Week 3d - Introduction to programming frameworks

Friday, August 14, 2020 11:30 PM

① Deep Learning Frameworks

→ Deep Learning Frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- Tensorflow
- Torch

→ Choosing frameworks

- ease of programming (development & deployment)
- Running speed
- Truly open (open source with good governance)

② Tensorflow

→ Problem : minimize $J(w) = w^2 - 10w + 25$

$(w-5)^2$

$\therefore w = 5$ is optimum

→ with Inputs specified

```
[1] %tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
[2] import numpy as np
    import tensorflow as tf
```

```
w = tf.Variable(0,dtype=tf.float32)
#cost = tf.add(tf.add(w**2,tf.multiply(-10.,w)),25)
cost = w**2 - 10*w + 25
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

init = tf.global_variables_initializer()
session = tf.Session()
session.run(init)
print(session.run(w))
```

→ allows tensorflow to construct computation graph & automatically compute back prop functions

```
0.0
```

```
[13] session.run(train) #1 step of gradient descent
     print(session.run(w))
```

```
0.099999994
```

```
[14] for i in range(1000): #1000 iterations
       session.run(train)
     print(session.run(w))
```

```
4.999988
```

→ with inputs fed as coefficients

```
[1] %tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
[2] import numpy as np
    import tensorflow as tf
```

```
[20] coefficients = np.array([[1.], [-20.], [100.]])

     w = tf.Variable(0,dtype=tf.float32)
     x = tf.placeholder(tf.float32, [3,1])
     #cost = tf.add(tf.add(w**2,tf.multiply(-10.,w)),25)
     #cost = w**2 - 10*w + 25
     cost = x[0][0]*w**2 + x[1][0]*w + x[2][0]
     train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

     init = tf.global_variables_initializer()
     session = tf.Session()
     session.run(init)
     print(session.run(w))
```

→ variable whose value can be supplied later

with tf.Session as session:
session.run(init)
Print(session.run(w)) //alt.syntax

```
0.0
```

```
[21] session.run(train, feed_dict={x:coefficients}) #1 step of gradient descent
     print(session.run(w))
```

↳ feed mini-batches here

```
0.19999999
```

```
[22] for i in range(1000): #1000 iterations
       session.run(train, feed_dict={x:coefficients})
     print(session.run(w))
```

```
9.999976
```