```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
# GradientDescentOptimizer 경사하강법
```

Sect3. Cost 최소화기법, How to minimize cost

Ex01. Multi-variable linear regression

```
from tqdm import tqdm_notebook
import tensorflow as tf

tf.set_random_seed(777)

x1_data = [73., 93., 89., 96., 73.]
x2_data = [80., 88., 91., 98., 66.]
x3_data = [75., 93., 90., 100., 70.]

y_data = [152., 185., 180., 196., 142.]

x1 = tf.placeholder(tf.float32)
x2 = tf.placeholder(tf.float32)
x3 = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

w1 = tf.Variable(tf.random_normal([1]), name='weight1')
w2 = tf.Variable(tf.random_normal([1]), name='weight2')
w3 = tf.Variable(tf.random_normal([1]), name='weight3')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = x1*w1 + x2*w2 + x3*w3 + b
print(hypothesis)

cost = tf.reduce_mean(tf.square(hypothesis - Y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 1e-5)
train = optimizer.minimize(cost)
```

```
Tensor("add_5:0", dtype=float32)
```

```
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in tqdm_notebook(range(2001)):
    cost_val, hy_val, _ = sess.run([cost, hypothesis, train],
                                   feed_dict={x1: x1_data, x2: x2_data, x3: x3_data, Y: y_data})

    if step % 100 == 0 or step < 10 :
        print("₩nStep : {} ₩nCost : {} ₩nPrediction :₩n{}".format(step, cost_val, hy_val))
```

```
HBox(children=(IntProgress(value=0, max=2001), HTML(value='')))

Step : 0
Cost : 27199.822265625
Prediction :
[-4.746768  15.750464   4.27555    5.7535353 16.655096 ]

Step : 1
Cost : 8558.369140625
Prediction :
[59.607788 93.09478  80.48686  88.745415 75.64831 ]

Step : 2
Cost : 2715.2490234375
Prediction :
[ 95.63871 136.3962  123.15513 135.20978 108.67539]

Step : 3
Cost : 883.7234497070312
Prediction :
[115.81229 160.63828 147.04388 161.22372 127.16497]

Step : 4
Cost : 309.62017822265625
Prediction :
[127.1079  174.2097  160.41869 175.78822 137.51556]

Step : 5
Cost : 129.65162658691406
Prediction :
[133.43309 181.80704 167.9071  183.94263 143.30942]

Step : 6
Cost : 73.22370910644531
Prediction :
[136.9755  186.0597  172.09995 188.5082  146.55212]

Step : 7
Cost : 55.51879119873047
Prediction :
[138.95995 188.43977 174.44772 191.06458 148.36653]

Step : 8
Cost : 49.951637268066406
Prediction :
[140.07216 189.77151 175.76254 192.49606 149.38132]

Step : 9
Cost : 48.18914031982422
Prediction :
[140.69601 190.5163  176.49901 193.29774 149.9484 ]

Step : 100
Cost : 45.12461471557617
Prediction :
[141.72708 191.30096 177.50871 194.36923 150.45708]
```

```
Step : 200
Cost : 42.75444412231445
Prediction :
[141.98502 191.1242  177.58788 194.42535 150.22609]

Step : 300
Cost : 40.509361267089844
Prediction :
[142.23611 190.95224 177.66498 194.48001 150.00134]

Step : 400
Cost : 38.38257598876953
Prediction :
[142.48048 190.7848  177.74    194.53316 149.78258]

Step : 500
Cost : 36.36799240112305
Prediction :
[142.71834 190.62184 177.81303 194.58485 149.56969]

Step : 600
Cost : 34.45978546142578
Prediction :
[142.94984 190.46327 177.88411 194.63515 149.3625 ]

Step : 700
Cost : 32.65210723876953
Prediction :
[143.1752  190.30888 177.9533  194.68407 149.1609 ]

Step : 800
Cost : 30.939807891845703
Prediction :
[143.39453 190.15863 178.02063 194.73167 148.96469]

Step : 900
Cost : 29.317895889282227
Prediction :
[143.60799 190.0124  178.0862  194.77797 148.77373]

Step : 1000
Cost : 27.781513214111328
Prediction :
[143.81577 189.87009 178.15    194.82301 148.58789]

Step : 1100
Cost : 26.326147079467773
Prediction :
[144.01802 189.73157 178.2121  194.86685 148.40706]

Step : 1200
Cost : 24.947603225708008
Prediction :
[144.21484 189.59674 178.27255 194.90945 148.23105]

Step : 1300
Cost : 23.64158058166504
Prediction :
```

```
Prediction :
[144.40646 189.46548 178.3314  194.95093 148.05977]

Step : 1400
Cost : 22.40459632873535
Prediction :
[144.59294 189.33775 178.38869 194.99123 147.89308]

Step : 1500
Cost : 21.232831954956055
Prediction :
[144.77444 189.21342 178.44444 195.03047 147.73087]

Step : 1600
Cost : 20.122867584228516
Prediction :
[144.95113 189.09242 178.49872 195.06862 147.57301]

Step : 1700
Cost : 19.071537017822266
Prediction :
[145.12308 188.97467 178.55156 195.10574 147.4194 ]

Step : 1800
Cost : 18.075523376464844
Prediction :
[145.29047 188.86003 178.60298 195.14183 147.2699 ]

Step : 1900
Cost : 17.132110595703125
Prediction :
[145.45338 188.74846 178.65303 195.17694 147.12442]

Step : 2000
Cost : 16.23847007751465
Prediction :
[145.61194 188.63986 178.70177 195.21109 146.98283]
```

Ex02. Multi-variable matmul linear regression

```
import tensorflow as tf

tf.set_random_seed(777)

x_data = [[73., 80., 75.],
          [93., 88., 93.],
          [89., 91., 90.],
          [96., 98., 100.],
          [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]

X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape = [None, 1])

W = tf.Variable(tf.random_normal([3,1]), name='weight')
```

```
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.matmul(X, W) + b

cost = tf.reduce_mean(tf.square(hypothesis - Y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate =1e-5)
train = optimizer.minimize(cost)
```

```
sess= tf.Session()

sess.run(tf.global_variables_initializer())

for step in tqdm_notebook(range(2001)) :
    cost_val, hy_val, _ = sess.run(
    [cost, hypothesis, train], feed_dict = {X: x_data, Y : y_data})

    if step % 100 == 0 or step < 10 :
        print("₩nStep : {} ₩nCost : {} ₩n Prediction : ₩n{}".format(
        step,cost_val,hy_val))
```

```
HBox(children=(IntProgress(value=0, max=2001), HTML(value='')))

Step : 0
Cost : 14491.9814453125
 Prediction :
[[43.877033]
 [56.758087]
 [53.818756]
 [58.703094]
 [44.26979 ]]

Step : 1
Cost : 4542.75244140625
 Prediction :
[[ 90.890015]
 [113.26388 ]
 [109.49504 ]
 [119.33313 ]
 [ 87.36943 ]]

Step : 2
Cost : 1424.196533203125
 Prediction :
[[117.21093]
 [144.89935]
 [140.66617]
 [153.27768]
 [111.49928]]

Step : 3
Cost : 446.69488525390625
 Prediction :
[[131.94711]
 [162.61084]
 [158.11777]
 [172.282  ]
 [125.00865]]

Step : 4
Cost : 140.30014038085938
 Prediction :
[[140.19743]
 [172.52681]
 [167.88829]
 [182.92183]
 [132.57198]]

Step : 5
Cost : 44.2617073059082
 Prediction :
[[144.81656]
 [178.07834]
 [173.35847]
 [188.87866]
 [136.80635]]

Step : 6
```

```
Cost : 14.158638000488281
 Prediction :
[[147.4027 ]
 [181.1864 ]
 [176.42105]
 [192.2137 ]
 [139.17697]]

Step : 7
Cost : 4.7228569984436035
 Prediction :
[[148.85066]
 [182.92644]
 [178.1357 ]
 [194.08087]
 [140.50414]]

Step : 8
Cost : 1.7651798725128174
 Prediction :
[[149.66139]
 [183.90057]
 [179.09569]
 [195.12625]
 [141.2471 ]]

Step : 9
Cost : 0.8380519151687622
 Prediction :
[[150.11536]
 [184.44589]
 [179.63318]
 [195.71152]
 [141.66301]]

Step : 100
Cost : 0.40681833028793335
 Prediction :
[[150.7071 ]
 [185.12993]
 [180.32143]
 [196.45796]
 [142.18054]]

Step : 200
Cost : 0.3984677195549011
 Prediction :
[[150.72263]
 [185.11943]
 [180.32635]
 [196.46005]
 [142.16803]]

Step : 300
Cost : 0.3905394673347473
 Prediction :
[[150.73773]
 [185.1008 ]
 [185.1092 ]
 [180.33118]
 [196.46204]
 [142.15587]]

Step : 400
Cost : 0.3830018937587738
 Prediction :
[[150.75247]
 [185.09927]
 [180.33588]
 [196.46396]
 [142.14407]]

Step : 500
Cost : 0.37583914399147034
 Prediction :
[[150.76683]
 [185.08957]
 [180.34047]
 [196.46579]
 [142.13261]]

Step : 600
Cost : 0.3690322935581207
 Prediction :
[[150.78082]
 [185.08012]
 [180.34496]
 [196.46754]
 [142.12148]]

Step : 700
Cost : 0.3625571131706238
 Prediction :
[[150.79448]
 [185.07095]
 [180.34932]
 [196.46925]
 [142.11067]]

Step : 800
Cost : 0.35641202330589294
 Prediction :
[[150.80774]
 [185.06198]
 [180.35358]
 [196.47084]
 [142.10013]]

Step : 900
Cost : 0.35056233406066895
 Prediction :
[[150.8207 ]
 [185.05325]
 [180.35773]
 [196.4724 ]
 [142.08992]]
```

```
Step : 1000
Cost : 0.34499743580818176
 Prediction :
[[150.83331]
 [185.04475]
 [180.36177]
 [196.47386]
 [142.08002]]

Step : 1100
Cost : 0.33970776200294495
 Prediction :
[[150.84561]
 [185.03647]
 [180.36574]
 [196.47527]
 [142.07039]]

Step : 1200
Cost : 0.33467310667037964
 Prediction :
[[150.85759]
 [185.0284 ]
 [180.36958]
 [196.4766 ]
 [142.06104]]

Step : 1300
Cost : 0.3298836350440979
 Prediction :
[[150.86926]
 [185.02054]
 [180.37334]
 [196.47786]
 [142.05196]]

Step : 1400
Cost : 0.3253173232078552
 Prediction :
[[150.88066]
 [185.0129 ]
 [180.37703]
 [196.47905]
 [142.04314]]

Step : 1500
Cost : 0.32098454236984253
 Prediction :
[[150.89174]
 [185.00542]
 [180.3806 ]
 [196.48021]
 [142.03455]]

Step : 1600
Cost : 0.31685104966163635
```

```
 Prediction :
[[150.90256]
 [184.99815]
 [180.38411]
 [196.4813 ]
 [142.02625]]

Step : 1700
Cost : 0.3129104971885681
 Prediction :
[[150.9131 ]
 [184.99107]
 [180.38751]
 [196.48232]
 [142.01817]]

Step : 1800
Cost : 0.30915629863739014
 Prediction :
[[150.92339]
 [184.98418]
 [180.39084]
 [196.48329]
 [142.01035]]

Step : 1900
Cost : 0.30559012293815613
 Prediction :
[[150.9334 ]
 [184.97745]
 [180.3941 ]
 [196.48422]
 [142.00272]]

Step : 2000
Cost : 0.30218052864074707
 Prediction :
[[150.94318]
 [184.97092]
 [180.39728]
```

Ex03. File input linear regression

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # for reproducibility

xy = np.loadtxt('./data/data-01-test-score.csv', delimiter = ',', dtype=np.float32)
```

```
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

print("x_data.shape : {}, ₩tlen(x_data) : {} ₩nx_data : ₩n{}".format(x_data.shape, len(x_data), x_d
print("-"*25)
print("y_data.shape : {}  ₩ny_data : ₩n{}".format(y_data.shape, y_data))

X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.matmul(X,W) + b

cost = tf.reduce_mean(tf.square(hypothesis - Y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 1e-5)
train = optimizer.minimize(cost)
```

```
x_data.shape : (25, 3),          len(x_data) : 25
x_data :
[[ 73.  80.  75.]
 [ 93.  88.  93.]
 [ 89.  91.  90.]
 [ 96.  98. 100.]
 [ 73.  66.  70.]
 [ 53.  46.  55.]
 [ 69.  74.  77.]
 [ 47.  56.  60.]
 [ 87.  79.  90.]
 [ 79.  70.  88.]
 [ 69.  70.  73.]
 [ 70.  65.  74.]
 [ 93.  95.  91.]
 [ 79.  80.  73.]
 [ 70.  73.  78.]
 [ 93.  89.  96.]
 [ 78.  75.  68.]
 [ 81.  90.  93.]
 [ 88.  92.  86.]
 [ 78.  83.  77.]
 [ 82.  86.  90.]
 [ 86.  82.  89.]
 [ 78.  83.  85.]
 [ 76.  83.  71.]
 [ 96.  93.  95.]]
-------------------------
y_data.shape : (25, 1)
y_data :
[[152.]
 [185.]
 [180.]
 [196.]
 [142.]
 [101.]
 [149.]
 [115.]
 [175.]
 [164.]
 [141.]
 [141.]
 [184.]
 [152.]
 [148.]
 [192.]
 [147.]
 [183.]
 [177.]
 [159.]
 [177.]
 [175.]
 [175.]
 [149.]
 [192.]]
```

```
sess = tf.Session()

sess.run(tf.global_variables_initializer())

for step in range(2001) :
    cost_val, hy_val, _ = sess.run(
    [cost, hypothesis, train], feed_dict = {X : x_data, Y : y_data})

    if step % 100 == 0 or step < 10 :
        print("₩nStep : {} ₩nCost : {} ₩nPrediction :₩n{}".format(step, cost_val, hy_val))
```

```
Step : 0
Cost : 30871.0390625
Prediction :
[[-20.513838  ]
 [ -6.5941644 ]
 [-15.889416  ]
 [-15.902521  ]
 [ -1.764229  ]
 [  2.8374803 ]
 [-14.993187  ]
 [-15.9369335 ]
 [ -0.33107403]
 [  4.3142447 ]
 [-10.500959  ]
 [ -1.8907492 ]
 [-17.482384  ]
 [-15.324179  ]
 [-12.220448  ]
 [ -7.0446    ]
 [-10.685949  ]
 [-21.23749   ]
 [-19.671099  ]
 [-19.334084  ]
 [-15.431911  ]
 [ -6.0225434 ]
 [-16.643566  ]
 [-23.306065  ]
 [-10.287249  ]]

Step : 1
Cost : 11448.2578125
Prediction :
[[44.18243 ]
 [71.159454]
 [60.72527 ]
 [67.53627 ]
 [57.53459 ]
 [46.55481 ]
 [47.46454 ]
 [30.366795]
 [72.33347 ]
 [71.61786 ]
 [49.672665]
 [57.439198]
 [61.67152 ]
 [50.47614 ]
 [50.523514]
 [71.85713 ]
 [51.979065]
 [53.718845]
 [55.791725]
 [48.18766 ]
 [57.80907 ]
 [66.9207  ]
 [53.18651 ]
 [41.924053]
 [70.29643 ]]

Step : 2
Cost : 4267.099609375
Prediction :
[[ 83.52226 ]
 [118.43691 ]
 [107.31133 ]
 [118.27162 ]
 [ 93.59035 ]
 [ 73.136   ]
 [ 85.44264 ]
 [ 58.52291 ]
 [116.515884]
 [112.54017 ]
 [ 86.261406]
 [ 93.51395 ]
 [109.80167 ]
 [ 90.48658 ]
 [ 88.67535 ]
 [119.832756]
 [ 90.082565]
 [ 99.29728 ]
 [101.677826]
 [ 89.2453  ]
 [102.343796]
 [111.27326 ]
 [ 95.64743 ]
 [ 81.588745]
 [119.295074]]

Step : 3
Cost : 1612.0084228515625
Prediction :
[[107.44403 ]
 [147.18326 ]
 [135.63838 ]
 [149.12157 ]
 [115.5131  ]
 [ 89.29754 ]
 [108.53587 ]
 [ 75.64431 ]
 [143.37967 ]
 [137.42125 ]
 [108.509315]
 [115.448326]
 [139.06773 ]
 [114.81543 ]
 [111.87386 ]
 [149.00368 ]
 [113.25138 ]
 [127.01242 ]
 [129.57973 ]
 [114.21137 ]
 [129.4236  ]
 [138.24118 ]
 [121.46648 ]
 [105.70829 ]
 [105.70832 ]
 [149.08838 ]]

Step : 4
Cost : 630.3294677734375
Prediction :
[[121.99082 ]
 [164.66173 ]
 [152.86304 ]
 [167.88013 ]
 [128.84222 ]
 [ 99.123375]
 [122.5783  ]
 [ 86.05602 ]
 [159.71284 ]
 [152.54843 ]
 [122.037224]
 [128.78459 ]
 [156.86345 ]
 [129.60904 ]
 [125.97999 ]
 [166.74036 ]
 [127.33912 ]
 [143.86572 ]
 [146.5463  ]
 [129.39291 ]
 [145.88992 ]
 [154.6383  ]
 [137.16638 ]
 [120.37563 ]
 [167.2038  ]]

Step : 5
Cost : 267.35614013671875
Prediction :
[[130.83714]
 [175.28877]
 [163.33687]
 [179.28656]
 [136.94604]
 [105.09677]
 [131.11739]
 [ 92.38792]
 [169.64285]
 [161.74478]
 [130.26295]
 [136.89279]
 [167.68462]
 [138.60475]
 [134.55748]
 [177.52446]
 [135.9051 ]
 [154.1145 ]
 [156.86363]
 [138.625  ]
 [155.90273]
 [164.60791]
 [146.71335]
 [129.2955 ]
 [178.21852]]

Step : 6
Cost : 133.13693237304688
Prediction :
[[136.21724 ]
 [181.74973 ]
 [169.70578 ]
 [186.22244 ]
 [141.87251 ]
 [108.727684]
 [136.31012 ]
 [ 96.23905 ]
 [175.67938 ]
 [167.33478 ]
 [135.26463 ]
 [141.82202 ]
 [174.26486 ]
 [144.07501 ]
 [139.77322 ]
 [184.08098 ]
 [141.11354 ]
 [160.34732 ]
 [163.13783 ]
 [144.23944 ]
 [161.99138 ]
 [170.66914 ]
 [152.51897 ]
 [134.72058 ]
 [184.91559 ]]

