선형 회귀(Linear Regression) 모델 구현하기

- 플레이스 홀더를 선언 후, 그래프 실행시에 데이터를 입력받아, 연산 수행

2-1. 라이브러리 불러오기 및 데이터 지정

In [6]:

```
import tensorflow as tf
```

In [7]:

```
x_data = [1,2,3,4,5]
y_data = [10,20,30,40,50]
```

2.2 W와 b를 각각 -1~1 사이의 균등분포(uniform distributiion)를 가진 무작위값으로 초기화수행

- 가중치(Weight)와 Bias를 임의의 값(-1 ~ 1)으로 초기화

In [8]:

```
W = tf.Variable(tf.random_uniform( [1], -1.0, 1.0) )
b = tf.Variable(tf.random_uniform( [1], -1.0, 1.0) )
print(W)
print(b)
```

```
<tf.Variable 'Variable_2:0' shape=(1,) dtype=float32_ref>
<tf.Variable 'Variable_3:0' shape=(1,) dtype=float32_ref>
```

2.3 플레이스 홀더(placeholder) 설정(이름 지정 - name)

- 플레이스 홀더는 나중에 데이터를 할당되는 심플한 변수이다.
- 데이터 없이도 텐서 그래프의 작성이 가능하다.
- feed_dict에 의해 나중에 값을 정의할 수 있다.
- 배열, matrix, 몇몇의 숫자 등의 다양한 형태의 값을 가질 수 있다.
- None은 임의의 행의 데이터를 가르킨다.

In [9]:

```
T = tf.placeholder(tf.float32)
print(T)
```

```
Tensor("Placeholder_1:0", dtype=float32)
```

In [10]:

```
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

print(X)
print(Y)
```

```
Tensor("X:0", dtype=float32)
Tensor("Y:0", dtype=float32)
```

2-4 선형관계 수식 작성

In [14]:

```
# 선형관계의 수식을 작성.
# W : 가중치(Weight), b : 편향(bias)
hypothesis = W * X + b
```

In [15]:

```
# hypothesis = W * X + b
```

2-5 손실함수(loss function)

- 우리는 나중에 학습시에 Loss를 최소화하는 W와 b의 값을 구하게 된다.
- 데이터에 대한 손실값을 계산하는 함수
- 손실값이란 실제값과 모델이 예측한 값이 얼마나 차이가 나는가를 나타내는 값.
- 손실값이 적을 수록 모델이 주어진 X값에 대한 Y값을 정확하게 예측할 수 있다라는 의미
- 손실을 전체 데이터에 대해 구한 경우 이를 비용(cost)이라 한다.

In [16]:

```
# hypothesis(예측) - Y(실제)
# tf.square(예측과실제의차이) -> 제곱
# tf.reduce_mean(a) -> a의 평균
cost = tf.reduce_mean(tf.square(hypothesis - Y))
cost
```

Out[16]:

```
<tf.Tensor 'Mean:0' shape=() dtype=float32>
```

2-6 최적화 함수(경사하강법)

- **경사하강법** : 함수의 기울기를 구하고, 기울기가 낮은 쪽으로 계속 이동시키면서 최적의 값을 찾아 나간다. (즉 손실값을 낮춰가며, 계속 최적의 값을 찾아간다.)
- 경사하강법(gradient descent)는 최적화 방법 중 가장 기본적인 알고리즘이다.
- 최적화 함수란 가중치(w)와 편향(b)을 변경해 가면서 손실값을 최소화시키는 가장 최적화된 가중치와 편향값을 찾아주는 함수.
- learning_rate는 학습을 얼마나 급하게 할 것인가를 설정하는 값

In [19]:

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train_op = optimizer.minimize(cost)
train_op
```

Out[19]:

```
<tf.Operation 'GradientDescent_1' type=NoOp>
```

- 학습을 진행하는 과정 중에 영향을 주는 변수를 **하이퍼파라미터(hyperparameter)**라 한다. 이에 따라 학습 속도와 신경망 성능이 달라질 수 있다

with를 이용하여 세션 블록(세션영역)을 생성

- 출력 순서
- step : 단계
- cost_val : cost 비용
- sess.run(W) : 가중치 값
- sess.run(b) : 편향 값

In [23]:

```
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())

  for step in range(100):
    _, cost_val = sess.run([train_op, cost], feed_dict={X:x_data, Y:y_data})
    print(step, cost_val, sess.run(W), sess.run(b))
```

```
0 1026.3518 [2.3315027] [1.0701916]
1 598.7691 [3.9543605] [1.5088975]
2 349.59058 [5.193867] [1.841458]
3 204.37727 [6.140729] [2.0929968]
4 119.74968 [6.8641887] [2.2826931]
5 70.428566 [7.4171057] [2.425188]
6 41.68227 [7.839831] [2.531658]
7 24.926077 [8.163169] [2.6106348]
8 15.15708 [8.410634] [2.668632]
9 9.459925 [8.600177] [2.7106214]
10 6.1356797 [8.745501] [2.7403984]
11 4.194274 [8.857066] [2.7608604]
12 3.05874 [8.94286] [2.7742193]
13 2.3928604 [9.008977] [2.7821634]
14 2.0007017 [9.060072] [2.7859814]
15 1.7680721 [9.099697] [2.7866576]
16 1.6284416 [9.130565] [2.7849426]
17 1.5430354 [9.154744] [2.78141]
18 1.4892559 [9.173816] [2.7764971]
19 1.4539309 [9.188987] [2.7705383]
20 1.4293896 [9.201178] [2.7637885]
21 1.411157 [9.211091] [2.756442]
22 1.3966323 [9.219264] [2.7486477]
23 1.3842895 [9.226107] [2.7405188]
24 1.3732473 [9.231932] [2.732142]
25 1.3629856 [9.236979] [2.7235832]
26 1.353211 [9.241428] [2.7148929]
27 1.3437395 [9.24542] [2.7061093]
28 1.3344724 [9.249062] [2.6972618]
29 1.3253498 [9.252432] [2.6883729]
30 1.3163357 [9.255594] [2.6794596]
31 1.3074131 [9.258596] [2.6705348]
32 1.298562 [9.261473] [2.6616085]
33 1.2897831 [9.264252] [2.652688]
34 1.2810705 [9.266955] [2.6437793]
35 1.2724174 [9.269598] [2.6348863]
36 1.2638242 [9.272193] [2.6260126]
37 1.2552941 [9.27475] [2.6171608]
38 1.2468187 [9.277275] [2.6083326]
39 1.2384017 [9.279775] [2.5995295]
40 1.2300417 [9.282252] [2.5907524]
41 1.2217375 [9.284712] [2.5820022]
42 1.2134911 [9.287155] [2.5732794]
43 1.2052988 [9.289584] [2.5645845]
44 1.1971629 [9.292001] [2.5559177]
45 1.1890804 [9.294406] [2.5472794]
46 1.1810533 [9.2968] [2.5386693]
47 1.1730818 [9.299184] [2.530088]
48 1.1651633 [9.3015585] [2.5215352]
49 1.157297 [9.303924] [2.513011]
50 1.149483 [9.30628] [2.5045154]
```

```
51 1.1417232 [9.308627] [2.4960482]
52 1.1340168 [9.3109665] [2.4876096]
53 1.1263626 [9.313297] [2.4791994]
54 1.1187587 [9.31562] [2.4708176]
55 1.1112064 [9.317935] [2.462464]
56 1.1037047 [9.320242] [2.4541388]
57 1.0962536 [9.32254] [2.4458416]
58 1.0888536 [9.324831] [2.4375722]
59 1.0815051 [9.327114] [2.4293308]
60 1.0742028 [9.32939] [2.4211173]
61 1.0669523 [9.331656] [2.4129317]
62 1.0597479 [9.333916] [2.4047737]
63 1.0525959 [9.336168] [2.3966434]
64 1.04549 [9.338412] [2.3885405]
65 1.0384312 [9.340649] [2.380465]
66 1.0314218 [9.342878] [2.3724167]
67 1.0244595 [9.3451] [2.3643956]
68 1.0175442 [9.347315] [2.3564017]
69 1.0106752 [9.349522] [2.3484347]
70 1.0038526 [9.351721] [2.3404946]
71 0.9970744 [9.353912] [2.3325815]
72 0.99034566 [9.356097] [2.324695]
73 0.98365945 [9.358274] [2.3168354]
74 0.9770201 [9.360444] [2.3090022]
75 0.9704248 [9.362606] [2.3011954]
76 0.9638726 [9.364761] [2.293415]
77 0.95736694 [9.366909] [2.285661]
78 0.95090485 [9.369049] [2.2779331]
79 0.94448584 [9.371182] [2.2702315]
80 0.9381086 [9.373308] [2.2625558]
81 0.93177825 [9.375427] [2.2549062]
82 0.9254862 [9.377539] [2.2472825]
83 0.91923726 [9.379643] [2.2396846]
84 0.9130341 [9.381741] [2.2321122]
85 0.90687025 [9.383831] [2.2245655]
86 0.9007481 [9.385914] [2.2170444]
87 0.8946684 [9.38799] [2.2095487]
88 0.8886293 [9.390059] [2.2020783]
89 0.8826289 [9.392121] [2.1946332]
90 0.87667257 [9.3941765] [2.1872132]
91 0.8707539 [9.396225] [2.1798184]
92 0.8648766 [9.398267] [2.1724486]
93 0.85903853 [9.400301] [2.1651037]
94 0.8532394 [9.4023285] [2.1577835]
95 0.84747887 [9.404349] [2.1504881]
96 0.8417565 [9.4063635] [2.1432173]
97 0.8360742 [9.40837] [2.135971]
98 0.83043116 [9.41037] [2.1287494]
99 0.82482624 [9.412363] [2.1215522]
```

생각해 보기

- 만약 for문을 계속 반복시켜가면 W의 값은 어떤 값에 가까워질까?

2-7 학습 후, 결과값 확인하기

In [25]:

```
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())

  for step in range(1000):
    _, cost_val = sess.run([train_op, cost], feed_dict={X:x_data, Y:y_data})
    if step%10==0:
      print(step, cost_val, sess.run(W), sess.run(b))

  print("X:5, Y:", sess.run(hypothesis, feed_dict={X:5}))
  print("X:2.5, Y:", sess.run(hypothesis, feed_dict={X:2.5}))
```

```
0 1039.3738 [2.6517003] [-0.26884192]
10 5.153329 [9.097385] [1.4576843]
20 0.45197162 [9.54419] [1.524576]
30 0.40266055 [9.587377] [1.4815658]
40 0.37620148 [9.603] [1.432751]
50 0.35156274 [9.616346] [1.3850754]
60 0.32853892 [9.62913] [1.3389549]
70 0.3070224 [9.64148] [1.294368]
80 0.28691462 [9.653419] [1.251266]
90 0.2681258 [9.66496] [1.2095994]
100 0.25056562 [9.676118] [1.1693197]
110 0.23415561 [9.686902] [1.1303815]
120 0.21882097 [9.697329] [1.0927402]
130 0.20449033 [9.707408] [1.056352]
140 0.191098 [9.717151] [1.0211755]
150 0.1785821 [9.72657] [0.9871706]
160 0.16688669 [9.735675] [0.95429766]
170 0.15595776 [9.744476] [0.9225198]
180 0.14574356 [9.752986] [0.8917999]
190 0.13619809 [9.761211] [0.86210257]
200 0.12727863 [9.769163] [0.83339447]
210 0.11894318 [9.77685] [0.80564255]
220 0.111153305 [9.784281] [0.77881473]
230 0.10387361 [9.791464] [0.75288033]
240 0.09707107 [9.7984085] [0.7278098]
250 0.09071436 [9.805121] [0.7035738]
260 0.08477298 [9.81161] [0.68014485]
270 0.0792213 [9.817884] [0.65749615]
280 0.07403344 [9.823948] [0.6356019]
290 0.06918495 [9.829811] [0.6144367]
300 0.064653434 [9.835477] [0.5939763]
310 0.060419608 [9.840958] [0.5741971]
320 0.056462485 [9.846253] [0.555076]
330 0.052764416 [9.851373] [0.5365918]
340 0.049308795 [9.856322] [0.5187233]
350 0.046079833 [9.861107] [0.5014498]
360 0.043061882 [9.865732] [0.48475152]
370 0.04024168 [9.870203] [0.46860912]
380 0.03760634 [9.874524] [0.4530046]
390 0.035143573 [9.878703] [0.43791974]
400 0.032841504 [9.882743] [0.42333698]
410 0.030690923 [9.886647] [0.40923983]
420 0.02868109 [9.890422] [0.39561218]
430 0.02680264 [9.894071] [0.3824385]
440 0.025047457 [9.897598] [0.36970338]
450 0.023406867 [9.901008] [0.3573922]
```

