

```

In [1]: import csv
import os
import numpy as np
import random

data = []
with open("data.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        temp = np.array(row).astype(np.float).tolist()
        data.append(temp)

maxi = [-999999 for i in range(len(data[0]))]
mini = [999999 for i in range(len(data[0]))]

for i in range(len(data)):
    for j in range(len(data[i])):
        if data[i][j]>maxi[j]:
            maxi[j] = data[i][j]
        if data[i][j]<mini[j]:
            mini[j] = data[i][j]

for i in range(len(data)):
    for j in range(len(data[i])):
        data[i][j] = (data[i][j]-mini[j])/(maxi[j]-mini[j])

random.shuffle(data)

In [2]: import tensorflow.compat.v1 as tf
tf.disable_eager_execution()

def initialize():
    #placeholder
    x = tf.placeholder(tf.float32, [None, D_in])
    y = tf.placeholder(tf.float32, [None, D_out])
    w = {
        'h1': tf.Variable(tf.random_normal([D_in, D_h1])),
        'h2': tf.Variable(tf.random_normal([D_h1, D_h2])),
        'h3': tf.Variable(tf.random_normal([D_h2, D_h3])),
        'h4': tf.Variable(tf.random_normal([D_h3, D_h4])),
        #'h5': tf.Variable(tf.random_normal([D_h4, D_h5])),
        #'h6': tf.Variable(tf.random_normal([D_h5, D_h6])),
        'out': tf.Variable(tf.random_normal([D_h4, D_out]))
    }
    b = {
        'h1': tf.Variable(tf.random_normal([D_h1])),
        'h2': tf.Variable(tf.random_normal([D_h2])),
        'h3': tf.Variable(tf.random_normal([D_h3])),
        'h4': tf.Variable(tf.random_normal([D_h4])),
        #'h5': tf.Variable(tf.random_normal([D_h5])),
        #'h6': tf.Variable(tf.random_normal([D_h6])),
        'out': tf.Variable(tf.random_normal([D_out]))
    }

    #activation functions #tf.layers.batch_normalization()
    def multilayer_perceptron(x):
        h1_layer = tf.sigmoid(tf.add(tf.matmul(x, w['h1']), b['h1']))
        h2_layer = tf.sigmoid(tf.add(tf.matmul(h1_layer, w['h2']), b['h2']))
        h3_layer = tf.sigmoid(tf.add(tf.matmul(h2_layer, w['h3']), b['h3']))
        h4_layer = tf.sigmoid(tf.add(tf.matmul(h3_layer, w['h4']), b['h4']))
        #'h5_layer = tf.sigmoid(tf.add(tf.matmul(h4_layer, w['h5']), b['h5']))
        #'h6_layer = tf.sigmoid(tf.add(tf.matmul(h5_layer, w['h6']), b['h6']))
        out_layer = tf.sigmoid(tf.add(tf.matmul(h4_layer, w['out']), b['out']))
        return out_layer
    pred = multilayer_perceptron(x)

    #loss function
    #cost = tf.reduce_sum(tf.where(tf.greater(pred, 0.932), 10*(y-pred)*(y-pred),
    #                               tf.where(tf.greater(0.042, pred), 10*(y-pred)*(y-pred),
    #                                       (y-pred)*(y-pred))))
    cost = tf.reduce_sum((y-pred)*(y-pred))

    #optimizer and others
    optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
    init = tf.global_variables_initializer()
    variables_dict = {
        'b1': b['h1'],
        'b2': b['h2'],
        'b3': b['h3'],
        'b4': b['h4'],
