



딥러닝 하루만에 끝내기

엑셈 연구컨텐츠팀

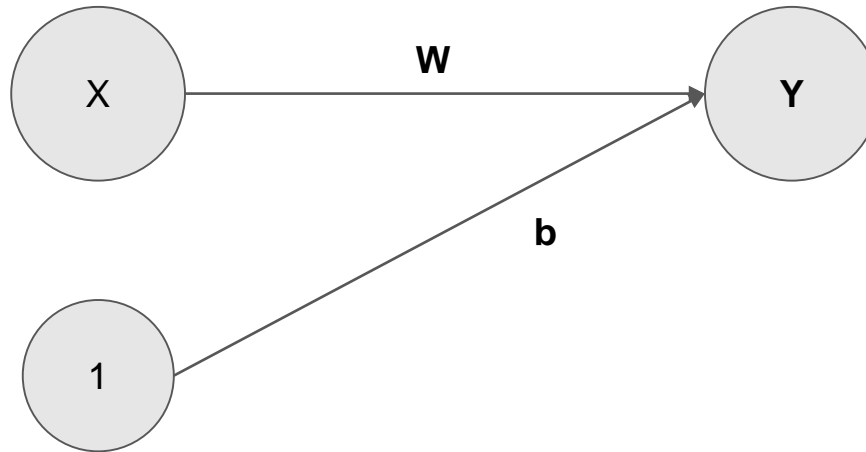
- Linear Regression
- multi-variable Linear Regression
- Softmax classifier
- 1,2,3 – CNN
- MNIST – CNN

<https://github.com/hunkim/DeepLearningZeroToAll>

처음 배우는 딥러닝 수학

Linear Regression 아키텍처

입력 변수가 1개인 데이터를 직선 방정식을 통해 표현되지 않은 데이터를 예측하기 위한 분석 모델.



Linear Regression 모델정의

```
# Model parameters
W = tf.Variable([.0], tf.float32)
b = tf.Variable([.0], tf.float32)

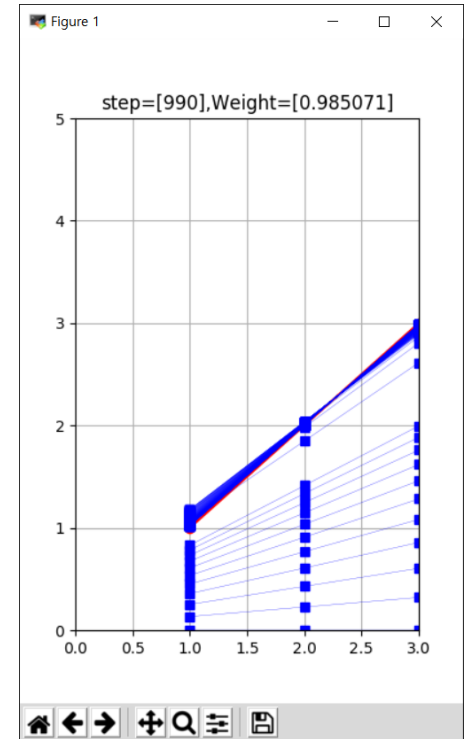
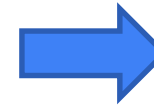
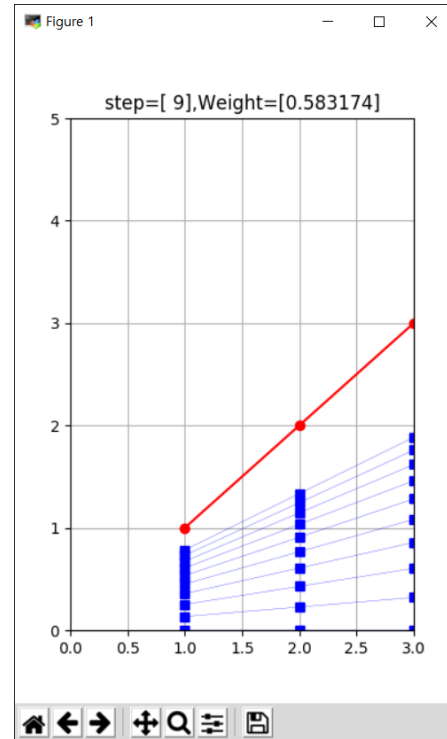
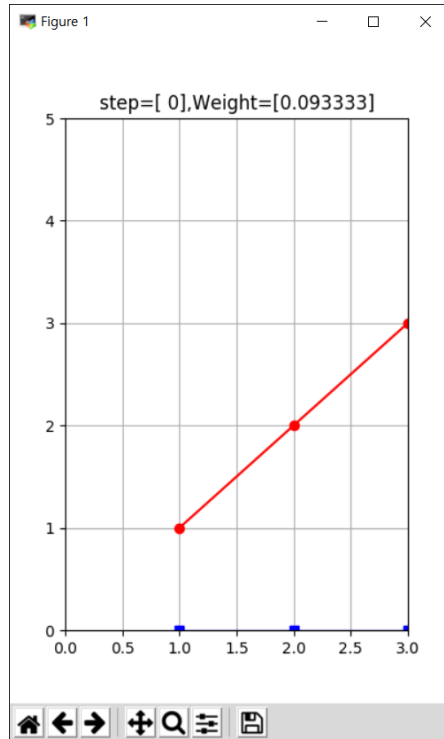
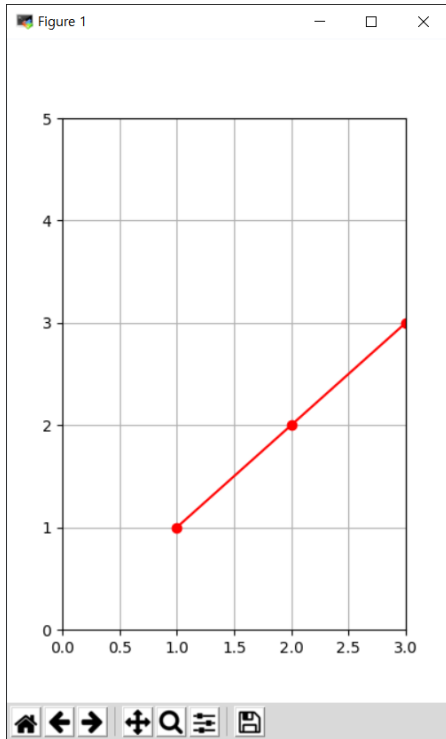
# Model input and output
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

linear_model = x * W + b

# cost/loss function
loss = tf.reduce_mean(tf.square(linear_model - y))  # sum of the squares

# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
```

Linear Regression 학습



Linear Regression 학습

```
(tf36) opm@opml:~/work/DeepLearningZeroToAll$ py lab-02-3-linear_regression_tensorflow.org.py
```

```
X :  1 2 3
```

```
Y :  1 2 3
```

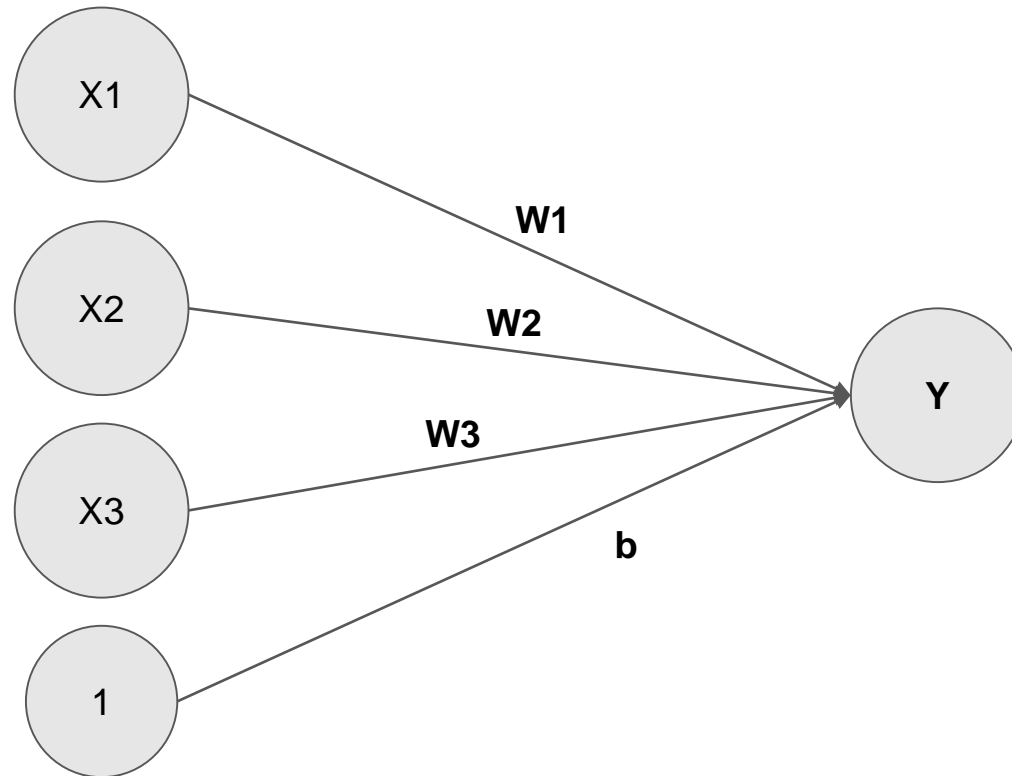
step	weight	bais	loss	H1	H2	H3
-1	0.0000000	0.0000000				
0	0.0933333	0.0400000	4.6666665	0.0000000	0.0000000	0.0000000
....						
1	0.1763556	0.0754667	3.6927407	0.1333333	0.2266667	0.3200000
....						
2	0.2502104	0.1069031	2.9228852	0.2518222	0.4281778	0.6045333
....						
3	0.3159146	0.1347566	2.3143361	0.3571135	0.6073239	0.8575343
....						
4	0.3743723	0.1594249	1.8332924	0.4506713	0.7665859	1.0825005
....						
5	0.4263873	0.1812615	1.4530338	0.5337973	0.9081696	1.2825419
....						
6	0.4726740	0.2005808	1.1524409	0.6076487	1.0340360	1.4604232
....						
7	0.5138679	0.2176622	0.9148195	0.6732548	1.1459287	1.6186028
....						
8	0.5505337	0.2327543	0.7269740	0.7315301	1.2453979	1.7592658
....						
Accu	0.9850714	0.0339363	0.0001660	1.0190535	2.0040889	2.9891241



Microsoft Excel
97-2003 워크시트

multi-variable Linear Regression 아키텍처

입력 변수가 2개이상인 데이터를 직선 방정식을 통해 표현되지 않은 데이터를 예측하기 위한 분석 모델.



multi-variable Linear Regression 모델정의

```
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

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

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

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

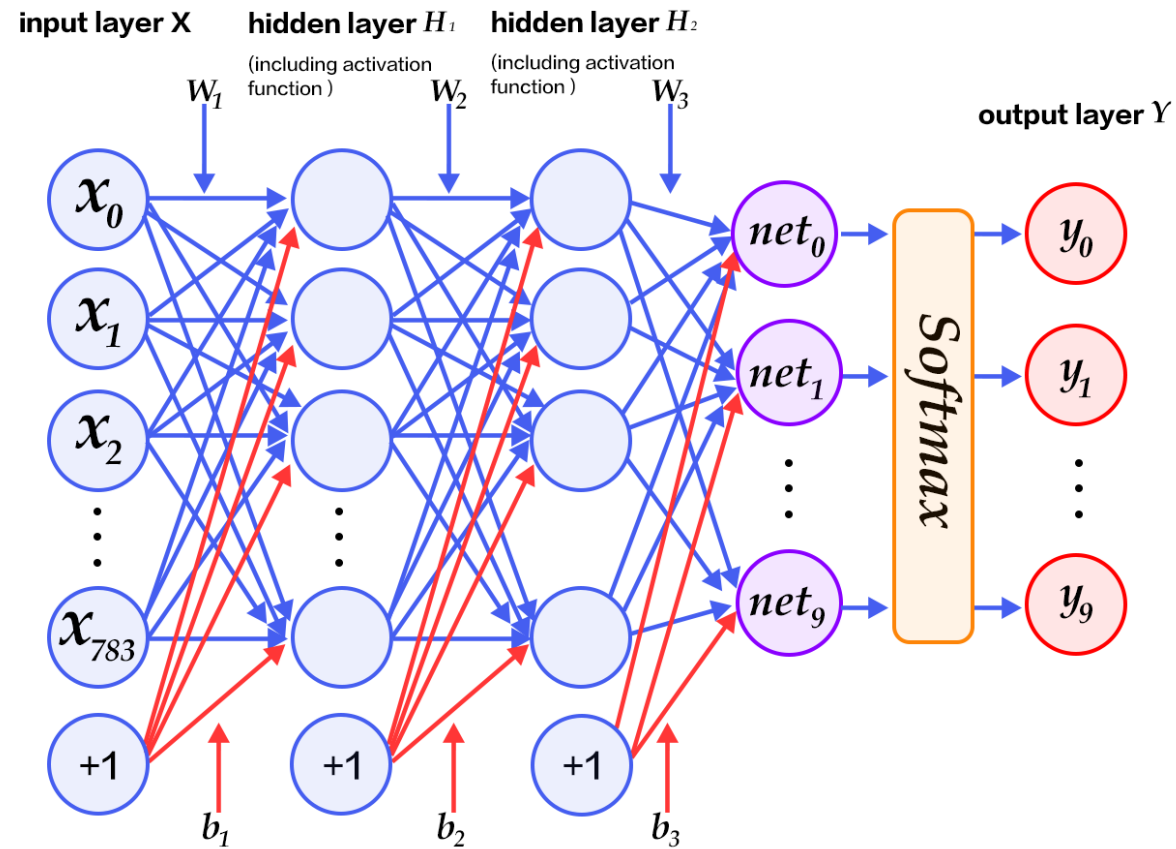
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)
```