Step : 7
Cost : 83.49371337890625
Prediction :
[[139.48973]
 [185.67752]
 [173.57874]
 [190.44   ]
 [144.86703]
 [110.93424]
 [139.46812]
 [ 98.58176]
 [179.34851]
 [170.73203]
 [138.30595]
 [144.8183 ]
 [178.26642]
 [147.40161]
 [142.9449 ]
 [188.06693]
 [144.28046]
 [164.13826]
 [166.95364]
 [147.6542 ]
 [165.69397]
 [174.35394]
 [156.04968]
 [138.0207 ]
 [188.9873 ]]

Step : 8
Cost : 65.12068939208984
Prediction :
[[141.48067]
 [188.06502]
 [175.934  ]
 [193.00471]
 [146.68684]
 [112.27471]
 [141.38889]
 [100.00725]
 [181.57808]
 [172.79587]
 [140.15526]
 [146.6392 ]
 [180.7    ]
 [149.42476]
 [144.87364]
 [190.48985]
 [146.20598]
 [166.44441]
 [169.27461]
 [149.73143]
 [167.94574]
 [176.59373]
 [158.1971 ]
 [140.0287 ]
 [191.46268]]

Step : 9
Cost : 58.30916976928711
Prediction :
[[142.69237 ]
 [189.51591 ]
 [177.36644 ]
 [194.56439 ]
 [147.79234 ]
 [113.08858 ]
 [142.55736 ]
 [100.875046]
 [182.93237 ]
 [174.04903 ]
 [141.27975 ]
 [147.74547 ]
 [182.18016 ]
 [150.65535 ]
 [146.04663 ]
 [191.96233 ]
 [147.3767  ]
 [167.84773 ]
 [170.68663 ]
 [150.99538 ]
 [169.31529 ]
 [177.95488 ]
 [177.95488 ]
 [159.50343 ]
 [141.251   ]
 [192.96739 ]]

Step : 100
Cost : 51.731849670410156
Prediction :
[[144.8182 ]
 [191.5841 ]
 [179.65768]
 [197.02872]
 [149.27426]
 [114.07639]
 [144.49251]
 [102.45059]
 [184.71442]
 [175.5875 ]
 [143.03088]
 [149.24539]
 [184.57022]
 [152.65337]
 [147.91452]
 [194.07657]
 [149.16595]
 [170.26343]
 [173.04509]
 [153.15274]
 [171.5257 ]
 [179.89615]
 [161.66095]
 [143.45084]
 [195.20166]]

Step : 200
Cost : 49.04936981201172
Prediction :
[[145.08403]
 [191.3858 ]
 [179.73175]
 [197.07741]
 [149.02066]
 [113.77967]
 [144.62685]
 [102.70233]
 [184.36995]
 [175.15158]
 [143.03871]
 [149.01239]
 [184.66992]
 [152.74663]
 [147.9686 ]
 [193.89285]
 [149.13269]
 [170.52463]
 [173.22496]
 [153.36397]
 [171.62027]
 [179.71225]
 [161.80406]
 [143.77637]
 [195.09225]]

Step : 300
Cost : 46.53648376464844
Prediction :
[[145.34044]
 [191.19354]
 [179.80266]
 [197.12482]
 [148.77423]
 [113.4936 ]
 [144.75856]
 [102.94928]
 [184.03783]
 [174.73341]
 [143.04697]
 [148.7883 ]
 [184.7645 ]
 [152.83368]
 [148.02289]
 [193.71573]
 [149.09628]
 [170.77997]
 [173.39688]
 [153.56659]
 [171.71344]
 [179.53499]
 [161.94377]
 [144.08786]
 [194.98535]]

Step : 400
Cost : 44.1815299987793
Prediction :
[[145.58778 ]
 [191.00716 ]
 [179.87053 ]
 [197.171   ]
 [148.53474 ]
 [113.21774 ]
 [144.88766 ]
 [103.191475]
 [183.71758 ]
 [174.33226 ]
 [143.05562 ]
 [148.57277 ]
 [184.85413 ]
 [152.91487 ]
 [148.07727 ]
 [193.54495 ]
 [149.05699 ]
 [171.02962 ]
 [173.56123 ]
 [153.76099 ]
 [171.80528 ]
 [171.80528 ]
 [179.36418 ]
 [162.08018 ]
 [144.38597 ]
 [194.88095 ]]

Step : 500
Cost : 41.97384262084961
Prediction :
[[145.8264  ]
 [190.8264  ]
 [179.93549 ]
 [197.21593 ]
 [148.30196 ]
 [112.95174 ]
 [145.0142  ]
 [103.428986]
 [183.4087  ]
 [173.94737 ]
 [143.06462 ]
 [148.36548 ]
 [184.93906 ]
 [152.99046 ]
 [148.1317  ]
 [193.38026 ]
 [149.01506 ]
 [171.27364 ]
 [173.71828 ]
 [153.94746 ]
 [171.8957  ]
 [179.19952 ]
 [162.21336 ]
 [144.67122 ]
 [194.77895 ]]

Step : 600
Cost : 39.903438568115234
Prediction :
[[146.05663]
 [190.65112]
 [179.99767]
 [197.25969]
 [148.07571]
 [112.69518]
 [145.13823]
 [103.66187]
 [183.1108 ]
 [173.57814]
 [143.07393]
 [148.16605]
 [185.01952]
 [153.06079]
 [148.18614]
 [193.22145]
 [148.97069]
 [171.51219]
 [173.8684 ]
 [154.12639]
 [154.42883]
 [171.98476]
 [179.04079]
 [162.34337]
 [144.94417]
 [194.67932]]

Step : 700
Cost : 37.961181640625
Prediction :
[[146.27878 ]
 [190.4812  ]
 [180.05722 ]
 [197.3023  ]
 [147.85583 ]
 [112.447754]
 [145.25977 ]
 [103.8902  ]
 [182.82349 ]
 [173.22386 ]
 [143.08356 ]
 [147.97421 ]
 [185.09578 ]
 [153.12613 ]
 [148.24051 ]
 [193.0683  ]
 [148.92415 ]
 [171.74539 ]
 [174.0119  ]
 [154.29808 ]
 [172.07245 ]
 [178.88779 ]
 [162.4703  ]
 [145.20544 ]
 [194.58206 ]]

Step : 800
Cost : 36.13839340209961
Prediction :
[[146.49315]
 [190.31635]
 [180.11421]
 [197.34378]
 [147.64207]
 [112.20907]
 [145.37883]
 [104.11401]
 [182.54631]
 [172.8839 ]
 [143.09343]
 [147.78966]
 [185.16795]
 [153.18672]
 [148.29475]
 [192.92056]
 [148.87556]
 [171.97333]
 [174.149  ]
 [154.46281]
 [172.15875]
 [178.74025]
 [162.59422]
 [145.45543]
 [194.48703]]

Step : 900
Cost : 34.42721939086914
Prediction :
[[146.70003]
 [190.15645]
 [180.16876]
 [197.38417]
 [147.43427]
 [111.97882]
 [145.49547]
 [104.33339]
 [182.2789 ]
 [172.55768]
 [143.10352]
 [147.61205]
 [185.23625]
 [153.24281]
 [148.3488 ]
 [192.77802]
 [148.82515]
 [172.19609]
 [174.28006]
 [154.62086]
 [172.24364]
 [178.59798]
 [162.71518]
 [145.69466]
 [194.39421]]

Step : 1000
Cost : 32.8203010559082
Prediction :
[[146.8997  ]
 [190.00137 ]
 [180.22098 ]
 [197.4235  ]
 [147.23227 ]
 [111.756676]
 [145.60976 ]
 [104.548386]
 [182.0209  ]
 [172.24463 ]
 [143.11383 ]
 [147.44116 ]
 [185.30093 ]
 [153.29468 ]
 [148.40265 ]
 [192.64053 ]
 [148.77307 ]
 [172.41382 ]
 [174.40532 ]
 [174.40532 ]
 [154.77252 ]
 [172.32716 ]
 [178.46078 ]
 [162.83325 ]
 [145.92358 ]
 [194.30359 ]]

Step : 1100
Cost : 31.310731887817383
Prediction :
[[147.09245]
 [189.85094]
 [180.271  ]
 [197.4618 ]
 [147.03587]
 [111.54234]
 [145.7217 ]
 [104.75907]
 [181.77196]
 [171.94423]
 [143.12431]
 [147.27672]
 [185.36208]
 [153.34254]
 [148.45625]
 [192.50789]
 [148.71953]
 [172.6266 ]
 [174.52502]
 [154.91809]
 [172.40932]
 [178.32849]
 [162.94853]
 [146.14268]
 [194.21509]]

Step : 1200
Cost : 29.892072677612305
Prediction :
[[147.27853]
 [189.70496]
 [180.31886]
 [197.49908]
 [146.84492]
 [111.33551]
 [145.83133]
 [104.9655 ]
 [181.53171]
 [171.6559 ]
 [143.13495]
 [147.11844]
 [185.41992]
 [153.3866 ]
 [148.50952]
 [192.37988]
 [148.66464]
 [172.83456]
 [174.63943]
 [155.0578 ]
 [172.49008]
 [178.20087]
 [163.06107]
 [146.35239]
 [194.12863]]

Step : 1300
Cost : 28.558568954467773
Prediction :
[[147.45818]
 [189.56334]
 [180.3647 ]
 [197.53542]
 [146.65926]
 [111.13591]
 [145.93867]
 [105.16774]
 [181.29985]
 [171.37917]
 [143.14574]
 [146.9661 ]
 [185.4746 ]
 [153.42705]
 [148.56248]
 [192.25636]
 [148.6086 ]
 [173.03777]
 [174.74876]
 [155.19188]
 [172.56947]
 [178.07777]
 [163.17091]
 [146.55305]
 [194.04422]]

