In [ ]:
```
import numpy as np
import tensorflow as tf
import pandas as pd
import time
from datetime import datetime
```

In [4]:
```
df1=pd.read_csv("C:/Users/Harsh Patel/Desktop/Mini Project/dataset/final_data.csv")
```

In [5]:
```
 epoch = datetime(1970, 1, 1)

def mapdateTotime(x):
    try:
        dt = datetime.strptime(x, "%m/%d/%Y")
    except ValueError:
        dt = datetime.strptime(x, "%Y-%m-%dT%H:%M:%S.%fZ")
    diff = dt - epoch
    return diff.total_seconds()

df1.Date = df1.Date.apply(mapdateTotime)
```

In [6]:
```
col1 = df1[['Date','Latitude','Longitude','Depth']]
col2 = df1['Magnitude']
#Convert to Numpy array
InputX1 = col1.as_matrix()
InputY1 = col2.as_matrix()
print(InputX1)
```

```
[[-1.5508800e+08  1.9246000e+01  1.4561600e+02  1.3160000e+02]
 [-1.4999040e+08  1.8630000e+00  1.2735200e+02  8.0000000e+01]
 [-1.4739840e+08 -2.0579000e+01 -1.7397200e+02  2.0000000e+01]
 ...
 [ 1.4828832e+09  3.6917900e+01  1.4042620e+02  1.0000000e+01]
 [ 1.4829696e+09 -9.0283000e+00  1.1866390e+02  7.9000000e+01]
 [ 1.4830560e+09  3.7397300e+01  1.4141030e+02  1.1940000e+01]]

C:\Anaconda\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: Method
.as_matrix will be removed in a future version. Use .values instead.
  after removing the cwd from sys.path.
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:5: FutureWarning: Method
.as_matrix will be removed in a future version. Use .values instead.
  """
```

In [7]:
```
#Min-max Normalization
X1_min = np.amin(InputX1,0)
X1_max = np.amax(InputX1,0)
print("Mininum values:",X1_min)
print("Maximum values:",X1_max)
Y1_min = np.amin(InputY1)
Y1_max = np.amax(InputY1)
InputX1_norm = (InputX1-X1_min)/(X1_max-X1_min)
InputY1_norm = InputY1  #No normalization in output

#Reshape
Xfeatures = 3 #Number of input features
Yfeatures = 1 #Number of output features
samples = 23000 # Number of samples

InputX1_reshape = np.resize(InputX1_norm,(samples,Xfeatures))
InputY1_reshape = np.resize(InputY1_norm,(samples,Yfeatures))
```

```
Mininum values: [-1.57680e+08 -7.70800e+01 -1.79997e+02 -1.10000e+00]
Maximum values: [1.483056e+09 8.600500e+01 1.799980e+02 7.000000e+02]
```

In [8]:
```
#Training data
batch_size = 2000
InputX1train = InputX1_reshape[0:batch_size,:]
InputY1train = InputY1_reshape[0:batch_size,:]
#Validation data
v_size = 2500
InputX1v = InputX1_reshape[batch_size:batch_size+v_size,:]
InputY1v = InputY1_reshape[batch_size:batch_size+v_size,:]
```

In [9]:
```
learning_rate = 0.001
training_iterations = 1000
display_iterations = 200
```

In [10]:
```
#Input
X = tf.placeholder(tf.float32,shape=(None,Xfeatures))
#Output
Y = tf.placeholder(tf.float32)
```

In [11]:
```
#Neurons
L1 = 3
L2 = 3
L3 = 3

#Layer1 weights
W_fc1 = tf.Variable(tf.random_uniform([Xfeatures,L1]))
b_fc1 = tf.Variable(tf.constant(0.1,shape=[L1]))

#Layer2 weights
W_fc2 = tf.Variable(tf.random_uniform([L1,L2]))
b_fc2 = tf.Variable(tf.constant(0.1,shape=[L2]))

#Layer3 weights
W_fc3 = tf.Variable(tf.random_uniform([L2,L3]))
b_fc3 = tf.Variable(tf.constant(0.1,shape=[L3]))

#Output layer weights
W_fO= tf.Variable(tf.random_uniform([L3,Yfeatures]))
b_fO = tf.Variable(tf.constant(0.1,shape=[Yfeatures]))
```