multi-variable Linear Regression 학습

```
(tf36) opm@opml:~/work/DeepLearningZeroToAll$ py lab-04-2-multi_variable_matmul_linear_regression.py
=====
[[ 73.  80.  75. 152.]
 [ 93.  88.  93. 185.]
 [ 89.  91.  90. 180.]
 [ 96.  98. 100. 196.]
 [ 73.  66.  70. 142.]]
      //W:  0.00000000  0.00000000  0.00000000  //b:  0.00000000
=====
0  //W:  0.2940120  0.2936000  0.2973800  //b:  0.0034200 //Cost:  29661.80078
      //H:      0.00000,      0.00000,      0.00000,      0.00000,      0.00000,
1  //W:  0.4586434  0.4579483  0.4638748  //b:  0.0053354 //Cost:   9298.52051
      //H:   67.25779,   80.83968,   79.65228,   86.73937,   61.66050,
2  //W:  0.5508393  0.5499333  0.5570918  //b:  0.0064084 //Cost:   2915.71289
      //H:  104.91276,  126.09896,  124.24662,  135.30151,   96.18212,
3  //W:  0.6024812  0.6014048  0.6092831  //b:  0.0070099 //Cost:    915.04053
      //H:  125.99422,  151.43813,  149.21329,  162.48962,  115.50970,
4  //W:  0.6314185  0.6301942  0.6385059  //b:  0.0073472 //Cost:    287.93600
      //H:  137.79675,  165.62471,  163.19115,  177.71118,  126.33068,
...
9  //W:  0.6664166  0.6645633  0.6736558  //b:  0.0077586 //Cost:     2.49314
      //H:  151.98238,  182.67888,  179.99278,  196.00789,  139.33961,
```

Softmax classifier 아키텍처

Softmax 를 이용해서 다수개의 분류로 데이터를 예측하기 위한 분석 모델.



Softmax classifier 모델정의

```
X = tf.placeholder("float", [None, 4])
Y = tf.placeholder("float", [None, 3])
nb_classes = 3

W = tf.Variable(tf.zeros([4, nb_classes]), name='weight')
b = tf.Variable(tf.zeros([nb_classes]), name='bias')

# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

diff=hypothesis - Y

# Cross entropy cost/loss
log_=tf.log(hypothesis)
cost = tf.reduce_mean(-tf.reduce_sum(Y * log_, axis=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

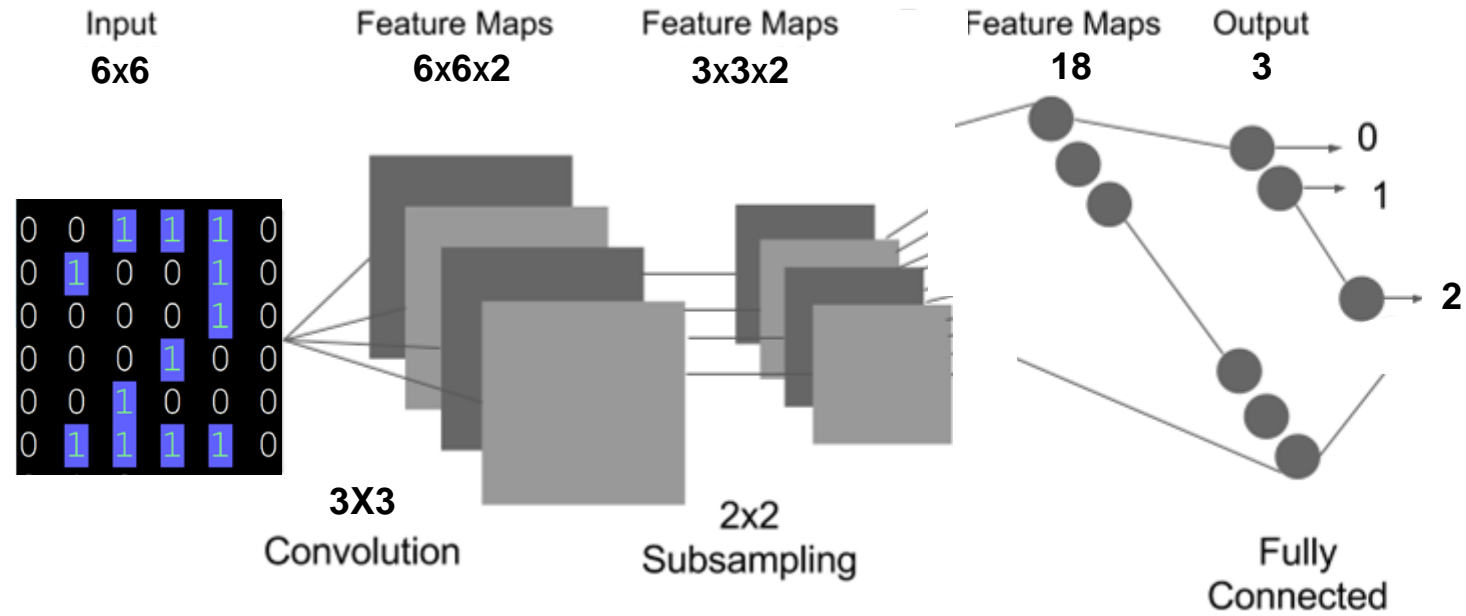
Softmax classifier 학습