Step : 1400
Cost : 27.304662704467773
Prediction :
[[147.63162 ]
 [189.42595 ]
 [180.40857 ]
 [197.57077 ]
 [146.47873 ]
 [110.943275]
 [146.0438  ]
 [105.36585 ]
 [181.07605 ]
 [171.11354 ]
 [143.15662 ]
 [146.81946 ]
 [185.52628 ]
 [153.46413 ]
 [148.61505 ]
 [192.13715 ]
 [148.5515  ]
 [173.23634 ]
 [174.85324 ]
 [155.32056 ]
 [172.6475  ]
 [177.95901 ]
 [163.27812 ]
 [146.74509 ]
 [193.96179 ]]

Step : 1500
Cost : 26.125165939331055
Prediction :
[[147.79912 ]
 [189.29262 ]
 [180.45058 ]
 [197.60521 ]
 [146.30318 ]
 [110.757324]
 [146.14674 ]
 [105.559906]
 [180.86003 ]
 [170.8586  ]
 [143.1676  ]
 [146.6783  ]
 [185.57507 ]
 [153.49802 ]
 [148.66722 ]
 [192.02208 ]
 [148.4935  ]
 [173.43039 ]
 [174.95308 ]
 [155.4441  ]
 [172.72418 ]
 [177.84444 ]
 [163.3828  ]
 [146.92891 ]
 [193.8813  ]]

Step : 1600
Cost : 25.015398025512695
Prediction :
[[147.96088]
 [189.16324]
 [180.4908 ]
 [197.63876]
 [146.13248]
 [110.57783]
 [146.2475 ]
 [105.74997]
 [180.65149]
 [170.61383]
 [143.17867]
 [146.5424 ]
 [185.62115]
 [153.52887]
 [148.71896]
 [191.91101]
 [148.43472]
 [148.40472]
 [173.62   ]
 [175.0485 ]
 [155.56268]
 [172.79951]
 [177.7339 ]
 [163.48495]
 [147.1048 ]
 [193.80272]]

Step : 1700
Cost : 23.970928192138672
Prediction :
[[148.11708]
 [189.03767]
 [180.52931]
 [197.67143]
 [145.96645]
 [110.40455]
 [146.34615]
 [105.93608]
 [180.45015]
 [170.37888]
 [143.18977]
 [146.41154]
 [185.66466]
 [153.55685]
 [148.77022]
 [191.8038 ]
 [148.37526]
 [173.80522]
 [175.13966]
 [155.67651]
 [172.8735 ]
 [177.62721]
 [163.58466]
 [147.27315]
 [193.72597]]

Step : 1800
Cost : 22.987531661987305
Prediction :
[[148.26799 ]
 [188.9158  ]
 [180.56618 ]
 [197.70326 ]
 [145.805   ]
 [110.237236]
 [146.44272 ]
 [106.11835 ]
 [180.25574 ]
 [170.1533  ]
 [143.20094 ]
 [146.28557 ]
 [185.70569 ]
 [153.58215 ]
 [148.82101 ]
 [191.70029 ]
 [148.31525 ]
 [173.98622 ]
 [175.2268  ]
 [155.78581 ]
 [172.94617 ]
 [177.52426 ]
 [163.68199 ]
 [147.43427 ]
 [193.65103 ]]

Step : 1900
Cost : 22.06146240234375
Prediction :
[[148.41376 ]
 [188.7975  ]
 [180.60147 ]
 [197.73424 ]
 [145.64798 ]
 [110.075676]
 [146.53723 ]
 [106.29678 ]
 [180.068   ]
 [169.93672 ]
 [143.21213 ]
 [146.16425 ]
 [185.74438 ]
 [153.60492 ]
 [148.8713  ]
 [191.60034 ]
 [148.25476 ]
 [174.16304 ]
 [175.31006 ]
 [155.89072 ]
 [173.01752 ]
 [177.42491 ]
 [163.77698 ]
 [147.58847 ]
 [193.57788 ]]

Step : 2000
Cost : 21.18906021118164
Prediction :
[[148.55458 ]
 [188.68266 ]
 [180.63527 ]
 [197.76442 ]
 [145.49522 ]
 [109.919655]
 [146.62971 ]
 [106.47149 ]
 [179.8867  ]
 [169.72878 ]
 [143.22334 ]
 [146.0474  ]
 [185.78082 ]
 [153.62527 ]
 [148.92107 ]
```

```
sess = tf.Session()

sess.run(tf.global_variables_initializer())

for step in tqdm_notebook(range(2001)) :
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train], feed_dict = {X : x_data, Y : y_data})

    if step % 100 == 0 or step < 10 :
        # print("₩nStep : {} ₩nCost : {} ₩nPrediction :₩n{}".format(step, cost_val, hy_val))
        print("Step : {} ₩tCost : {} ".format(step, cost_val))
```

```
HBox(children=(IntProgress(value=0, max=2001), HTML(value='')))
Step : 0        Cost : 30871.0390625
Step : 1        Cost : 11448.2578125
Step : 2        Cost : 4267.099609375
Step : 3        Cost : 1612.0084228515625
Step : 4        Cost : 630.3294677734375
Step : 5        Cost : 267.35614013671875
Step : 6        Cost : 133.13693237304688
Step : 7        Cost : 83.49371337890625
Step : 8        Cost : 65.12068939208984
Step : 9        Cost : 58.30916976928711
Step : 100      Cost : 51.731849670410156
Step : 200      Cost : 49.04936981201172
Step : 300      Cost : 46.53648376464844
Step : 400      Cost : 44.1815299987793
Step : 500      Cost : 41.97384262084961
Step : 600      Cost : 39.903438568115234
Step : 700      Cost : 37.961181640625
Step : 800      Cost : 36.13839340209961
Step : 900      Cost : 34.42721939086914
Step : 1000     Cost : 32.8203010559082
Step : 1100     Cost : 31.310731887817383
Step : 1200     Cost : 29.892072677612305
Step : 1300     Cost : 28.558568954467773
Step : 1400     Cost : 27.304662704467773
Step : 1500     Cost : 26.125165939331055
Step : 1600     Cost : 25.015398025512695
Step : 1700     Cost : 23.970928192138672
Step : 1800     Cost : 22.987531661987305
Step : 1900     Cost : 22.06146240234375
Step : 2000     Cost : 21.18906021118164
```

```
print("Your score ₩t: ₩n", sess.run(hypothesis, feed_dict={X: [[100, 70, 101]]}))
```

```
Your score      :
 [[207.16539]]
```

```
print("₩nOther scores ₩t: ₩n", sess.run(hypothesis,
                                        feed_dict={X: [[60, 70, 110], [90, 100, 80]]}))
```

```
Other scores    :
 [[169.10136]
 [170.76434]]
```

Ex04. TF reader linear regression 5

```
import tensorflow as tf

tf.set_random_seed(777)

filename_queue = tf.train.string_input_producer(
['./data/data-01-test-scort.csv'], shuffle = False, name = 'filename_queue')

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)

record_defaults = [[0.],[0.],[0.],[0.]]
xy = tf.decode_csv(value, record_defaults = record_defaults)

train_x_batch, train_y_batch = ₩
    tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)



X = tf.placeholder(tf.float32, shape=[None,3])
Y = tf.placeholder(tf.float32, shape=[None,1])
```

```
WARNING: Logging before flag parsing goes to stderr.
W0911 11:14:16.658407  5588 deprecation.py:323] From <ipython-input-1-3a86c153738d>:6: string_
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.Dataset.from_tensor_
W0911 11:14:16.672038  5588 deprecation.py:323] From C:₩Python₩Anaconda3₩lib₩site-packages₩ter
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.Dataset.from_tensor_
W0911 11:14:16.673033  5588 deprecation.py:323] From C:₩Python₩Anaconda3₩lib₩site-packages₩ter
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.Dataset.from_tensors
W0911 11:14:16.676922  5588 deprecation.py:323] From C:₩Python₩Anaconda3₩lib₩site-packages₩ter
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
W0911 11:14:16.679851  5588 deprecation.py:323] From C:₩Python₩Anaconda3₩lib₩site-packages₩ter
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
W0911 11:14:16.685712  5588 deprecation.py:323] From <ipython-input-1-3a86c153738d>:8: TextLir
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.TextLineDataset`.
W0911 11:14:16.692548  5588 deprecation.py:323] From <ipython-input-1-3a86c153738d>:15: batch
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.Dataset.batch(batch_
W = tf.Variable(tf.random_normal([3,1]), name = 'weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.matmul(X, W) + b

cost = tf.reduce_mean(tf.square(hypothesis - Y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 1e-5)
train = optimizer.minimize(cost)
```

```
sess = tf.Session()

sess.run(tf.global_variables_initializer())

coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

Step_val = []
Cost_val = []

for step in tqdm_notebook(range(2001)):

    x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train], feed_dict={X: x_batch, Y: y_batch})

    Step_val.append(step)
    Cost_val.append(cost_val)

    if step % 100 == 0 or step < 10 :
        print("₩nStep : {} ₩nCost : {} ₩nPrediction :₩n{}".format(step, cost_val, hy_val))

coord.request_stop()
coord.join(threads)
```

```
W0911 11:14:20.996477  5588 deprecation.py:323] From <ipython-input-3-888db051c089>:6: start_c
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
```