**TensorFlow GPU**

一、TensorFlow 设备分配

1、设备分配规则

If a TensorFlow operation has both CPU and GPU implementations, the GPU devices will be given priority when the operation is assigned to a device。

2、手动指定设备分配

如果你不想让系统自动为 operation 分配设备, 而是自己手动指定, 可以用 with tf.device()创建一个设备环境, 这个环境下的 operation 都统一运行在指定的设备上。

"/cpu:0"：机器的 CPU;

"/device:GPU:0"：机器的 GPU（如果有一个）；

"/device:GPU:1"：机器的第二个 GPU（以此类推）；

# op 在 cpu 上运算

with tf.device('/cpu:0'):

a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')

b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')

# op 在 gpu 上运算

with tf.device('/device:GPU:0'):

a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')

b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')

二、TensorFlow GPU 配置

要找出您的指令和张量被分配到哪个设备，请创建会话并将 log\_device\_placement 配置选项设为 True： sess = tf.Session(config=tf.ConfigProto(log\_device\_placement=True))

1、半遮面——指定可以被看见的GPU设备

# 默认情况，TF 会占用所有 GPU 的所有内存, 我们可以指定

# 只有 GPU0 和 GPU1 这两块卡被看到，从而达到限制其使用所有GPU的目的

import os

os.environ['CUDA\_VISIBLE\_DEVICES'] = '0, 1'

# 打印 TF 可用的 GPU

print os.environ['CUDA\_VISIBLE\_DEVICES']

>>> 0, 1

2、圈其地——限定使用显存的比例

# 在开启对话session前，先创建一个 tf.ConfigProto() 实例对象

# 通过 allow\_soft\_placement 参数自动将无法放在 GPU 上的操作放回 CPU

gpuConfig = tf.ConfigProto(allow\_soft\_placement=True)

# 限制一个进程使用 60% 的显存

gpuConfig.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.6

# 把你的配置部署到session

with tf.Session(config=gpuConfig) as sess:

pass

这样，如果你指定的卡的显存是8000M的话，你这个进程只能用4800M。

3、任其行——需要多少拿多少

# 在开启对话session前，先创建一个 tf.ConfigProto() 实例对象

# 通过 allow\_soft\_placement 参数自动将无法放在 GPU 上的操作放回 CPU

gpuConfig = tf.ConfigProto(allow\_soft\_placement=True)

# 运行时需要多少再给多少

gpuConfig.gpu\_options.allow\_growth = True

# 把你的配置部署到session

with tf.Session(config=gpuConfig) as sess:

pass

4、GPU 使用总结

import os

os.environ['CUDA\_VISIBLE\_DEVICES'] = '0, 1'

gpuConfig = tf.ConfigProto(allow\_soft\_placement=True)

gpuConfig.gpu\_options.allow\_growth = True

with tf.Session(config=gpuConfig) as sess:

pass

**TensorFlow教程**

先上代码：

import tensorflow as tf

import numpy as np

#created data

x\_data=np.random.rand(100).astype(np.float32)

y\_data=x\_data\*0.1+0.3

###create tensorflow structure start###

weights=tf.Variable(tf.random\_uniform([1],-1.0,1.0))

biases=tf.Variable(tf.zeros([1]))

y=weights\*x\_data+biases

loss=tf.reduce\_mean(tf.square(y-y\_data))

optimizer=tf.train.GradientDescentOptimizer(0.5)

train=optimizer.minimize(loss)

init=tf.global\_variables\_initializer()

###create tensorflow structure end ###

sess=tf.Session()

sess.run(init) #very important 可以把run当做指针

for step in range(200):

sess.run(train)

if step%20==0:

print(step,sess.run(weights),sess.run(biases))

模型搭建模式：

1. impute：数据，预处理
2. 定义、构建网络结构(图)
3. 初始化网络(图)：sess.run(init)；

训练网络(图)：sess.run(train)

网络(图)输出：print(step,sess.run(weights),sess.run(biases))

**Session：会话控制**

sess.run(op):输出指定节点OP的计算结果；

import tensorflow as tf

import numpy as np

matrix1=tf.constant([[3,3]])

matrix2=tf.constant([[2],

[2]] )

product=tf.matmul(matrix1,matrix2) #matrix multiply **等同于np.dot(m1,m2)**

#method1

sess=tf.Session()

result=sess.run(product)

print(result)

sess.close() #会话关闭

#method2

with tf.Session() as sess:

result2=sess.run(product)

print(result2)

**Variable：网络(图)变量**

TensorFlow中需要通过tf.Variable()定义变量

import tensorflow as tf

import numpy as np

state=tf.Variable(0,name='counter')

# print(state.name)

one=tf.constant(1)

new\_value=tf.add(state,one)

update=tf.assign(state,new\_value) #将new\_value加载更新state

**#网络含有变量的时候，一定要初始化变量**

init=tf.global\_variables\_initializer() #must have if define variable

with tf.Session() as sess:

sess.run(init)

for i in range(3):

sess.run(update) #网络(图)训练

print(sess.run(state)) #输出指定节点OP

**Placeholder：占位符**

**TensorFlow模型：训练特征提取**

**Tensorflow：如何保存/恢复模型？汇总整理**

<https://vimsky.com/article/3614.html>

TensorFlow教程（1）

常用函数API：

（1）

**tf.multiply(a,b)：**矩阵a和矩阵b的shape必须相等。 tf. multiply ()是矩阵的element-wise相乘（即Hadamard乘积）

**tf.matmul(a,b)：**矩阵a和矩阵b的shape应是a的行数对等与b的列数。tf.matmul()是矩阵的一般相乘。

tf.argmax(a, 1) 与 tf.multinomial(a,1) 的返回值比较

**tf.argmax(input, axis=None, name=None, dimension=None)**：

返回矩阵input在轴axis1上的最大值所对应的序列号

**tf.multinomial(a,1)**：根据分布概率的大小随机返回矩阵a在轴1上的序列号。

**tf.reduce\_sum(input\_tensor,reduction\_indices=None, keep\_dims=False, name=None)**

**#解释：input\_tensor为tensor即说明可以是矩阵形式**

import tensorflow as tf

A = [[2],[3],[1]]

B = [[1],[3],[4]]

with tf.Session() as sess:

print(sess.run(tf.reduce\_sum(**[A,B]**,0)))

tf.app.flags.FLAGS模块----flags 配置

（2）

**TensorBoard可视化助手**

1. <https://blog.csdn.net/smf0504/article/details/56369758>）

TensorBoard可以将训练过程中的各种绘制数据展示出来，包括标量（scalars），图片（images），音频（Audio）,计算图（graph）,数据分布，直方图（histograms）和嵌入式向量。

使用TensorBoard展示数据，需要在执行Tensorflow计算图的过程中，将各种类型的数据汇总并记录到日志文件中。然后使用TensorBoard读取这些日志文件，解析数据并生产数据可视化的Web页面，让我们可以在浏览器中观察各种汇总数据。

summary\_op包括了summary.scalar、summary.histogram、summary.image等操作，这些操作输出的是各种summary protobuf，最后通过summary.writer写入到event文件中。

Tensorflow API中包含系列生成summary数据的API接口，这些函数将汇总信息存放在protobuf（protocol buffer数据存储格式，类似于XML，JSON）中，以字符串形式表达。

数据集地址data\_dir和汇总数据的日志存放路径log\_dir

1.对标量数据汇总和记录使用tf.summary.scalar，一般在画loss,accuary时会用到这个函数。函数格式如下：

tf.summary.scalar(tags, values, collections=None, name=None)

2.使用tf.summary.histogram直接记录变量var的直方图，输出带直方图的汇总的protobuf，函数格式如下：

tf.summary.histogram(tag, values, collections=None, name=None）

3.输出带图像的probuf，汇总数据的图像的的形式如下： ' tag /image/0', ' tag /image/1', etc.，如：input/image/0等。

tf.summary.image(tag, tensor, max\_images=3, collections=None, name=None)

4.将上面几种类型的汇总再进行一次合并，具体合并哪些由inputs指定，格式如下：

tf.summary.merge(inputs, collections=None, name=None)

5、tf.summary.text

可以将文本类型的数据转换为tensor写入summary中：

例如：

text = """/a/b/c\\\_d/f\\\_g\\\_h\\\_2017"""

summary\_op0 = tf.summary.text('text', tf.convert\_to\_tensor(text))

6.合并默认图形中的所有汇总：

tf. summary.merge\_all(key='summaries')

7.将汇总的protobuf写入到event文件中去的相关的类： SummaryWriter是一个类，它可以调用以下成员函数来往event文件中添加相关的数据 add\_summary(), add\_session\_log(), add\_event(), or add\_graph()。

tf.summary.FileWriter

计算图形的信息通过add\_graph写入到event文件中

**TensorBorad如何使用**