```
(tf36) opm@opml:~/work/DeepLearningZeroToAll$ py lab-06-1-softmax_classifier.py
X 2 (8, 4)
```

```
1 2 1 1
2 1 3 2
3 1 3 4
4 1 5 5
1 7 5 5
1 2 5 6
1 6 6 6
1 7 7 7
Y 2 (8, 3)
0 0 1
0 0 1
0 0 1
0 1 0
0 1 0
0 1 0
1 0 0
1 0 0
W 2 (4, 3)
0.0000000 0.0000000 0.0000000
0.0000000 0.0000000 0.0000000
0.0000000 0.0000000 0.0000000
0.0000000 0.0000000 0.0000000
b 1 (3,)
0.0000000 0.0000000 0.0000000
0 1.0986123
H 2 (8, 3)
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
0.3333333 0.3333333 0.3333333
H-Y 2 (8, 3)
0.3333333 0.3333333 -0.6666666
0.3333333 0.3333333 -0.6666666
0.3333333 0.3333333 -0.6666666
0.3333333 -0.6666666 0.3333333
0.3333333 -0.6666666 0.3333333
0.3333333 -0.6666666 0.3333333
0.3333333 -0.6666666 0.3333333
-0.6666666 0.3333333 0.3333333
-0.6666666 0.3333333 0.3333333
```

```
Log 2
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
-1.0986123 -1.0986123 -1.0986123
W 2 (4, 3)
-0.0333333 0.0166667 0.0166667
0.0500000 0.0125000 -0.0625000
0.0166667 0.0416667 -0.0583333
0.0125000 0.0500000 -0.0625000
b 1 (3,)
-0.0083333 0.0041667 0.0041667
1 1.0312738
H 2 (8, 3)
0.3593391 0.3777628 0.2628981
0.3401809 0.4260164 0.2338027
0.3288443 0.4666523 0.2045034
0.3174314 0.5168548 0.1657137
0.4259214 0.4766369 0.0974418
0.3604972 0.5052151 0.1342877
0.4069798 0.5033388 0.0896814
0.4102466 0.5202234 0.0695300
H-Y 2 (8, 3)
0.3593391 0.3777628 -0.7371019
0.3401809 0.4260164 -0.7661973
0.3288443 0.4666523 -0.7954966
0.3174314 -0.4831452 0.1657137
0.4259214 -0.5233631 0.0974418
0.3604972 -0.4947849 0.1342877
-0.5930202 0.5033388 0.0896814
-0.5897534 0.5202234 0.0695300
Log 2
-1.0234888 -0.9734887 -1.3359888
-1.0782776 -0.8532775 -1.4532776
-1.1121709 -0.7621709 -1.5871708
-1.1474935 -0.6599934 -1.7974935
-0.8535004 -0.7410003 -2.3285003
-1.0202711 -0.6827710 -2.0077710
-0.8989918 -0.6864917 -2.4114919
-0.8909969 -0.6534969 -2.6659970
W 2 (4, 3)
```

1,2,3 – CNN 아키텍처



1,2,3 – CNN 모델정의

```
# input place holders
X = tf.placeholder(tf.float32, [None, 36])
X_img = tf.reshape(X, [-1, 6, 6, 1]) # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 3])

# L1 ImgIn shape=(?, 6, 6, 1)
###W1 = tf.Variable(tf.zeros([3, 3, 1, 2]))
W1 = tf.Variable(tf.random_normal([3, 3, 1, 2], stddev=0.01))
# Conv -> (?, 6, 6, 2)
# Pool -> (?, 3, 3, 2)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L11 = tf.nn.relu(L1)
L12 = tf.nn.max_pool(L11, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

L2_flat = tf.reshape(L12, [-1, 3 * 3 * 2])

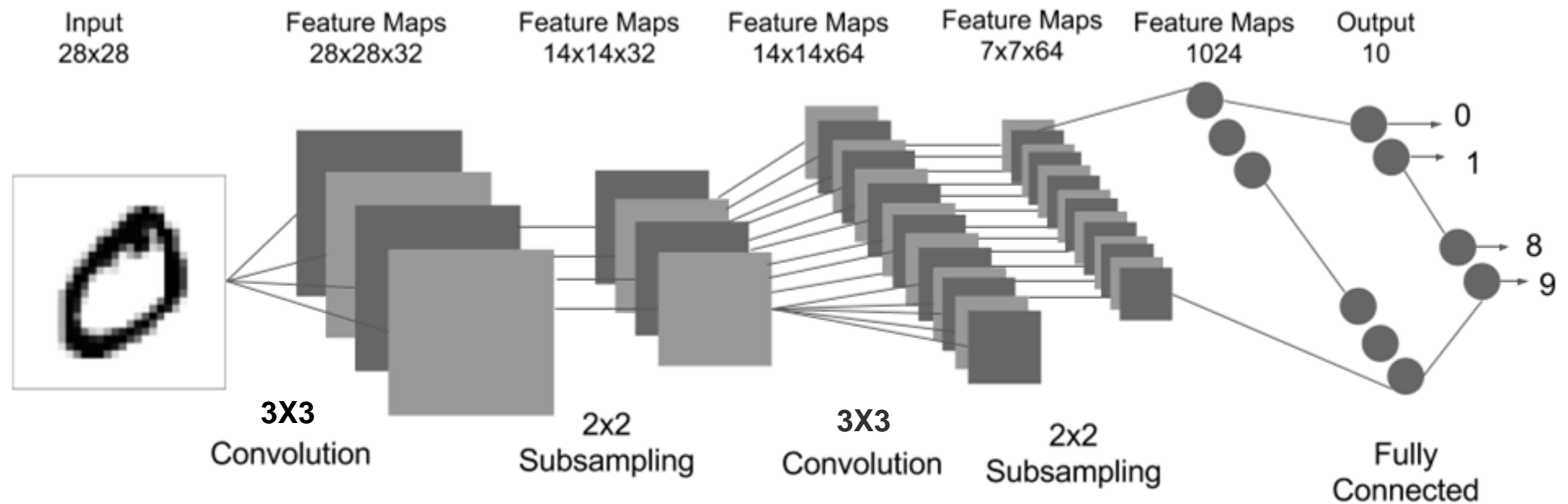
# Final FC 3x3x4 inputs -> 3 outputs
#####W3 = tf.Variable(tf.zeros([3 * 3 * 2, 3]))
W3 = tf.get_variable("W3", shape=[3 * 3 * 2, 3],
initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.zeros([3]))
####b = tf.Variable(tf.random_normal([3]))
logits = tf.matmul(L2_flat, W3) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