In [12]:
```
#Layer 1
matmul_fc1=tf.matmul(X, W_fc1) + b_fc1
h_fc1 = tf.nn.relu(matmul_fc1)   #ReLU activation
#Layer 2
matmul_fc2=tf.matmul(h_fc1, W_fc2) + b_fc2
h_fc2 = tf.nn.relu(matmul_fc2)   #ReLU activation
#Layer 3
matmul_fc3=tf.matmul(h_fc2, W_fc3) + b_fc3
h_fc3 = tf.nn.relu(matmul_fc3)   #ReLU activation
#Output layer
matmul_fc4=tf.matmul(h_fc3, W_fO) + b_fO
output_layer = matmul_fc4  #linear activation
```

In [13]:
```
#Loss function
mean_square =  tf.reduce_mean(tf.square(Y-output_layer))
train_step = tf.train.AdamOptimizer(learning_rate).minimize(mean_square)

#Operation to save variables
saver = tf.train.Saver()
```

In [14]:
```
#Initialization and session
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    print("Training loss:",sess.run([mean_square],feed_dict={X:InputX1train,Y:
InputY1train}))
    for i in range(training_iterations):
        sess.run([train_step],feed_dict={X:InputX1train,Y:InputY1train})
        if i%display_iterations ==0:
            print("Training loss is:",sess.run([mean_square],feed_dict={X:Inpu
tX1train,Y:InputY1train}),"at itertion:",i)
            print("Validation loss is:",sess.run([mean_square],feed_dict={X:In
putX1v,Y:InputY1v}),"at itertion:",i)
    # Save the variables to disk.
    save_path = saver.save(sess, "/tmp/earthquake_model.ckpt")
    print("Model saved in file: %s" % save_path)

    print("Final training loss:",sess.run([mean_square],feed_dict={X:InputX1tr
ain,Y:InputY1train}))
    print("Final validation loss:",sess.run([mean_square],feed_dict={X:InputX1
v,Y:InputY1v}))
```

```
Training loss: [13.861377]
Training loss is: [13.684432] at itertion: 0
Validation loss is: [12.185986] at itertion: 0
Training loss is: [1.8331605] at itertion: 200
Validation loss is: [1.6050934] at itertion: 200
Training loss is: [1.1761605] at itertion: 400
Validation loss is: [1.0898186] at itertion: 400
Training loss is: [0.7910553] at itertion: 600
Validation loss is: [0.7700378] at itertion: 600
Training loss is: [0.55884093] at itertion: 800
Validation loss is: [0.5634174] at itertion: 800
Model saved in file: /tmp/earthquake_model.ckpt
Final training loss: [0.414103]
Final validation loss: [0.4259131]
```

In [15]:
```
#Testing
lat = input("Enter Latitude between -77 to 86:")
long = input("Enter Longitude between -180 to 180:")
depth = input("Enter Depth between 0 to 700:")
date = input("Enter the date (Month/Day/Year format):")
InputX2 = np.asarray([[lat,long,depth,mapdateTotime(date)]],dtype=np.float32)
InputX2_norm = (InputX2-X1_min)/(X1_max-X1_min)
InputX1test = np.resize(InputX2_norm,(1,Xfeatures))
with tf.Session() as sess:
    # Restore variables from disk for validation.
    saver.restore(sess, "/tmp/earthquake_model.ckpt")
    print("Model restored.")
    #print("Final validation loss:",sess.run([mean_square],feed_dict={X:InputX
1v,Y:InputY1v}))
    print("output:",sess.run([output_layer],feed_dict={X:InputX1test}))
```

```
Enter Latitude between -77 to 86:75
Enter Longitude between -180 to 180:181
Enter Depth between 0 to 700:600
Enter the date (Month/Day/Year format):04/02/1965
INFO:tensorflow:Restoring parameters from /tmp/earthquake_model.ckpt
Model restored.
output: [array([[8.778511]], dtype=float32)]
```

In [ ]:

In [ ]:

In [ ]: