

2017

기계학습 및 딥러닝 실습

MNIST

Weka, TensorFlow로 MNIST 데이터 분석하기

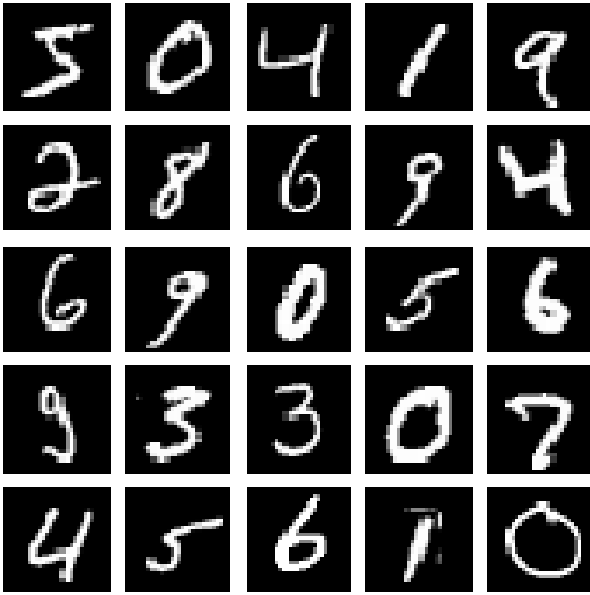


Index

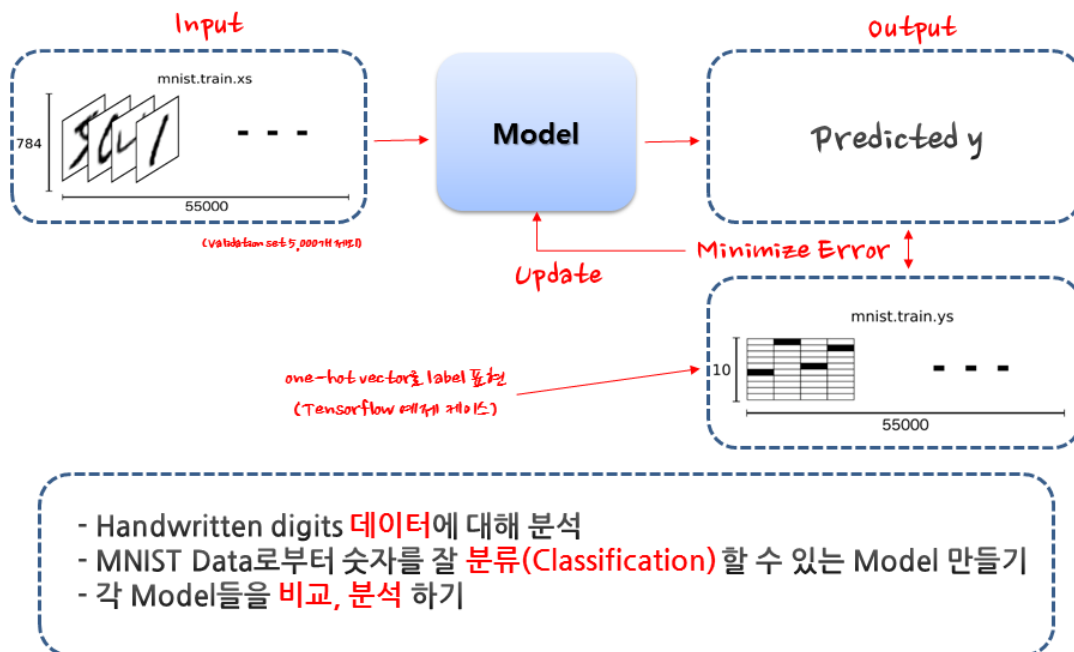
1. MNIST 문제 정의	4
1.1. 데이터 소개	5
MNIST란?	5
1.2. Previous experiments	5
1.3. MNIST DataSet 다운로드	6
Weka : arff type	6
http://axon.cs.byu.edu/data/mnist/train.arff	6
TensorFlow : raw data	7
2. Weka MNIST data 분석	9
2.1. Intro	9
2.1.1. MNIST Dataset 불러오기	9
2.1.2. Data 확인	9
2.1.3. Data Handling	10
2.1.4. MNIST Dataset Visualize	13
2.1.5. Simple classifier : Zero R, J48, User classifier	13
2.2. Evaluation	15
2.2.1. Test set	15
2.2.2. Cross-validation	16
Split	16
2.2.3. Baseline과 다양한 알고리즘 비교 : ZERO R, J48, naïve bayes, IBk	17
2.3. Rule Based Classifier	18
2.3.1. One R	18
2.3.2. PRISM	18
2.4. Linear Classifier	19

2.4.1. Logistic	19
2.4.2. Perceptron	20
3. Tensorflow MNIST data분석	21
3.1. Neural Network(NN)	22
3.1. CNN	오류! 책갈피가 정의되어 있지 않습니다.
Tensorboard 가이드	26
부록	32
4. 실습환경 설정	32
4.1. Weka	32
windows / mac 설치	32
ubuntu 설치	32
unofficial package 다운로드	32
4.2. Tensorflow	34
Windows 설치	34
A. Docker installation	34
B. Anaconda installation 이용	37

1. MNIST 문제 정의



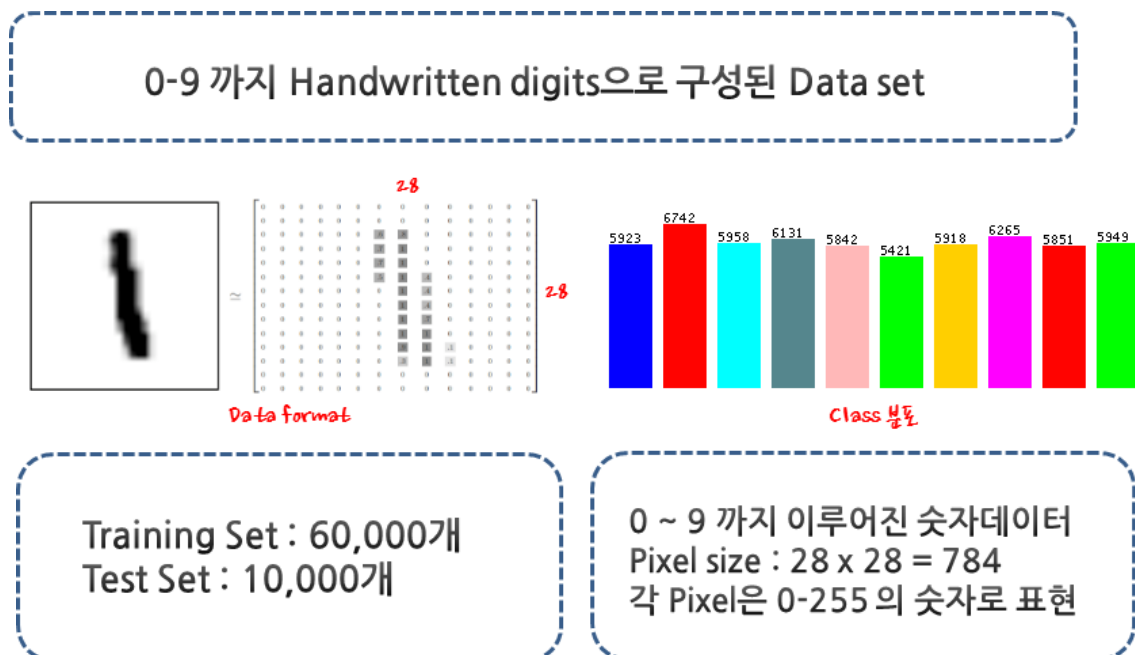
MNIST는 위와 같이 손으로 쓰여진 0부터 9까지의 숫자를 스캔한 데이터 셋입니다. 이 데이터 각각은 28*28 pixel로 이루어져 있으며 각 데이터는 어떤 숫자를 의미하는지에 대한 값을 가지고 있습니다. 데이터는 총 70000개이며, 60000개의 training set, 10000개의 test set으로 구성되어 있습니다. 이 데이터를 가지고 특정 모델을 학습시킨 후 임의의 숫자 이미지 데이터를 넣어 이미지 데이터가 0~9중 어느 숫자인지를 예측합니다.



1.1. 데이터 소개

MNIST란?

Mixed National Institute of Standards and Technology database의 약어로, Yann LeCun 교수가 제공하는 데이터 셋이다.



1.2. Previous experiments

1. <http://yann.lecun.com/exdb/mnist/>

CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
K-Nearest Neighbors			
K-nearest-neighbors, Euclidean (L2)	none	5.0	LeCun et al. 1998
Boosted Stumps			
boosted stumps	none	7.7	Kegl et al., ICML 2009
Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998
SVMs			
SVM, Gaussian Kernel	none	1.4	
Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
Convolutional nets			
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	LeCun et al. 1998

2. http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

MNIST

who is the best in MNIST ?



MNIST 50 results collected

Units: error %

Classify handwritten digits. Some additional results are available on the original dataset page.

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details

1.3. MNIST DataSet 다운로드

Weka : arff type

<http://axon.cs.byu.edu/data/mnist/train.arff>

Index of /data/mnist

Name	Last modified	Size	Description
Parent Directory	-		
test.arff	2015-10-15 13:16	17M	
train.arff	2015-10-15 13:16	16M	

Apache/2.4.6 (Ubuntu)
Port 80

새 탭에서 링크 열기(T)
새 창에서 링크 열기(W)
시크릿 창에서 링크 열기(G)
다른 이름으로 링크 저장(K)...
링크 주소 복사(E)
검색(N) Ctrl+Shift+I

[illegible]

예제코드 내에 Data를 <http://yann.lecun.com/exdb/mnist/> 에서 다운받는 코드가 있음

```
def read_data_sets(train_dir,
                   fake_data=False,
                   one_hot=False,
                   dtype=dtypes.float32,
                   reshape=True,
                   validation_size=5000):
    if fake_data:
        def fake():
            return DataSet([], [], fake_data=True, one_hot=one_hot, dtype=dtype)
        train = fake()
        validation = fake()
        test = fake()
        return base.Datasets(train=train, validation=validation, test=test)

    TRAIN_IMAGES = 'train-images-idx3-ubyte.gz'
    TRAIN_LABELS = 'train-labels-idx1-ubyte.gz'
    TEST_IMAGES = 't10k-images-idx3-ubyte.gz'
    TEST_LABELS = 't10k-labels-idx1-ubyte.gz'

    local_file = base.maybe_download(TRAIN_IMAGES, train_dir,
                                      SOURCE_URL + TRAIN_IMAGES)
    with open(local_file, 'rb') as f:
        train_images = extract_images(f)

    local_file = base.maybe_download(TRAIN_LABELS, train_dir,
                                      SOURCE_URL + TRAIN_LABELS)
    with open(local_file, 'rb') as f:
        train_labels = extract_labels(f, one_hot=one_hot)

    local_file = base.maybe_download(TEST_IMAGES, train_dir,
                                      SOURCE_URL + TEST_IMAGES)
    with open(local_file, 'rb') as f:
        test_images = extract_images(f)

    with open(local_file, 'rb') as f:
        train_labels = extract_labels(f, one_hot=one_hot)

    if not 0 <= validation_size <= len(train_images):
        raise ValueError(
            'Validation size should be between 0 and {}'. Received: {}. '
            .format(len(train_images), validation_size))

    validation_images = train_images[:validation_size]
    validation_labels = train_labels[:validation_size]
    train_images = train_images[validation_size:]
    train_labels = train_labels[validation_size:]

    train = DataSet(train_images, train_labels, dtype=dtype, reshape=reshape)
    validation = DataSet(validation_images,
                          validation_labels,
                          dtype=dtype,
                          reshape=reshape)
    test = DataSet(test_images, test_labels, dtype=dtype, reshape=reshape)

    return base.Datasets(train=train, validation=validation, test=test)
```

THE MNIST DATABASE

of handwritten digits

[Yann LeCun](#), Courant Institute, NYU
[Corinna Cortes](#), Google Labs, New York
[Christopher J.C. Burges](#), Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

```

t10-train-images (10000x28x32 bytes)
t10-train-labels (10000x1 bytes)
t10-test-images (1000x28x32 bytes)
t10-test-labels (1000x1 bytes)

```

please note that your browser may decompress these files without telling you. If the files you downloaded have a larger size than the above, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. Some people have asked me "my application can't open my image file". These files are not in any standard image format. You have to write your own (very simple) program to read them. The file format is described at the bottom of this page.

The original black and white (dithered) images from NIST were size normalized to fit in a 28x28 pixel box while preserving their aspect ratio. The resulting images contain gray levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the digits, and translating the image so as to position this point at the center of the 28x28 field.

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000001 (2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

- Tensorflow 코드 내에서는 0-255 스케일의 value를 0-1 사이의 value로 normalize해서 사용
- 총 4가지의 파일로 이루어져있음

TRAINING SET LABEL FILE (train-labels-idx1-ubyte)
 TRAINING SET IMAGE FILE (train-images-idx3-ubyte)
 TEST SET LABEL FILE (t10k-labels-idx1-ubyte)
 TEST SET IMAGE FILE (t10k-images-idx3-ubyte)