1,2,3 – CNN 학습

```
W1 (3, 3, 1, 2) 3
0.0058813 0.0083458 -0.0228624 |-0.0055283 -0.0007458 0.0026713 |
0.0203310 -0.0057832 -0.0105126 | 0.0017287 0.0156341 0.0077299 |
-0.0094315 0.0147203 -0.0078645 | 0.0031510 0.0061995 -0.0058039 |
W3 (18, 3) 18
-0.4459611 0.3958464 0.2539913 0.2090968 0.0659516 0.4497849 -0.1583560 -0.3866279 0.1737968 0.1767806 -0.1945766 0.1939116 0.4328153 -0.41256
0.1992146 0.1379229 0.5235286 -0.0413567 0.1206975 0.4771679 -0.4046668 0.2714725 0.0790120 -0.4295341 -0.4528130 -0.0522701 -0.3656027 -0.53380
0.0499464 0.3127901 -0.2819926 -0.4314600 0.3288334 0.2625230 0.0258836 0.2351133 -0.0245297 -0.0470927 0.0410023 -0.4904741 0.3526267 -0.16524
bias (3,)
0.6979344 -1.6097815 1.0620146
...Learning started. It takes sometime.
_X_img (1, 6, 6, 1)
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 |
W1 (3, 3, 1, 2) 3
-0.0041187 0.0183458 -0.0228624 |-0.0055283 -0.0107457 0.0026713 |
0.0103310 0.0042168 -0.0105126 | 0.0117287 0.0256341 0.0077299 |
-0.0194315 0.0247203 -0.0078645 | 0.0131510 0.0161994 -0.0058039 |
L1_conv (1, 6, 6, 2)
0.0000000 0.0000000 -0.0183772 0.0089371 0.0108995 0.0000000 |
0.0000000 0.0000000 -0.0412395 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 -0.0412395 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 -0.0412395 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 -0.0412395 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 -0.0412395 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 -0.0333750 0.0025626 0.0262122 0.0000000 |
=====
0.0000000 0.0000000 0.0019260 0.0218336 0.0048797 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 -0.0006486 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 -0.0006486 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 -0.0006486 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 -0.0006486 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 -0.0006486 0.0000000 |
0.0000000 0.0000000 0.0104011 0.0148884 -0.0037996 0.0000000 |
L1_relu (1, 6, 6, 2)
0.0000000 0.0000000 0.0000000 0.0089371 0.0108995 0.0000000 |
0.0000000 0.0000000 0.0000000 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 0.0000000 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 0.0000000 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 0.0000000 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 0.0000000 0.0172829 0.0167807 0.0000000 |
0.0000000 0.0000000 0.0025626 0.0262122 0.0000000 |
=====
0.0000000 0.0000000 0.0019260 0.0218336 0.0048797 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0045973 0.0210878 0.0000000 0.0000000 |
0.0000000 0.0000000 0.0104011 0.0148884 0.0000000 0.0000000 |
L1_maxpool (1, 3, 3, 2)
0.0000000 0.0172829 0.0167807 |
0.0000000 0.0172829 0.0167807 |
0.0000000 0.0172829 0.0262122 |
=====
0.0000000 0.0218336 0.0048797 |
0.0000000 0.0210878 0.0000000 |
0.0000000 0.0210878 0.0000000 |
L2_flat (1, 18)
0.0000000 0.0000000 0.0172829 0.0218336 0.0167807 0.0048797 0.0000000 0.0000000 0.0172829 0.0210878 0.0167807 0.0000000 0.0000000 0.00000
W3 (18, 3)
-0.4459611 0.3958464 0.2639910 0.2190966 0.0759513 0.4597838 -0.1583560 -0.3866279 0.1837965 0.1867803 -0.1845769 0.1939116 0.4328153 -0.41256
0.1992146 0.1379229 0.5135333 -0.0513530 0.1107023 0.4671844 -0.4046668 0.2714725 0.0690166 -0.4395303 -0.4628082 -0.0522701 -0.3656027 -0.53380
```

MNIST – CNN 아키텍처



MNIST – CNN 모델정의

```
# hyper parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 1

# input place holders
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1]) # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])

# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#   Conv      -> (?, 28, 28, 32)
#   Pool      -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L11 = tf.nn.relu(L1)
L12 = tf.nn.max_pool(L11, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
'''

Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
'''
```

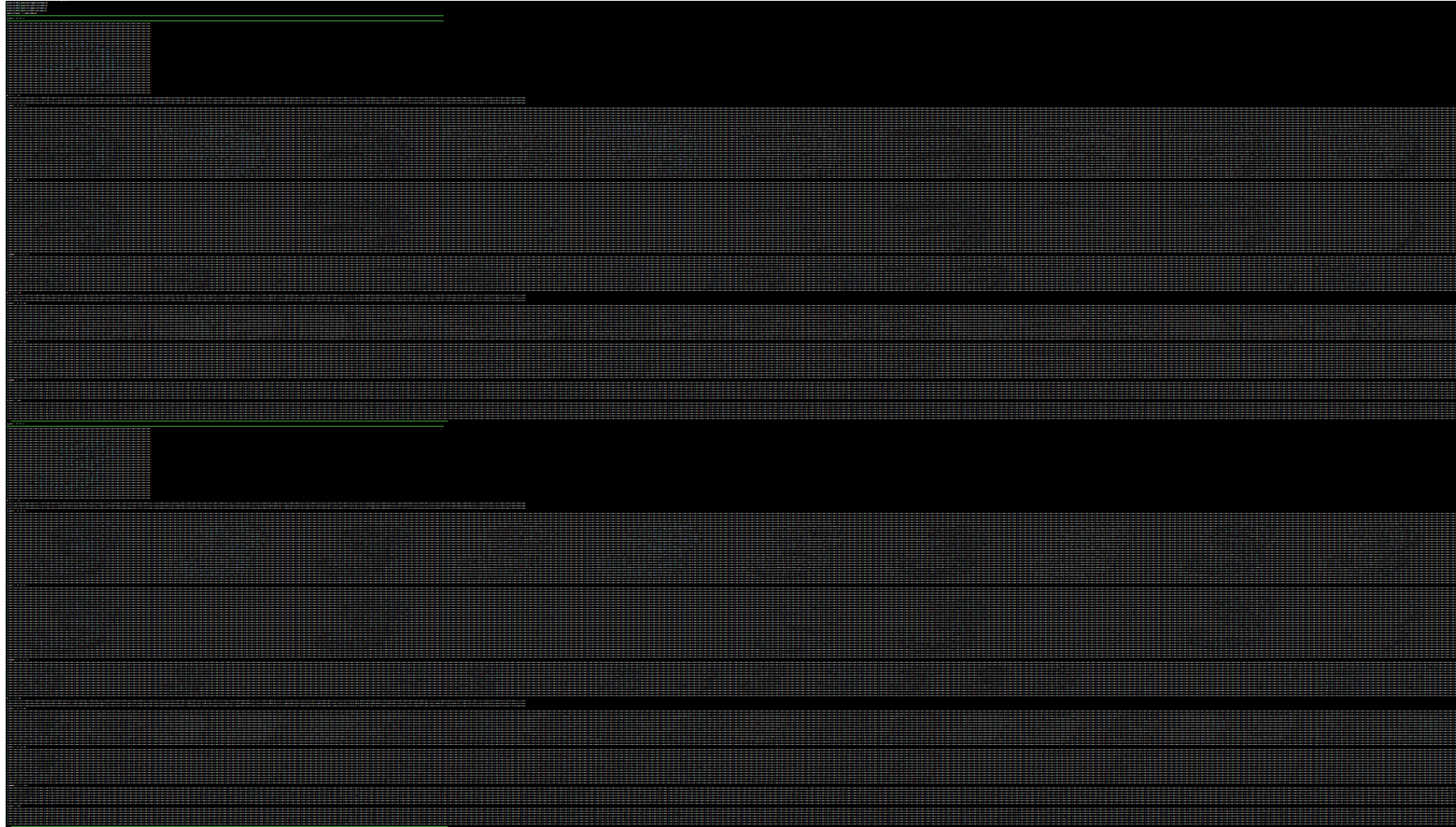
MNIST – CNN 모델정의

```
# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
# Conv ->(?, 14, 14, 64)
# Pool ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L12, W2, strides=[1, 1, 1, 1], padding='SAME')
L21 = tf.nn.relu(L2)
L22 = tf.nn.max_pool(L21, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L2_flat = tf.reshape(L22, [-1, 7 * 7 * 64])
'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
'''

# Final FC 7x7x64 inputs -> 10 outputs
W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10],
initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([10]))
logits = tf.matmul(L2_flat, W3) + b

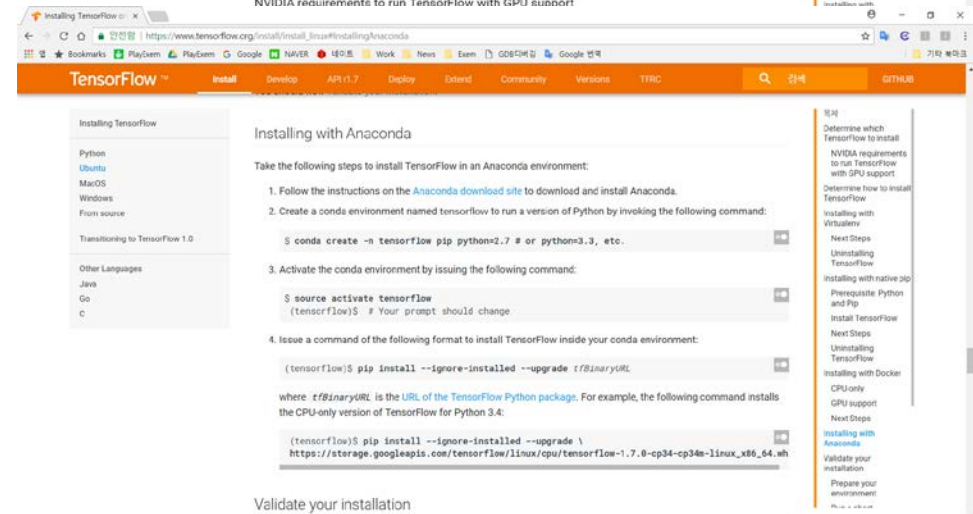
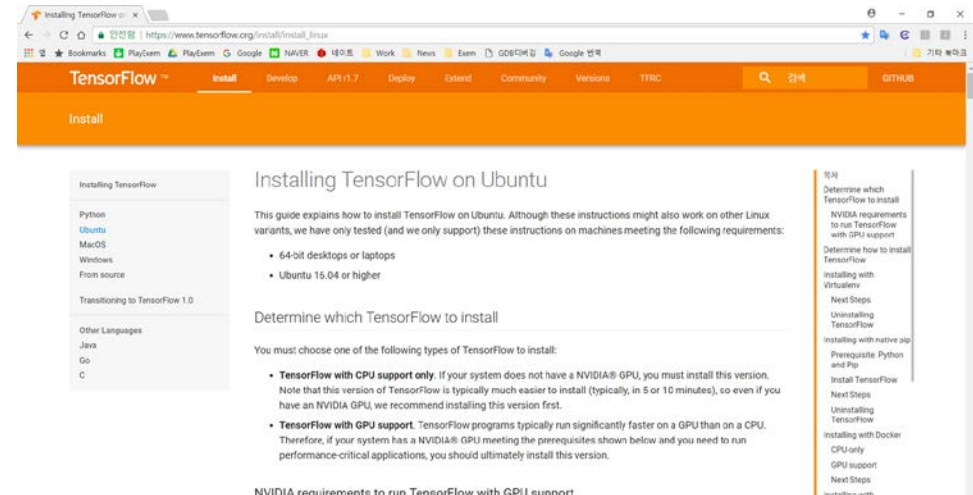
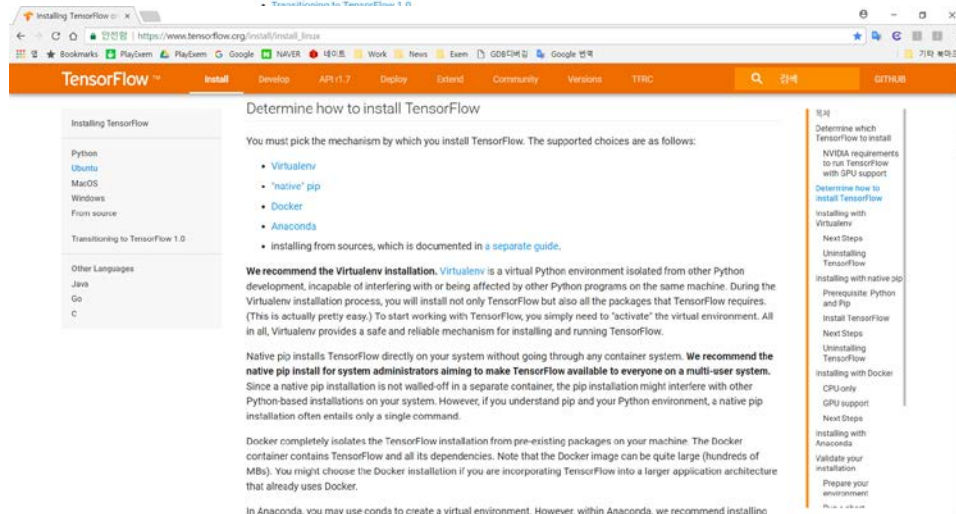
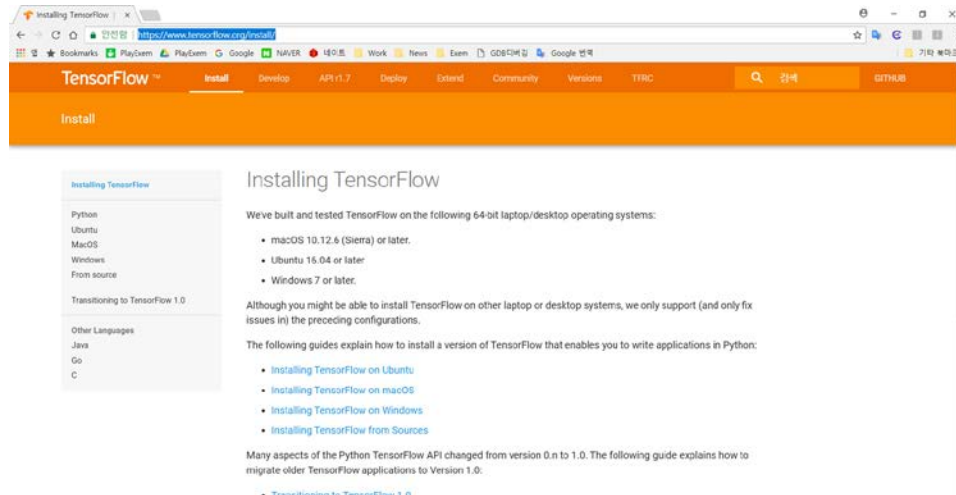
# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits( logits=logits, labels=Y ))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

MNIST – CNN 모델정의



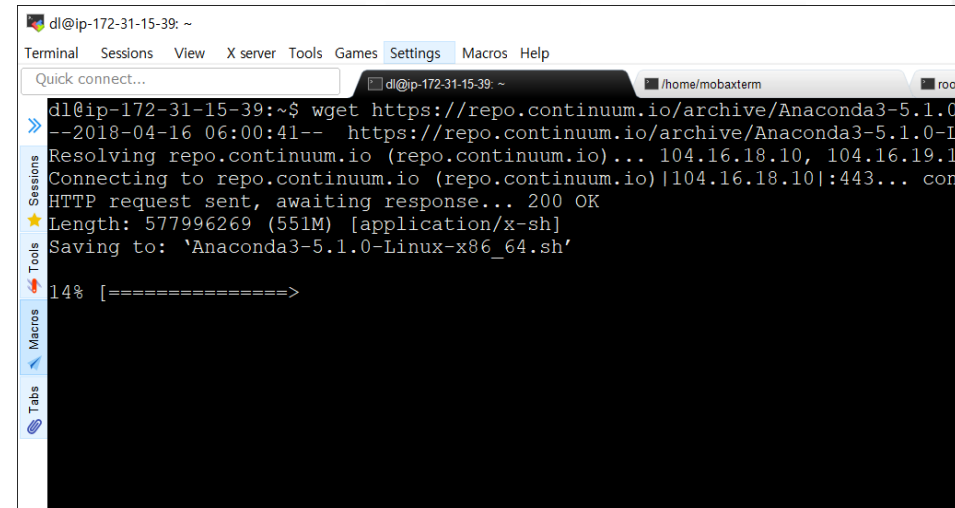
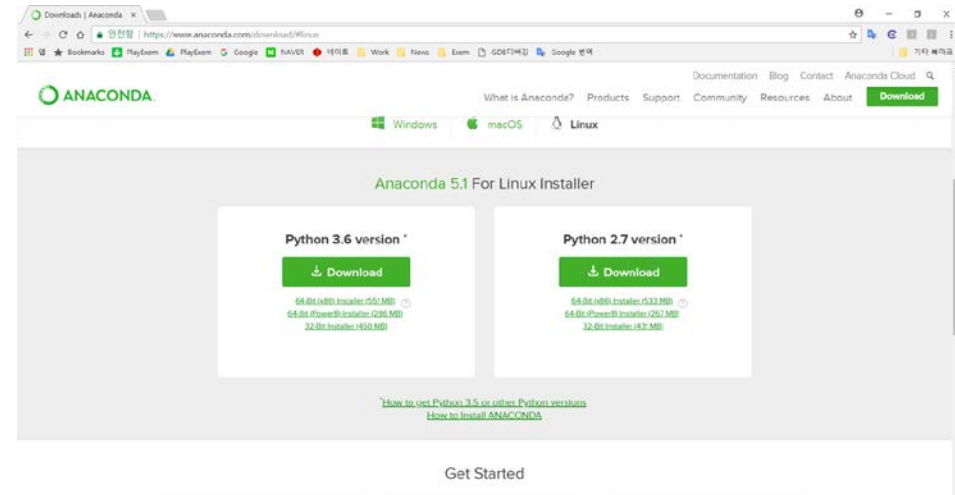
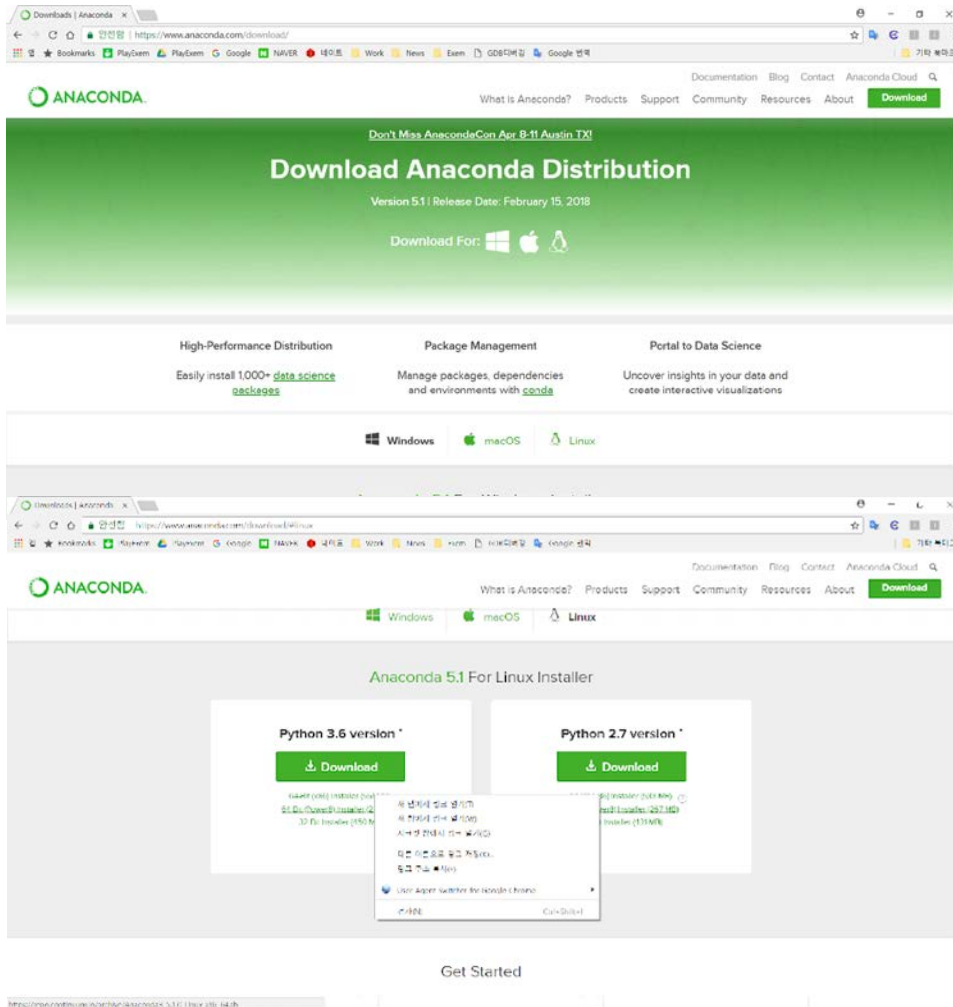
Tensorflow 설치

<https://www.tensorflow.org/install/>



Tensorflow 설치

<https://www.anaconda.com/download/>



행렬의 기본

$$\begin{array}{c}
 \left[\begin{array}{cccc}
 A_{11}, A_{12}, A_{13}, A_{14} \\
 A_{21}, A_{22}, A_{23}, A_{24} \\
 A_{31}, A_{32}, A_{33}, A_{34}
 \end{array} \right] \begin{array}{c} \downarrow \\ \text{3행} \end{array} \\
 \xrightarrow{\text{4열}}
 \end{array}
 \quad
 \begin{array}{c}
 \left[\begin{array}{ccc}
 A_{11}, A_{12}, A_{13} \\
 A_{21}, A_{22}, A_{23} \\
 A_{31}, A_{32}, A_{33}
 \end{array} \right] \times \left[\begin{array}{cc}
 B_{11}, B_{12} \\
 B_{21}, B_{22} \\
 B_{31}, B_{32}
 \end{array} \right] = \left[\begin{array}{cc}
 A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}, & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32} \\
 A_{21} \cdot B_{11} + A_{22} \cdot B_{21} + A_{23} \cdot B_{31}, & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} + A_{23} \cdot B_{32} \\
 A_{31} \cdot B_{11} + A_{32} \cdot B_{21} + A_{33} \cdot B_{31}, & A_{31} \cdot B_{12} + A_{32} \cdot B_{22} + A_{33} \cdot B_{32}
 \end{array} \right] \\
 \begin{array}{cc}
 3 \times 3 & 3 \times 2 \\
 (m \times k) & (k \times n)
 \end{array}
 \qquad \qquad \qquad \begin{array}{c}
 3 \times 2 \\
 (m \times n)
 \end{array}
 \end{array}$$

$$\begin{array}{l}
 \left[\begin{array}{ccc}
 A_{11}, A_{12}, A_{13} \\
 A_{21}, A_{22}, A_{23} \\
 A_{31}, A_{32}, A_{33}
 \end{array} \right] \times \left[\begin{array}{cc}
 B_{11}, B_{12} \\
 B_{21}, B_{22} \\
 B_{31}, B_{32}
 \end{array} \right] = \left[\begin{array}{c}
 A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31} \\
 \\
 \end{array} \right] \\
 \left[\begin{array}{ccc}
 A_{11}, A_{12}, A_{13} \\
 A_{21}, A_{22}, A_{23} \\
 A_{31}, A_{32}, A_{33}
 \end{array} \right] \times \left[\begin{array}{cc}
 B_{11}, B_{12} \\
 B_{21}, B_{22} \\
 B_{31}, B_{32}
 \end{array} \right] = \left[\begin{array}{c}
 A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}, \quad A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32} \\
 \\
 \end{array} \right] \\
 \left[\begin{array}{ccc}
 A_{11}, A_{12}, A_{13} \\
 A_{21}, A_{22}, A_{23} \\
 A_{31}, A_{32}, A_{33}
 \end{array} \right] \times \left[\begin{array}{cc}
 B_{11}, B_{12} \\
 B_{21}, B_{22} \\
 B_{31}, B_{32}
 \end{array} \right] = \left[\begin{array}{c}
 A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}, \quad A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32} \\
 A_{21} \cdot B_{11} + A_{22} \cdot B_{21} + A_{23} \cdot B_{31} \\
 \end{array} \right] \\
 \left[\begin{array}{ccc}
 A_{11}, A_{12}, A_{13} \\
 A_{21}, A_{22}, A_{23} \\
 A_{31}, A_{32}, A_{33}
 \end{array} \right] \times \left[\begin{array}{cc}
 B_{11}, B_{12} \\
 B_{21}, B_{22} \\
 B_{31}, B_{32}
 \end{array} \right] = \left[\begin{array}{c}
 A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}, \quad A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32} \\
 A_{21} \cdot B_{11} + A_{22} \cdot B_{21} + A_{23} \cdot B_{31}, \quad A_{21} \cdot B_{12} + A_{22} \cdot B_{22} + A_{23} \cdot B_{32} \\
 \end{array} \right]
 \end{array}$$



딥러닝 하루만에 끝내기

감사합니다.