        #'b5': b['h5'],
        #'b6': b['h6'],
        'b'out': b['out'],
        'w1': w['h1'],
        'w2': w['h2'],
        'w3': w['h3'],
        'w4': w['h4'],
        #'w5': w['h5'],
        #'w6': w['h6'],
        'w'out': w['out']
    }
    saver = tf.train.Saver(variables_dict)
    """
    tf.summary.scalar('loss', cost)
    tf.summary.histogram('b1', b['h1'])
    tf.summary.histogram('b2', b['h2'])
    tf.summary.histogram('b3', b['h3'])
    tf.summary.histogram('b4', b['h4'])
    tf.summary.histogram('b'out', b['out'])
    tf.summary.histogram('w1', w['h1'])
    tf.summary.histogram('w2', w['h2'])
    tf.summary.histogram('w3', w['h3'])
    tf.summary.histogram('w4', w['h4'])
    tf.summary.histogram('w'out', w['out'])
    """
    saver = tf.train.Saver(max_to_keep=1000, keep_checkpoint_every_n_hours=1)
    return x, y, pred, cost, optimizer, init, saver

def evaluate(filename, dimensionx):
    xdata = []
    with open(filename) as f:
        lines = f.readlines()
    temp = []
    for line in lines:
        if len(temp)==dimensionx:
            xdata.append(temp)
            temp = []
    with tf.Session() as sess:
        saver.restore(sess, "./model.ckpt")
        yhat = pred.eval(feed_dict = {x: xdata})

    def evaluatewhentrain(xdata, modelfile):
        with tf.Session() as sess:
            saver.restore(sess, modelfile)
            yhat = pred.eval(feed_dict = {x: xdata})
        return yhat

    def train(X_test, y_test):
        with tf.Session() as sess:
            sess.run(init)
            merged_all = tf.summary.merge_all()
            writer = tf.summary.FileWriter('logs/', sess.graph)
            for epoch in range(training_epochs):
                avg_cost = 0
                total_batch = int(num/batch_size)
                if num%batch_size != 0:
                    total_batch = total_batch + 1
                for i in range(total_batch):
                    batch_xs = xdata[i*batch_size:(i+1)*batch_size]
                    batch_ys = ydata[i*batch_size:(i+1)*batch_size]
                    _, c = sess.run([optimizer, cost], feed_dict={x: batch_xs, y: batch_ys})
                    avg_cost += c / total_batch
                if (epoch+1) % display_step == 0:
                    print("Epoch: %d %d%% (%d) % Step %d % Cost: %.9f" % (epoch+1, epoch*100/len(xdata), epoch, epoch+1, cost))
                    merged = sess.run(merged_all, feed_dict={x: batch_xs, y: batch_ys})
                    writer.add_summary(merged, epoch)
                if (epoch+1) % save_step == 0:
                    saver.save(sess, "./model"+str(epoch)+".ckpt")
                    print("Save model done.")
                    yhat = np.array(evaluatewhentrain(X_test, "./model"+str(epoch)+".ckpt"))
                    accuracy = 100 - np.around(np.median(np.abs(yhat-y_test))/yhat*100, axis=0, decimals=1)
                    print('5-layer ANN', accuracy, '% Step '+str(epoch))

    saver = tf.train.Saver(max_to_keep=1000, keep_checkpoint_every_n_hours=1)
    return x, y, pred, cost, optimizer, init, saver

In [3]: #training settings
training_epochs = 1000
save_step = 100
batch_size = 100
display_step = 10

#network structure
D_in, D_out = 8, 1
D_h1 = 20
D_h2 = 20
D_h3 = 20
D_h4 = 20
#D_h5 = 20
#D_h6 = 20

#initialize
x, y, pred, cost, optimizer, init, saver = initialize()

#load data
xdata = np.array(data)[500:,:8]
ydata = np.mat(np.array(data)[500,:8]).T
X_test = np.array(data)[500,:8]
y_test = np.mat(np.array(data)[500,:8]).T

num = len(xdata)
#print(X_test)
#print(y_test)

In [4]: train(X_test, y_test)
Epoch: 0001 cost: 7.634255886
Epoch: 0010 cost: 1.160608697
Epoch: 0020 cost: 1.084276414
Epoch: 0030 cost: 1.033107996
Epoch: 0040 cost: 0.986758208
Epoch: 0050 cost: 0.942865157
Epoch: 0060 cost: 0.902528405
Epoch: 0070 cost: 0.868660140
Epoch: 0080 cost: 0.844467199
Epoch: 0090 cost: 0.826716185
Epoch: 0100 cost: 0.810775602
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[77.91]] % Step 99
Epoch: 0110 cost: 0.794251847
Epoch: 0120 cost: 0.776368058
Epoch: 0130 cost: 0.757749736
Epoch: 0140 cost: 0.740274036
Epoch: 0150 cost: 0.725482833
Epoch: 0160 cost: 0.713436246
Epoch: 0170 cost: 0.703327596
Epoch: 0180 cost: 0.694372804
Epoch: 0190 cost: 0.686074901
Epoch: 0200 cost: 0.678069425
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[75.21]] % Step 199
Epoch: 0210 cost: 0.670149934
Epoch: 0220 cost: 0.662253439
Epoch: 0230 cost: 0.654404473
Epoch: 0240 cost: 0.646673894
Epoch: 0250 cost: 0.639148051
Epoch: 0260 cost: 0.631910449
Epoch: 0270 cost: 0.625022859
Epoch: 0280 cost: 0.618547177
Epoch: 0290 cost: 0.612489545
Epoch: 0300 cost: 0.606857407
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[73.31]] % Step 499
Epoch: 0310 cost: 0.6013636124
Epoch: 0320 cost: 0.596800685
Epoch: 0330 cost: 0.592323381
Epoch: 0340 cost: 0.588180643
Epoch: 0350 cost: 0.584358054
Epoch: 0360 cost: 0.580849177
Epoch: 0370 cost: 0.577648592
Epoch: 0380 cost: 0.574746019
Epoch: 0390 cost: 0.572122055
Epoch: 0400 cost: 0.569751126
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[73.31]] % Step 499
Epoch: 0410 cost: 0.567603499
Epoch: 0420 cost: 0.565640980
Epoch: 0430 cost: 0.563863558
Epoch: 0440 cost: 0.562229168
Epoch: 0450 cost: 0.560690956
Epoch: 0460 cost: 0.559285700
Epoch: 0470 cost: 0.557063848
Epoch: 0480 cost: 0.556722593
Epoch: 0490 cost: 0.555552632
Epoch: 0500 cost: 0.554444952
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[73.31]] % Step 499
Epoch: 0510 cost: 0.553395730
Epoch: 0520 cost: 0.552396345
Epoch: 0530 cost: 0.551442826
Epoch: 0540 cost: 0.550530952
Epoch: 0550 cost: 0.549656731
Epoch: 0560 cost: 0.548016967
Epoch: 0570 cost: 0.548009394
Epoch: 0580 cost: 0.547228312
Epoch: 0590 cost: 0.546474326
Epoch: 0600 cost: 0.545743996
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[73.31]] % Step 499
Epoch: 0610 cost: 0.545935243
Epoch: 0620 cost: 0.545346652
Epoch: 0630 cost: 0.543674695
Epoch: 0640 cost: 0.542272278
Epoch: 0650 cost: 0.541752231
Epoch: 0660 cost: 0.541437368
Epoch: 0670 cost: 0.540532329
Epoch: 0680 cost: 0.539938879
Epoch: 0690 cost: 0.539353412
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[73.31]] % Step 499
Epoch: 0700 cost: 0.538775432
Epoch: 0710 cost: 0.538204642
Epoch: 0720 cost: 0.537639785
Epoch: 0730 cost: 0.537079936
Epoch: 0740 cost: 0.536524666
Epoch: 0750 cost: 0.535972893
Epoch: 0760 cost: 0.535424137
Epoch: 0770 cost: 0.534877580
Epoch: 0780 cost: 0.534332538
Epoch: 0790 cost: 0.53378306
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 0800 cost: 0.53378306
Epoch: 0810 cost: 0.530509198
Epoch: 0820 cost: 0.529954863
Epoch: 0830 cost: 0.529396987
Epoch: 0840 cost: 0.528834993
Epoch: 0850 cost: 0.528268188
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 0860 cost: 0.527696365
Epoch: 0870 cost: 0.527118677
Epoch: 0880 cost: 0.526534790
Epoch: 0890 cost: 0.525944167
Epoch: 0900 cost: 0.524126214
Epoch: 0910 cost: 0.522769365
Epoch: 0920 cost: 0.522118677
Epoch: 0930 cost: 0.526534790
Epoch: 0940 cost: 0.525944167
Epoch: 0950 cost: 0.525346196
Epoch: 0960 cost: 0.524740416
Epoch: 0970 cost: 0.524126214
Epoch: 0980 cost: 0.523502970
Epoch: 0990 cost: 0.522870326
Epoch: 1000 cost: 0.522227478
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 1010 cost: 0.522227478
Epoch: 1020 cost: 0.521648877
Epoch: 1030 cost: 0.521069206
Epoch: 1040 cost: 0.520489535
Epoch: 1050 cost: 0.520009864
Epoch: 1060 cost: 0.519420193
Epoch: 1070 cost: 0.518830522
Epoch: 1080 cost: 0.518240851
Epoch: 1090 cost: 0.517651180
Epoch: 1100 cost: 0.517061509
Epoch: 1110 cost: 0.516471838
Epoch: 1120 cost: 0.515882167
Epoch: 1130 cost: 0.515292496
Epoch: 1140 cost: 0.514702825
Epoch: 1150 cost: 0.514113154
Epoch: 1160 cost: 0.513523483
Epoch: 1170 cost: 0.512933812
Epoch: 1180 cost: 0.512344141
Epoch: 1190 cost: 0.511754470
Epoch: 1200 cost: 0.511164800
Epoch: 1210 cost: 0.510575129
Epoch: 1220 cost: 0.509985458
Epoch: 1230 cost: 0.509395787
Epoch: 1240 cost: 0.508806116
Epoch: 1250 cost: 0.508216445
Epoch: 1260 cost: 0.507626774
Epoch: 1270 cost: 0.507037103
Epoch: 1280 cost: 0.506447432
Epoch: 1290 cost: 0.505857761
Epoch: 1300 cost: 0.505268090
Epoch: 1310 cost: 0.504678419
Epoch: 1320 cost: 0.504088748
Epoch: 1330 cost: 0.503499077
Epoch: 1340 cost: 0.502909406
Epoch: 1350 cost: 0.502319735
Epoch: 1360 cost: 0.501730064
Epoch: 1370 cost: 0.501140393
Epoch: 1380 cost: 0.500550722
Epoch: 1390 cost: 0.500000000
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 1400 cost: 0.499410330
Epoch: 1410 cost: 0.498820659
Epoch: 1420 cost: 0.498230988
Epoch: 1430 cost: 0.497641317
Epoch: 1440 cost: 0.497051646
Epoch: 1450 cost: 0.496461975
Epoch: 1460 cost: 0.495872304
Epoch: 1470 cost: 0.495282633
Epoch: 1480 cost: 0.494692962
Epoch: 1490 cost: 0.494103291
Epoch: 1500 cost: 0.493513620
Epoch: 1510 cost: 0.492923949
Epoch: 1520 cost: 0.492334278
Epoch: 1530 cost: 0.491744607
Epoch: 1540 cost: 0.491154936
Epoch: 1550 cost: 0.490565265
Epoch: 1560 cost: 0.489975594
Epoch: 1570 cost: 0.489385923
Epoch: 1580 cost: 0.488796252
Epoch: 1590 cost: 0.488206581
Epoch: 1600 cost: 0.487616910
Epoch: 1610 cost: 0.487027239
Epoch: 1620 cost: 0.486437568
Epoch: 1630 cost: 0.485847897
Epoch: 1640 cost: 0.485258226
Epoch: 1650 cost: 0.484668555
Epoch: 1660 cost: 0.484078884
Epoch: 1670 cost: 0.483489213
Epoch: 1680 cost: 0.482899542
Epoch: 1690 cost: 0.482309871
Epoch: 1700 cost: 0.481720200
Epoch: 1710 cost: 0.481130529
Epoch: 1720 cost: 0.480540858
Epoch: 1730 cost: 0.480000000
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 1740 cost: 0.479411330
Epoch: 1750 cost: 0.478821659
Epoch: 1760 cost: 0.478231988
Epoch: 1770 cost: 0.477642317
Epoch: 1780 cost: 0.477052646
Epoch: 1790 cost: 0.476462975
Epoch: 1800 cost: 0.475873304
Epoch: 1810 cost: 0.475283633
Epoch: 1820 cost: 0.474693962
Epoch: 1830 cost: 0.474104291
Epoch: 1840 cost: 0.473514620
Epoch: 1850 cost: 0.472924949
Epoch: 1860 cost: 0.472335278
Epoch: 1870 cost: 0.471745607
Epoch: 1880 cost: 0.471155936
Epoch: 1890 cost: 0.470566265
Epoch: 1900 cost: 0.470000000
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 1910 cost: 0.469411330
Epoch: 1920 cost: 0.468821659
Epoch: 1930 cost: 0.468231988
Epoch: 1940 cost: 0.467642317
Epoch: 1950 cost: 0.467052646
Epoch: 1960 cost: 0.466462975
Epoch: 1970 cost: 0.465873304
Epoch: 1980 cost: 0.465283633
Epoch: 1990 cost: 0.464693962
Epoch: 2000 cost: 0.464104291
Epoch: 2010 cost: 0.463514620
Epoch: 2020 cost: 0.462924949
Epoch: 2030 cost: 0.462335278
Epoch: 2040 cost: 0.461745607
Epoch: 2050 cost: 0.461155936
Epoch: 2060 cost: 0.460566265
Epoch: 2070 cost: 0.460000000
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 2080 cost: 0.459411330
Epoch: 2090 cost: 0.458821659
Epoch: 2100 cost: 0.458231988
Epoch: 2110 cost: 0.457642317
Epoch: 2120 cost: 0.457052646
Epoch: 2130 cost: 0.456462975
Epoch: 2140 cost: 0.455873304
Epoch: 2150 cost: 0.455283633
Epoch: 2160 cost: 0.454693962
Epoch: 2170 cost: 0.454104291
Epoch: 2180 cost: 0.453514620
Epoch: 2190 cost: 0.452924949
Epoch: 2200 cost: 0.452335278
Epoch: 2210 cost: 0.451745607
Epoch: 2220 cost: 0.451155936
Epoch: 2230 cost: 0.450566265
Epoch: 2240 cost: 0.450000000
save model done.
INFO:tensorflow:Restoring parameters from ./model199.ckpt
5-layer ANN [[72.41]] % Step 899
Epoch: 2250 cost: 0.449411330
Epoch: 2260 cost: 0.448821659
Epoch: 2270 cost: 0.448231988
Epoch: 2280 cost: 0.447642317
Epoch: 2290 cost: 0.447052646
Epoch: 2300 cost: 0.446462975
Epoch: 2310 cost: 0.445873304
Epoch: 2320 cost: 0.445283633
Epoch: 2330 cost: 0.444693962
Epoch: 2340 cost: 0.444104291
Epoch: 2350 cost: 0.443514620
Epoch: 2360 cost: 0.442924949
Epoch: 2370 cost: 0.442335278
Epoch: 2380 cost: 0.4417
```