Project of Disease Data Set

This project is about using related attributes to determine if a patient has a heart attack. As an assistant tool for doctors, it has great significance. This database contains 76 attributes, I choose 14 of them, which are

0. age
1. sex
2. chest pain type -Value 1: typical angina
   -Value 2: atypical angina
   -Value 3: non-anginal pain
   -Value 4: asymptomatic
3. resting blood pressure (in mm Hg on admission to the hospital)
4. serum cholestoral in mg / dl
5. (fasting blood sugar> 120 mg / dl) (1 = true; 0 = false)
6. resting electrocardiographic results -Value 0: normal
   -Value 1: having ST-T wave abnormality (T wave inversions and / or ST elevation or depression of> 0.05 mV)
   -Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
7. maximum heart rate achieved
8. exercise induced angina (1 = yes; 0 = no)
9. ST depression induced by exercise relative to rest
10. the slope of the peak exercise ST segment -Value 1: upsloping
    -Value 2: flat
    -Value 3: downsloping
11. number of major vessels (0-3) colored by flourosopy
12. thal -Value 3: normal
    -Value 6: fixed defect
    -Value 7: reversable defect
13. diagnosis of heart disease (angiographic disease status) -Value 0: <50% diameter narrowing
    -Value 1:> 50% diameter narrowing

In this database, there is some missing data, so we need to eliminate the missing samples. Then separate the test and training sets.

In [1]:

```
import tensorflow.compat.v1 as tf
```

In [2]:

```
f = open("processed.cleveland.data","r")
dataset = []
for line in f:
    line = f.readline()
    line = line.strip('\n').split(',')
    dataset.append(line)
dataset = dataset[0:-1]

def feature(datum):
    #feat = [1, float(datum[0]), float(datum[1]), float(datum[11]), float(datum[12])]
    feat = [1, float(datum[0]), float(datum[1]), float(datum[2]), float(datum[3]), float(da
    return feat

import random
dataset_train = random.shuffle(dataset)
X = [feature(d) for d in dataset if d[11] != '?' and  d[12] != '?']
y = [float(d[13]) for d in dataset if d[11] != '?' and  d[12] != '?']

N = len(X)
X_train = X[:N//2]
X_test = X[N//2:]
y_train = y[:N//2]
y_test = y[N//2:]
```

After data cleaning, we use tensorflow to iterate and use MSE to determine the optimal solution.

In [3]:

```
tf.disable_v2_behavior()
y_train = tf.constant(y_train ,shape=[len(y_train),1])
K = len(X[0])
def MSE(X, y,theta):
    return tf.reduce_mean((tf.matmul(X_train,theta) - y_train)**2)
theta = tf.Variable(tf.constant([0.0]*K,shape=[K,1]))
optimizer = tf.train.AdamOptimizer(0.01)
objective = MSE(X_train,y_train,theta)
train = optimizer.minimize(objective)
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for iteration in range(10000):
    cvalues = sess.run([train,objective])
    print("objective = =" + str(cvalues[1]))

with sess.as_default():
    print(MSE(X,y,theta).eval())
    print(theta.eval())
```

```
WARNING:tensorflow:From D:\anaconda3\envs\newenvt\lib\site-packages\tensor
flow_core\python\compat\v2_compat.py:88: disable_resource_variables (from
tensorflow.python.ops.variable_scope) is deprecated and will be removed in
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
objective = =2.5
objective = =28.277552
objective = =3.675342
objective = =7.3388686
objective = =16.199606
objective = =10.870259
objective = =2.97788
objective = =2.3994956
objective = =7.202259
objective = =9.166967
objective = =6.128783
objective = =2.3240914
objective = =1.7455987
```

so we see after convergence mean squared error is 0.67, and we have fourteen values of theta. Thank you for your evaluation, let's learn and progress together.