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

2. Weka MNIST data 분석

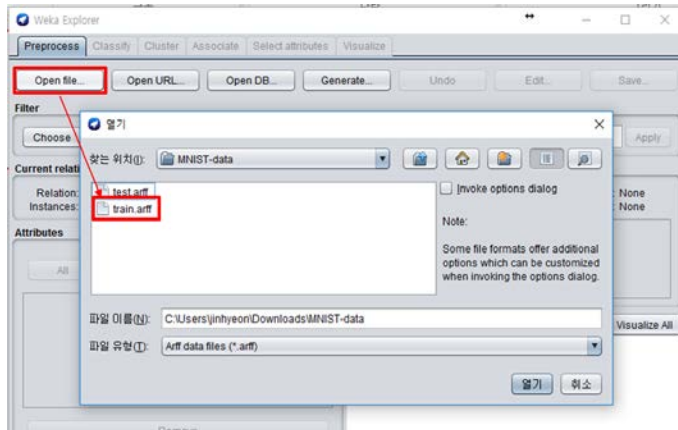
2.1. Intro

2.1.1. MNIST Dataset 불러오기

① Explorer 선택

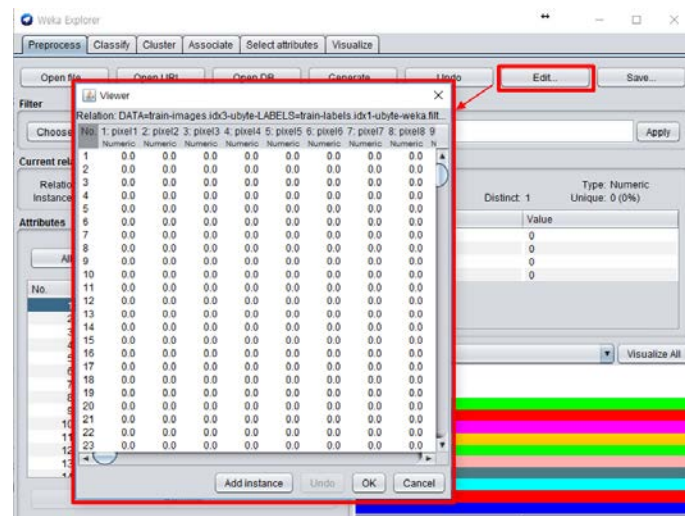


② Open file > 다운받은 폴더의 train.arff > 열기 선택



2.1.2. Data 확인

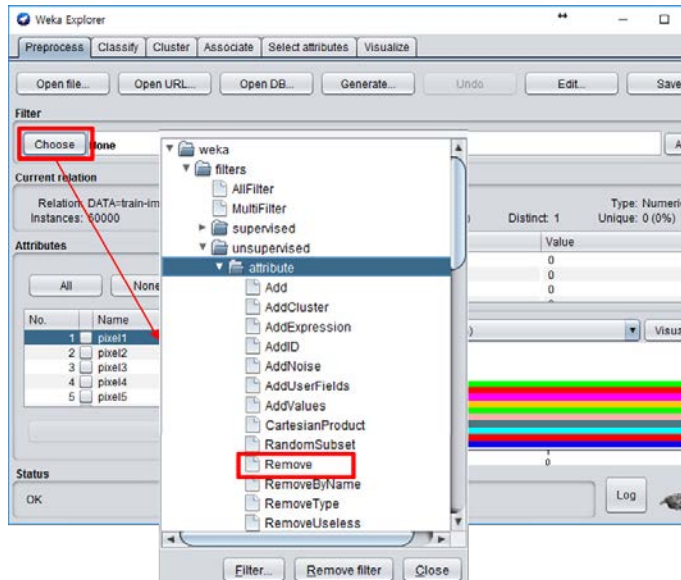
① Edit > Viewer



2.1.3. Data Handling

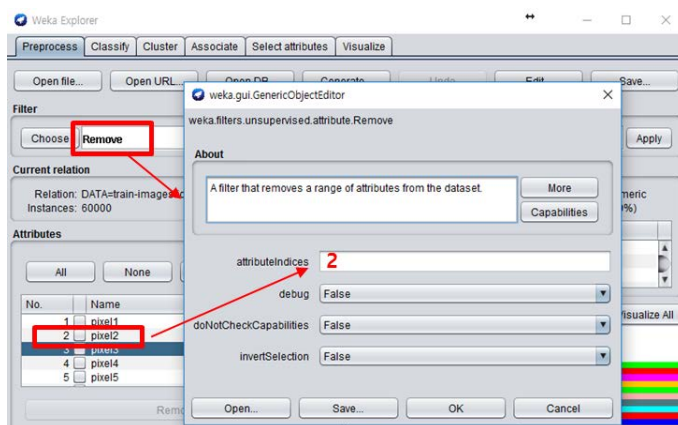
- Remove : attribute를 선택하여 삭제

- ① Filter > weka > filters > unsupervised > attribute



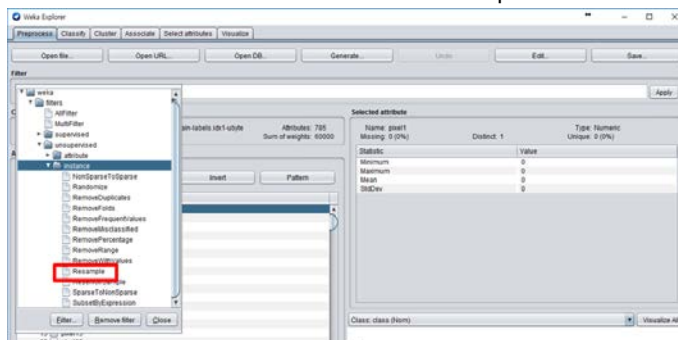
참고) Filter > supervised는 classifier를 선택하여 classifier 기준으로 data를 filtering할 수 있도록 하나, 여기서는 사용자가 원하는 특정 attribute의 no를 호출하여 삭제하는 것을 원하므로 unsupervised 메뉴에서 선택한다.

- ② Remove > attributeIndicate에 삭제하려는 attribute의 No. 입력 > OK > Apply

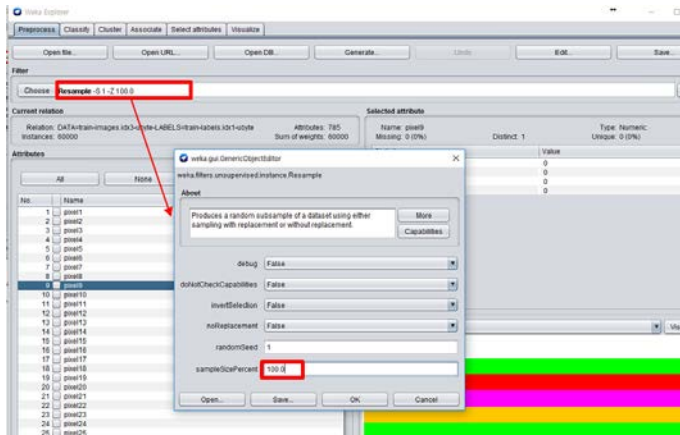


- Resampling

- ① Filter > Choose > Weka > filters > unsupervised > instance > Resample



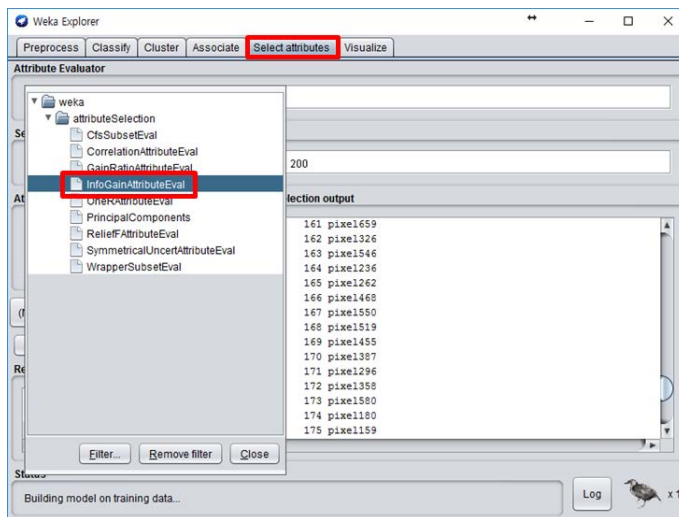
- ② Resample 클릭 > samplingSizePercent > OK > Apply



참고) samplingSizePercent에 전체 데이터 사이즈 중 사용할 데이터의 퍼센트를 입력한다.

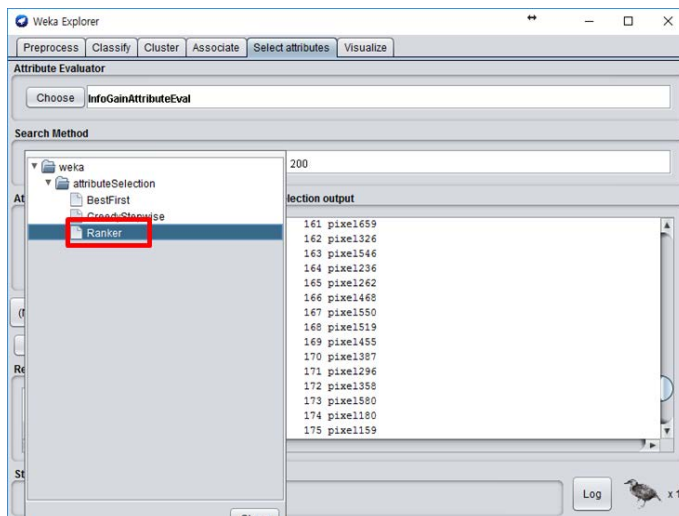
- Select Attribute

- ① Select attributes > Attribute Evaluator – choose > InfoGainAttributeEval



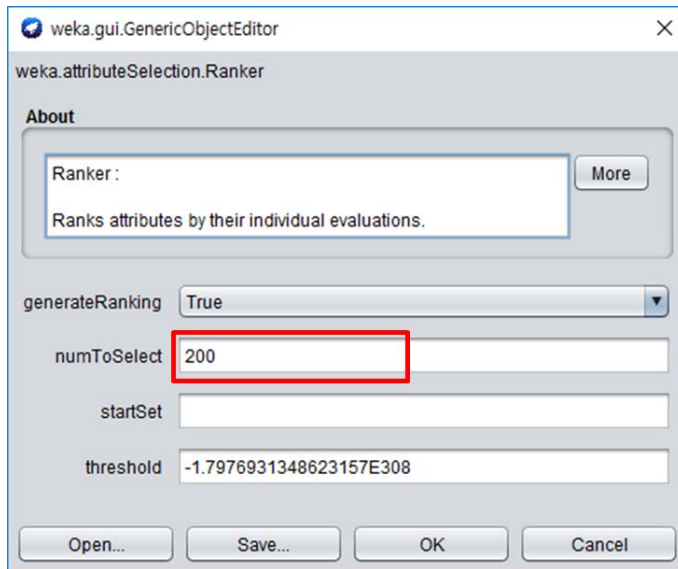
참고) InfoGain은 gain값을 기준으로 attribute를 선택하는 매서드이다.

- ② Search Method – choose > Ranker



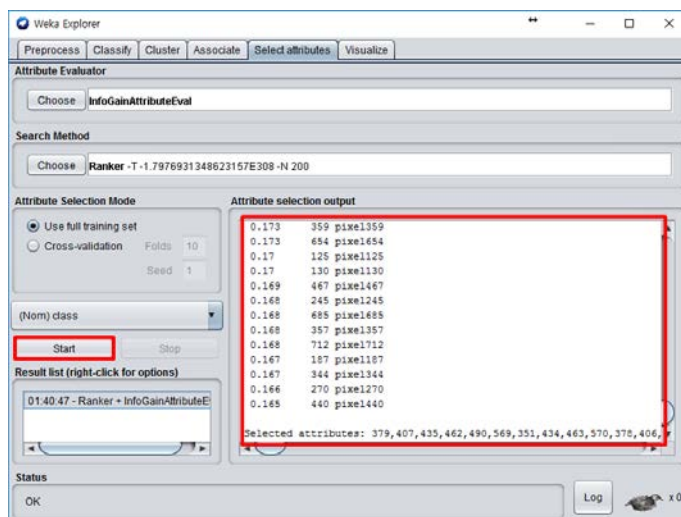
참고) InfoGain은 attribute를 rank하여 원하는 개수를 선택하는 매서드이다. Ranker를 사용한다.

③ Ranker > numToSelect



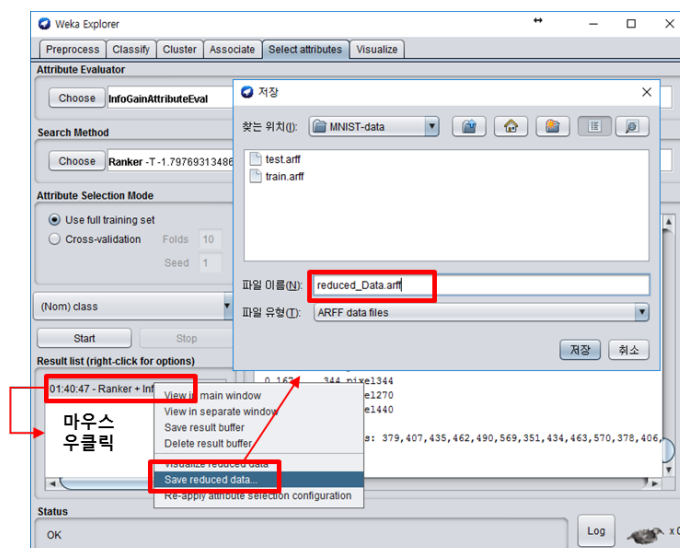
참고) Ranker의 numToSelect는 attribute를 gain값 순서대로 rank를 매겨서 선택할 attribute의 갯수이다.

④ Start > 값 확인

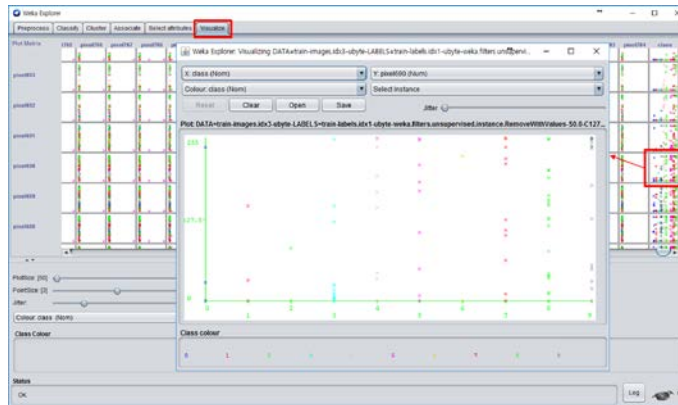


참고) gain 순서대로 rank되어 200개의 선택된 attribute들을 확인할 수 있다.

⑤ Result list > 결과 마우스 우클릭 > save reduced data > 이름 설정 후 저장



2.1.4. MNIST Dataset Visualize

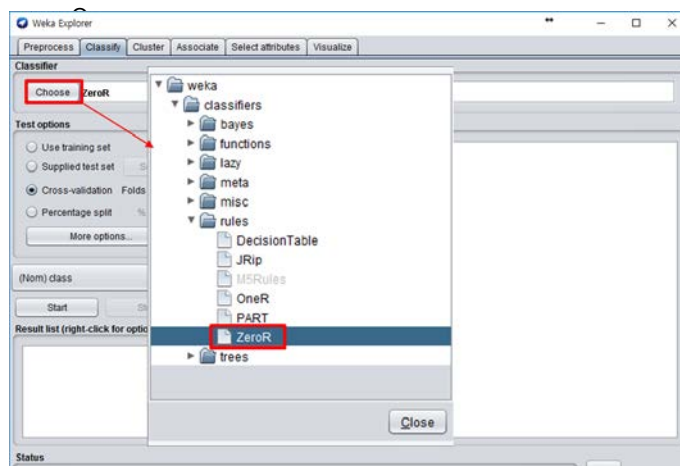


참고) 모든 attribute의 조합을 보여주며 각 칸은 두개의 attribute를 이용하여 2차원으로 보여주는 형태이다. Class별 색상이 정해져 있다.

2.1.5. Simple classifier : Zero R, J48

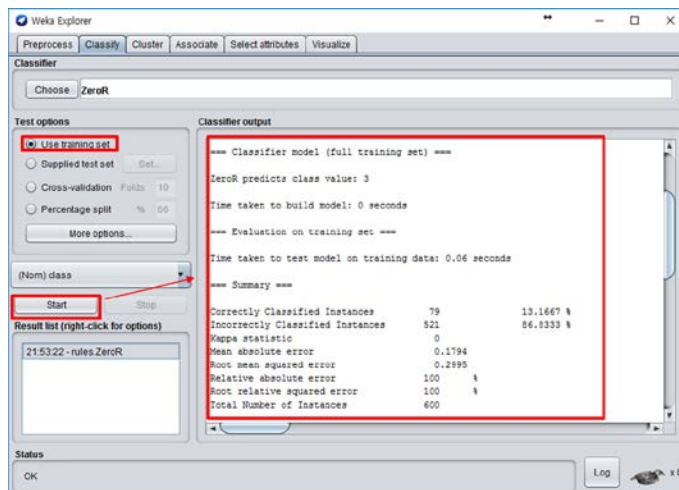
- Zero R : 모든 값을 하나의 클래스로 예측

① Classify > Choose > weka > classifiers > rules > ZeroR



참고) Zero R은 Baseline으로 쓰이는 알고리즘이다. 모든 attribute를 class중 가장 많은 class로 예측한다.

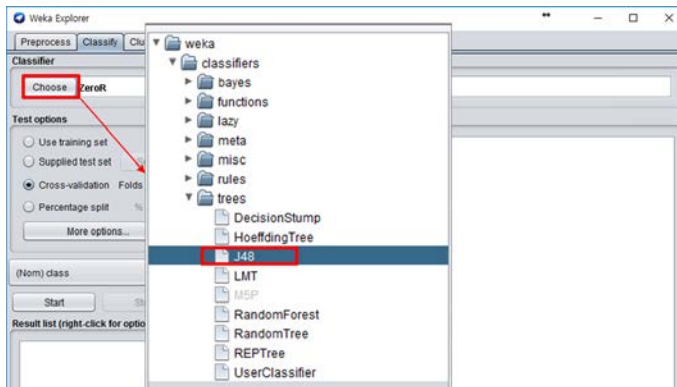
② Use training Set > Start



참고) Zero R classifier는 MNIST데이터의 attribute들을 모두 3으로 분류되는 class로 예측하였다. 0~9까지의 class 중 13%가 3이기 때문에 13%의 데이터를 옳게 분류하였다.

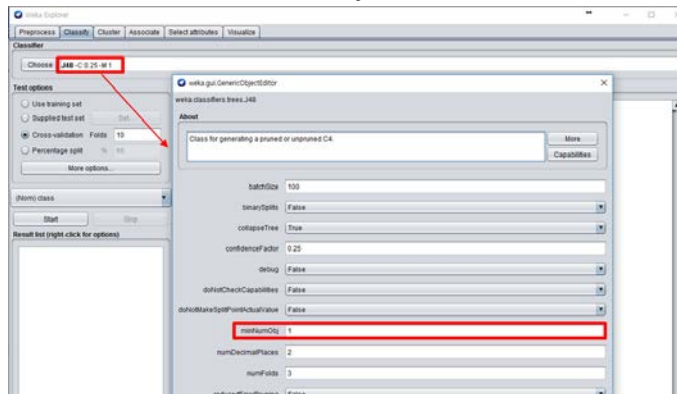
- J48 : decision tree algorithm.

① Classify > Choose > weka > classifiers > trees > J48



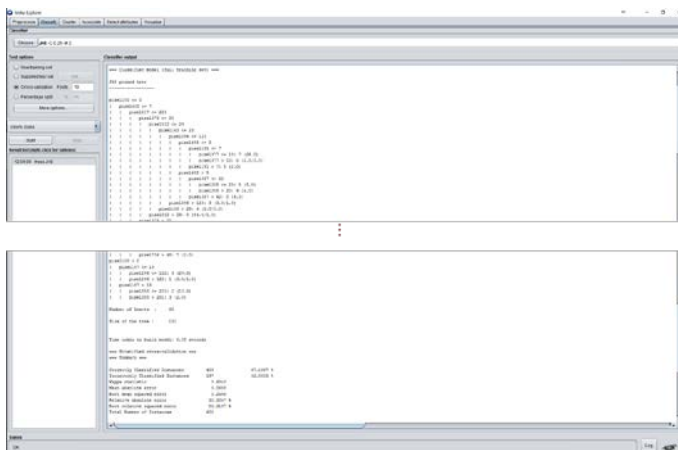
참고) Ross Quinla이 ID3를 기반으로 구현한 C45 알고리즘의 개정판 C48을 weka에서 java로 구현하여 J48이라는 이름을 붙였다. continuous하거나 discrete한 attribute를 처리할 수 있고, missing value를 처리하여 모델을 학습시킬 수 있으며 각 attribute에 다른 cost를 줄 수 있고 pruning도 가능하다.

② (선택사항) J48 > minNumObj의 숫자를 정한다.



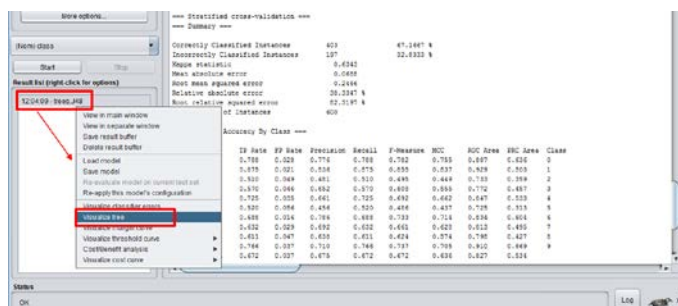
참고) minNumObj값도 다양한 조건을 추가할 수 있다. Tree를 Prun하기 위한 값 조정도 가능하다.

③ Cross-validation > start



참고) 모델의 generalization을 위하여 Cross validation을 시행하였다. Decision Tree는 67.1667 %의 정확도를 보인다. 위의 그래프는 트리를 text로 시각화 한 것이다.

④ 결과 목록에서 우클릭 > Visualize Tree



⑤ Tree 확인하기



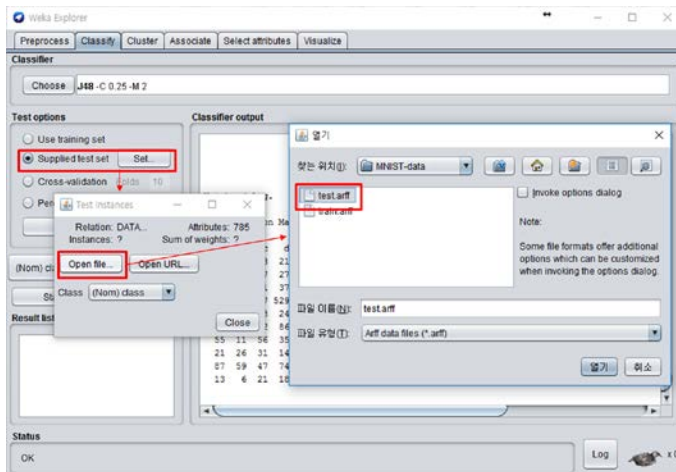
참고) tree를 visualize했을 시, 볼 수 없을 만큼 작게 나오는 경우가 있다. 윈도우의 크기를 키운 후, 오른쪽 버튼 클릭하여 fit to Screen을 클릭한다.

원형의 노드는 기준을 제시하고 연결선은 참인지 거짓인지 판별한다. 사각형의 노드는 분류된 값이다.

2.2. Evaluation

2.2.1. Test set

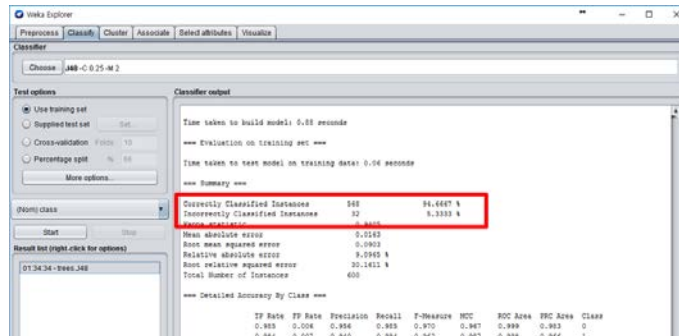
- ① Training set은 앞에서 실행한 Preprocess 탭에서 데이터 파일을 여는 형식으로 불러온다. 우리는 MNIST training set을 불러와 resampling 해 두었다.
- ② Test set은 아래와 같이 설정한다.



참고) MNIST는 test set이 이미 주어져 있으므로, 사용할 수 있지만 다른 데이터 set은 없는 경우가 있다. 그런 경우, Cross-validation이나 training data를 split하여 Evaluation한다.

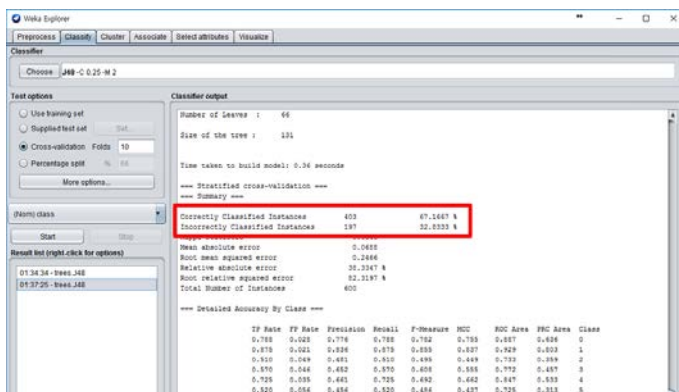
2.2.2. Cross-validation

-Cross-validation을 사용하지 않은 모델의 데이터 분류 정도



참고) training set과 test set의 데이터를 같게 놓고 모델을 테스트 하였다. 옳게 분류한 값이 많으나, 이것은 general 한 모델이 아님을 유추할 수 있다.

-Cross-validation을 사용한 모델의 데이터 분류 정도

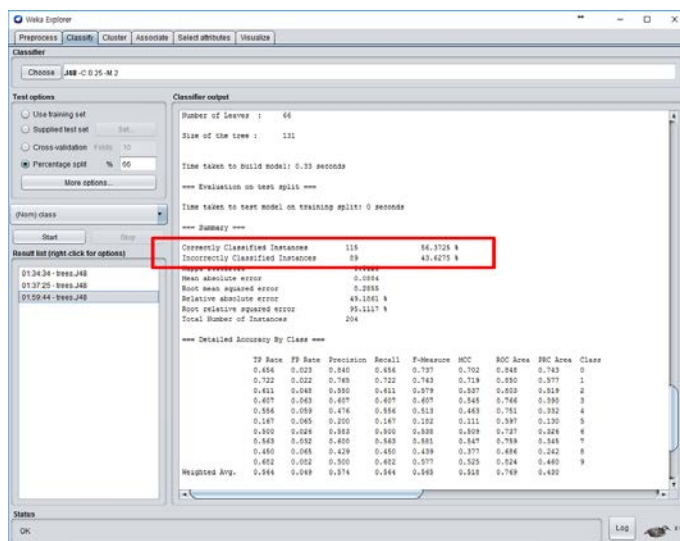


참고) test option을 cross-validation으로 지정하였다.

10은 10 fold cross validation을 의미한다. 비교적 general 한 모델이 학습되었음을 알 수 있다.

2.2.3. Split

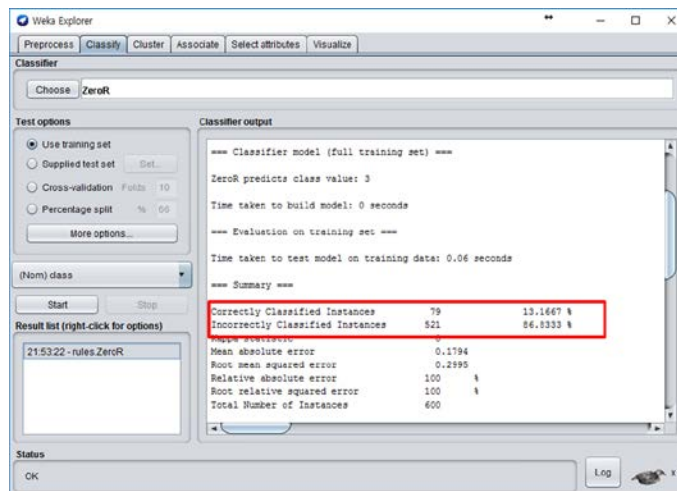
① Test option의 split 선택



참고) 66%로 나누어 66%는 training set, 나머지 44%는 test set으로 사용한다.

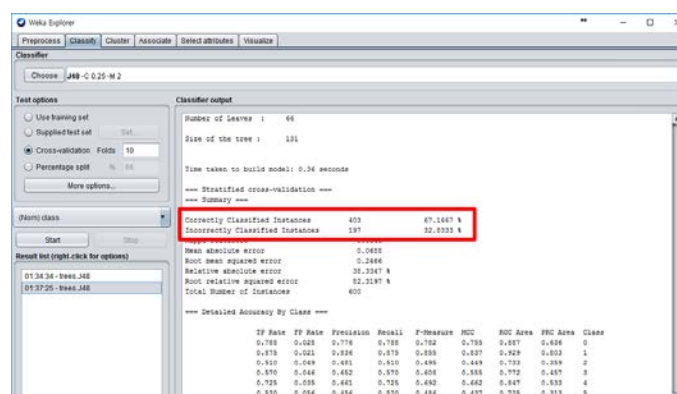
2.2.4. Baseline과 다양한 알고리즘 비교 : ZERO R, J48, naïve bayes, IBk

- Zero R

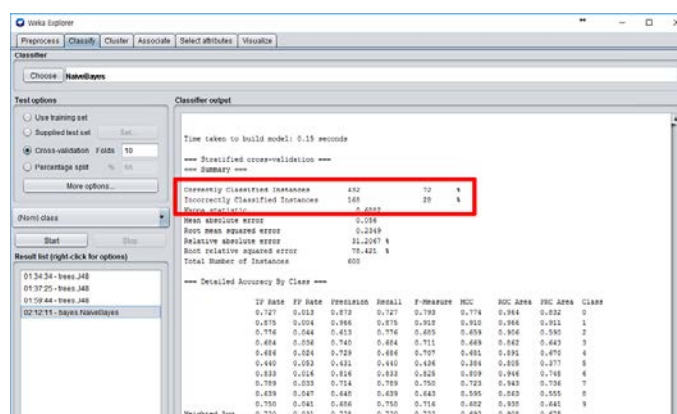


참고) Zero R은 Baseline으로 사용되기 때문에 test option 역시 use training set으로 설정한다.

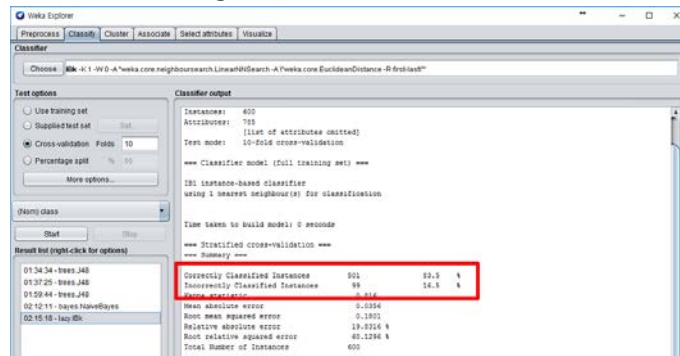
- J48



- Naive bayes

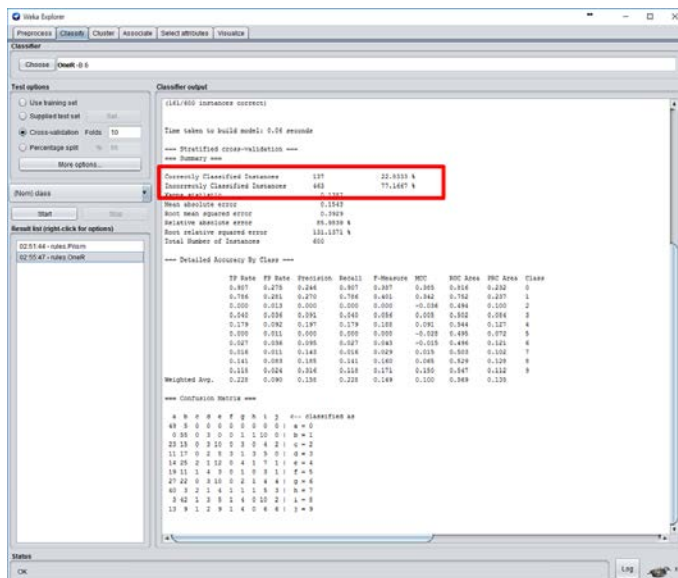


- IBk(K-nearest neighbor classifier)



2.3. Rule Based Classifier

2.3.1. One R

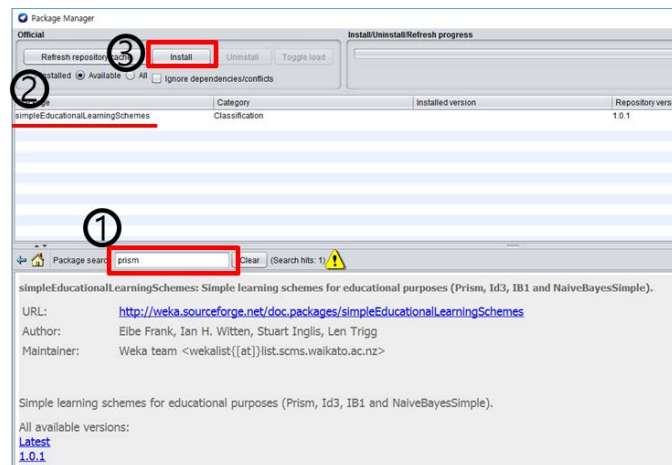


참고) 에러율 확인해서 가장 에러율이 작은 하나의 attribute를 골라서 class를 나누는 classifier이다.

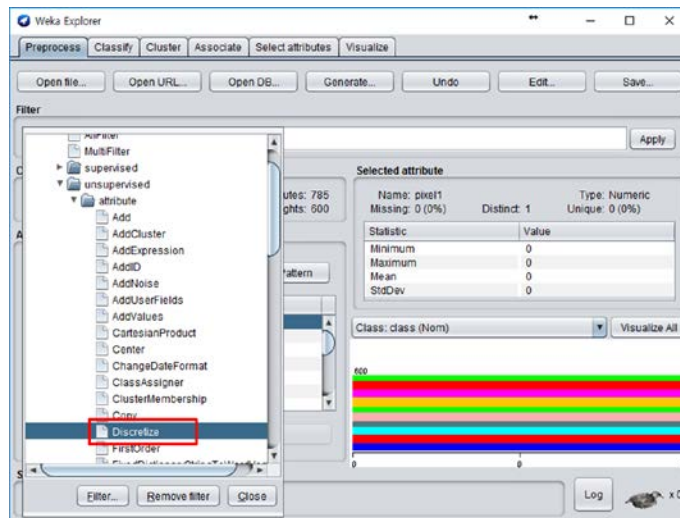
아주 간단한 데이터셋이나 noise가 많이 포함된 데이터셋, 데이터에서 학습할 것이 없을 때 사용한다

2.3.2. PRISM

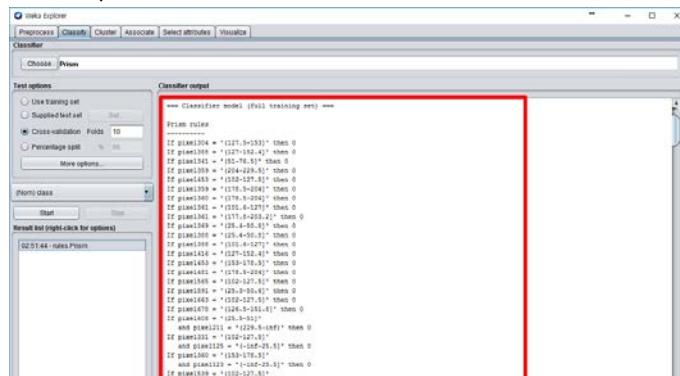
- ① 위에서 언급한 unofficial package에 속한다. Package manager에서 아래의 순서로 다운받는다.



- ② Prism은 연속적인 숫자 data는 처리할 수 없다. Resample 뿐만 아니라, discretize도 실행한다.



- ③ Rule > prism

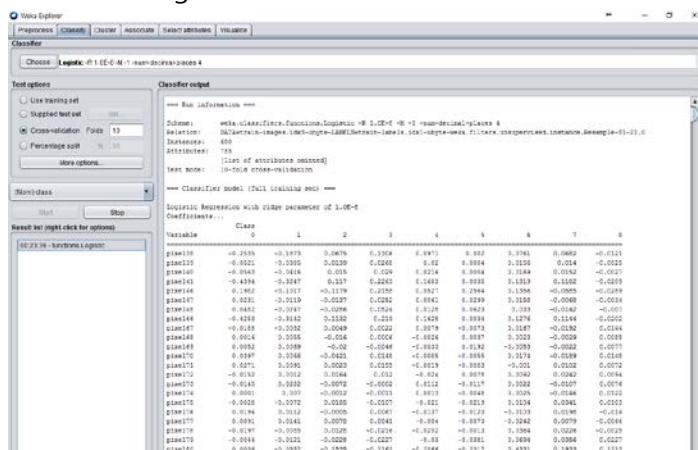


참고) overfitting이 일어나기 쉬운 구조이다.