```
460 0.021873895 [9.9043045] [0.3454912]
470 0.020441344 [9.907491] [0.3339864]
480 0.019102853 [9.910571] [0.32286468]
490 0.017851625 [9.913549] [0.3121134]
500 0.016682671 [9.916428] [0.3017203]
510 0.015590018 [9.91921] [0.29167345]
520 0.014568965 [9.921903] [0.28196073]
530 0.013614802 [9.924502] [0.272571]
540 0.012723058 [9.927016] [0.2634941]
550 0.011889997 [9.929447] [0.2547197]
560 0.01111119 [9.931796] [0.24623753]
570 0.010383541 [9.934067] [0.23803782]
580 0.00970357 [9.936263] [0.2301112]
590 0.00906822 [9.938385] [0.22244863]
600 0.008474186 [9.940437] [0.21504128]
610 0.007919246 [9.942421] [0.20788047]
620 0.0074005155 [9.944338] [0.20095801]
630 0.006916047 [9.946192] [0.19426626]
640 0.0064629926 [9.947984] [0.18779701]
650 0.0060396586 [9.949716] [0.18154319]
660 0.005644145 [9.95139] [0.17549787]
670 0.005274383 [9.953009] [0.16965367]
680 0.0049290294 [9.954574] [0.16400425]
690 0.0046062768 [9.956086] [0.1585428]
700 0.00430462 [9.957549] [0.15326326]
710 0.004022652 [9.958962] [0.14815961]
720 0.003759252 [9.960329] [0.1432259]
730 0.0035129078 [9.96165] [0.13845617]
740 0.0032829705 [9.962927] [0.13384567]
750 0.003067917 [9.964161] [0.12938865]
760 0.002867043 [9.965355] [0.12508015]
770 0.0026792635 [9.966508] [0.12091497]
780 0.002503737 [9.967624] [0.11688866]
790 0.002339792 [9.968701] [0.11299627]
800 0.0021865538 [9.969744] [0.10923375]
810 0.002043493 [9.970751] [0.10559649]
820 0.001909524 [9.971725] [0.1020804]
830 0.001784602 [9.972667] [0.09868106]
840 0.001667643 [9.9735775] [0.09539487]
850 0.0015584955 [9.974457] [0.09221815]
860 0.0014563377 [9.975307] [0.08914735]
870 0.0013610256 [9.97613] [0.08617876]
880 0.0012718763 [9.976925] [0.08330899]
890 0.0011885648 [9.977694] [0.08053476]
900 0.0011107274 [9.978436] [0.07785293]
910 0.0010379642 [9.979154] [0.07526045]
920 0.00097001885 [9.979848] [0.07275448]
930 0.00090644154 [9.980519] [0.07033175]
940 0.0008471029 [9.981168] [0.06798965]
950 0.00079165085 [9.981794] [0.06572589]
960 0.0007398249 [9.982402] [0.06353734]
970 0.00069135585 [9.982987] [0.06142158]
980 0.00064607814 [9.983554] [0.05937638]
990 0.0006037728 [9.984101] [0.0573993]
X:5, Y: [49.978565]
X:2.5, Y: [25.01712]
```

In [0]: