Quiz

Tensorflow

Data types

A = Tf.constant(2)

Tf.zeros((3,3))

Tf.ones((3,3))

Tf.fill((3,3,),20)

With tf.Session() as sess1:

Print(sess1.run©)

Tf.linspace(1.0,10,0,20) # linspace takes start, end and number of values while

Tf.range(1,10,0.5). # range takes start, limit, delta. The limit is not included.

Tf.placeholder(tf.float32,shape=(2,2)). Feed\_dict= {a:ra}))

tf.matmul(a, a)

w= tf.variable(tf.constant([2,3,5],name=”weights”))

m=tf.add(w,biases)

init\_all\_op = tf.global\_variables\_initializer()

with tf.Session() as sess:

sess.run(init\_all\_op)

sess=tf.InteractiveSession()

m2\_out=m1.eval()

sess.close()

mathematical operations

tf.truediv(t1,t2) element by element division which return floating point

tf.floordiv(t1,t2) element by element division which return integer point

tf.transpose(t2,perm=[1,0]) transpose t2, so that we can perform matrix multiplication

tf.matmul(t1,t3) matrix multiplication

tf.matrix\_inverse(t1)

tf.reduce\_sum(t1)

tf.reduce\_prod(t2)

tf.reduce\_min(t2)

tf.accumulate\_n(t1,t2,t1) element by element sum of t1+t2+t3

tf.cumsum(t1,t2,t3) cumulative sum

tf.cumprod(t1,t2,t3) cumulative product

reluval = tf.cond(x<=0,lambda:zeroval, lambda:x ) # tf.cond works like if-else statement and allows branching in a graph

transformations

tf.cast(t1,tf.float32) cast the invidual elements from int32 to float32

tf.to\_float(t1)

shapes

tf.size(t1) #size gives the number of elements

tf.shapes(t1) #shape gives the size of each dimension

tf.reshape(t1, (12,5,6) #reshape tensor

t2.squeeze(t1) #remove diemension of size 1

tf.slice(t1,[2,0],[1,2])

inspection

tf.set\_random\_seed(3423)

tf.random\_uniform(shape=(5,), minval=1,maxval=59,dtype=tf.int32)

tf.concat([mainno,lastno,mainno],0)

print(dir(mainno))

simple\_examples

tf.random\_uniform(shape=(5,), minval=1,maxval=59,dtype=tf.int32, seed=1) to generate the same sample value

tf.set\_random\_seed(2323)

numerator\_np[1::2] = -1.0 change the alternate value -1

Numpy

np.zeros((3, 3))

np.ones((3,3))

np.linspace(1.0,10.0,20)

np.arange(10) creating a simple array

np.add(a,3)

b.shape #size of the matrix

b.ndim #number of dimensions

b.dtype #data type of each element

b.itemsize #memory occupied by each element

b.tolist() #convert ndarray to list

b.to\_file(file=”data.csv”, sep=”,”, format= “%0.3f”)

b.dump(“data.pickle”) #write to a pickle file

np.save(“b.npy”, [a,b]) write a numpy file

arrayshape

a.reshape((2,4),order=”C”). or order=”F” # Change the matrix to size 2x4 or with F

C [[0 1 2 3]. F [[0 2 4 6]

[4 5 6 7]] [1 3 5 7]]

a.resize((4,2))

a.flatten()

a.ravel() convert 1d vector. #ravel is not copy, any change to ravel affects a

a.copy() deep copy

array manipulation

a.sort()

a.nonzero()

b.any()

b.sum()

b.prod()

b.max()

np.arange(start=10, stop=20, step=2, dtype=np.float32)

d= np.zeroes(shape=(3,4), dtype=np.int32)

d= np.ones(shape=(3,4), dtype=np.int32)

np.where(g>2)) #returns x y coordinates

np.histogram(m)

np.linspace(start=1,stop=20,num=10)

operations

d-c subtract

10\*c multiply