创建session
  - sess = tf.Session()
  - # 调用sess的run()来执行矩阵乘法op, feed用来传入标量数据
  - # 返回值”result”是一个numpy对象
  - result = sess.run(output, feed_dict{input2: [6.0,6.0,6.0]})
  - sess.close()

一元线性回归

- 回归分析
  - 确定两种或者两种以上变量间相互依赖的定量关系的一种统计分析方法
- 一元线性回归
  - 只涉及一个自变量和一个应变量
  - 应变量和自变量呈线性关系
  - 应变量与自变量的关系可以用一个线性方程表示：

$$\hat{y} = \omega x + b$$

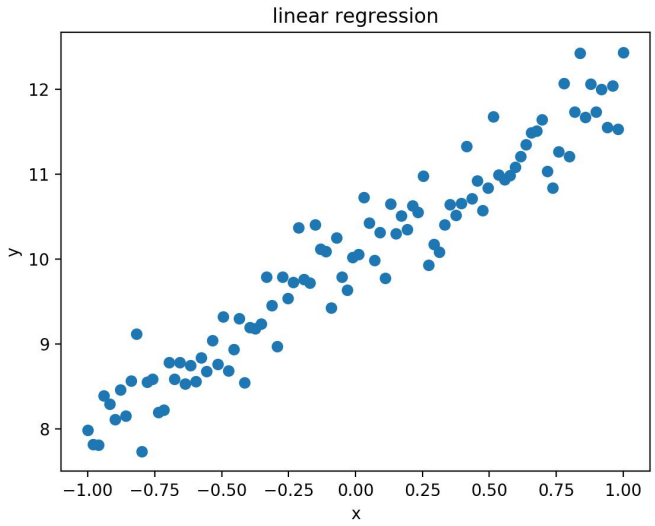
广告费（万）	4	8	9	8	7	12	6	10	6	9
销售额（万）	9	20	22	15	17	23	18	25	10	20



一元线性回归

- 最小二乘
  - 对于第i个应变量x_i估计值 $\hat{y}_i = \omega x_i + b$
  - 对于每一个x_i，估计值$\hat{y}_i$与实际值y_i距离的平方和
  - $$Q = \sum (y_i - \hat{y}_i)^2 = \sum (y_i - \omega x_i - b)^2$$
  - 要达到最小，则对ω和b求偏导数，Q的极小值点在偏导数为0的时候取得
- 梯度下降算法
  - 利用计算机求极值，可以用梯度下降算法进行多次迭代，逼近极值点，求出ω和b

一元线性回归



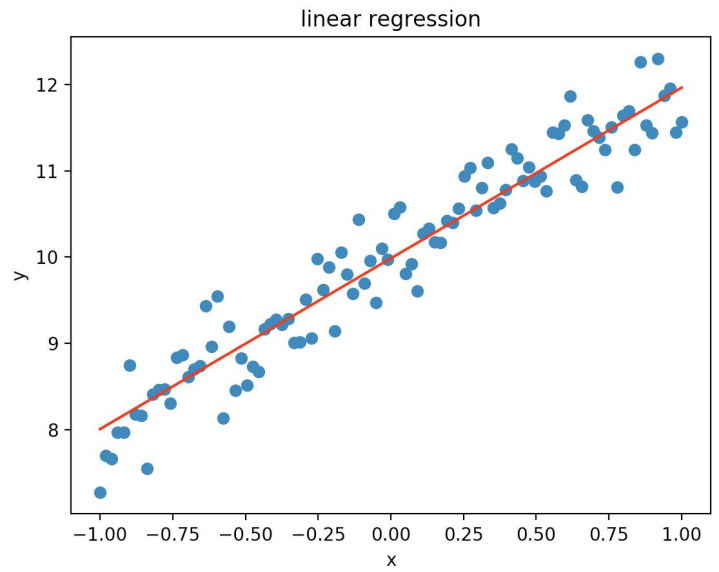
- 100个散点样本，求出回归方程

```
import tensorflow as tf
import numpy as np
train_X = np.linspace(-1,1,100)
train_Y = 2*train_X + \
          np.random.rand(*train_X.shape)*0.33 +10
# Construct Flow
X = tf.placeholder(tf.float32, name="x")
Y = tf.placeholder(tf.float32, name="y")
w = tf.Variable(0.0, name="weight")
b = tf.Variable(0.0, name="bias")
loss = tf.square(Y - X * w - b)
train_op = tf.train.GradientDescentOptimizer(0.01).\
           minimize(loss)
```

一元线性回归

- 100个散点样本，求出回归方程
- 运行计算图

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(20):
        for (x,y) in zip(train_X, train_Y):
            op_result, w_result, b_result = \
                sess.run([train_op, w, b], feed_dict={X:x,Y:y})
```




```
Epoch: 14, w: 1.95590674877, b: 10.0173501968
Epoch: 15, w: 1.95616412163, b: 10.0172538757
Epoch: 16, w: 1.95629501343, b: 10.0172052383
Epoch: 17, w: 1.95636260509, b: 10.0171775818
Epoch: 18, w: 1.95639693737, b: 10.017162323
Epoch: 19, w: 1.95641410351, b: 10.0171556473
Epoch: 20, w: 1.95642268658, b: 10.0171527863

Process finished with exit code 0
```

THE END

THANK YOU!