2.4. Linear Classifier

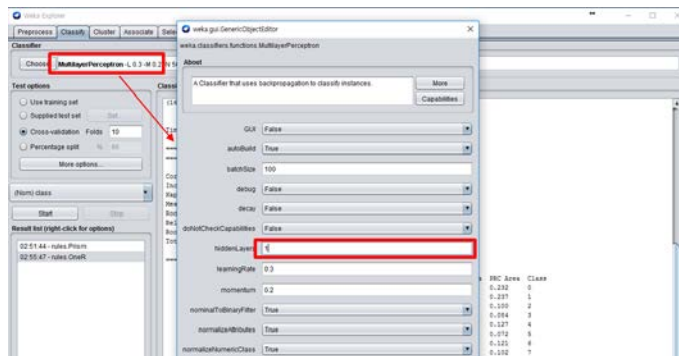
2.4.1. Logistic

- ① function > Logistic



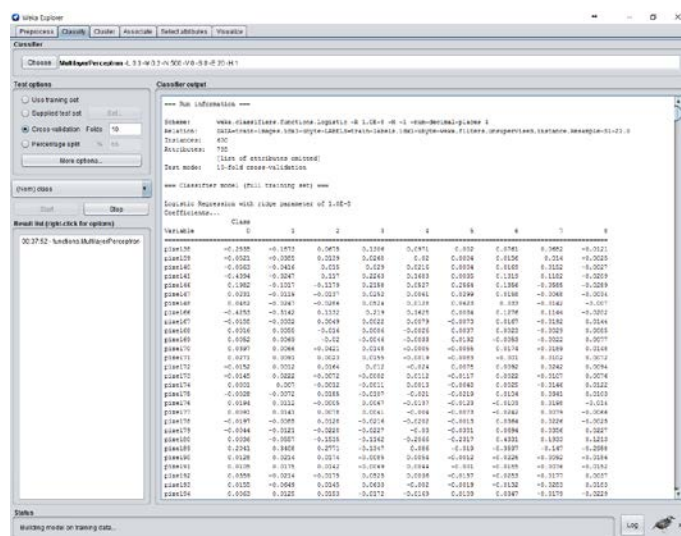
2.4.2. Perceptron

① Function > multilayerPerceptron




참고) Multi-Layer Perceptron의 hidden layer 개수를 조정하여 perceptron을 만든다. 1은 single perceptron이고, a 는 hidden layer를 auto로 설정하는 것이다. 상당히 긴 시간이 걸린다.

② 상당히 오랜 시간이 걸려서 결과가 출력되는 것을 확인할 수 있다.



3. TensorFlow MNIST data분석

Jupyter Notebook 접속 및 예제코드 실행

jupyter

FilesRunningClustersConda

Select items to perform actions on them.

UploadNew↺

☐

TensorFlow / iPythonNotebook

☐

..

☐

assets

☐

Basics(Docker example)

☐

Convolutioanl Neural Network

☐

MNIST_data

☐

Neural Network


☐

test


☐

train


☐

 CNN_1.ipynbRunning


☐

 CNN_2_advanced.ipynb


☐

 RNN_1.ipynb


☐

 RNN_2.ipynb

☐

 Tensorflow_basic.ipynb

☐

 Word2vec_Gensim_Tensorboard.ipynb

3.1. Neural Network (NN)

코드 설명 Part1

```
In [1]: # -*- coding: utf-8 -*-
from tensorflow.examples.tutorials.mnist import input_data  # MNIST 데이터를 다운받는 코드
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True) # Yann LeCun's website에서 MNIST 다운

# MNIST 객체로부터 데이터를 얻어온다.
x_train = mnist.train.images
y_train = mnist.train.labels
x_test = mnist.test.images
y_test = mnist.test.labels

print("x_train Shape: ", x_train.shape)
print("y_train Shape: ", y_train.shape)
print("x_test Shape: ", x_test.shape)
print("y_test Shape: ", y_test.shape)
```

코드 설명 part2

```
In [2]: import tensorflow as tf

##### Computational Graph 설정 시작 #####
x = tf.placeholder(tf.float32, [None, 784]) # 데이터 값을 placeholder 선언
y_ = tf.placeholder(tf.float32, [None, 10]) # True Label 값

W = tf.Variable(tf.zeros([784, 10])) # 학습할 Weight Matrix
b = tf.Variable(tf.zeros([10])) # 학습할 bias

y = tf.nn.softmax(tf.matmul(x, W) + b) # 예측 Label 값

cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1])) # Loss

train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

##### Computational Graph 설정 끝 #####
```

Computational Graph 정의

코드 설명 part3

```
In [3]: init = tf.global_variables_initializer() # 변수 초기화 (텐서플로우 필수과정)
sess = tf.Session() # 세션 열기
sess.run(init) # 초기화

##### Computational Graph를 실행시키기 위한 Session 선언 및 Variable 초기화 #####

In [4]: for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100) # train, validation, test 에 데이터가 들어가 있을 # return s
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

##### 테스트를 위한 Computational Graph 설정 시작 ##### (꼭 이 자리가 아니고 위에 선언되도 괜찮음)
prediction = tf.argmax(y, 1)
target = tf.argmax(y_, 1)

correct_prediction = tf.equal(prediction, target) # argmax(input, dimension of the input Tensor to reduce across)
incorrect_prediction = tf.not_equal(prediction, target)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
##### 테스트를 위한 Computational Graph 설정 끝 #####

print(mnist.test.labels) # Label data가 어떻게 생겼는지 확인 (One-hot representation)
acc, pred, incorrect_pred = sess.run([accuracy, prediction, incorrect_prediction], feed_dict={x: mnist.test.images, y_: mnist.test.labels})
print("accuracy:")
print(acc)
```

mini batch로 Training (SGD)

성능 평가를 위한 Computational Graph 정의

성능 평가를 위한 Computational graph 실행 (Run within a Session)

3.2. Convolutional Neural Network (CNN)

코드 설명 Part1

```
In [3]: # Tensorflow 모듈을 import한다.
import tensorflow as tf

In [4]: # Input data를 담을 변수인 Placeholder를 선언한다.
with tf.name_scope('input'):
    x = tf.placeholder(tf.float32, shape=[None, 784])
    y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

Input data를 담을 Placeholder 선언

```
In [5]: # Weight를 초기화 한다.
def weight_variable(shape, name):
    initial = tf.truncated_normal(shape, stddev=0.1) # weights_initializer=tf.contrib.layers.xavier_initializer()
    return tf.Variable(initial, name=name)

def bias_variable(shape, name):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial, name=name)

# Convolution과 Pooling 을 정의한다.
# Padding이 'SAME'이라는 것은 Zero padding(여백)을 허용한다는 뜻이다.
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x, name):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1], padding='SAME', name=name)
```

편하게 변수를 선언하기 위한 함수

편하게 Convolution layer와 max pooling을 하기 위한 함수

코드 설명 part2

```
In [7]: # Convolution을 하기 위해 Input에 대해서 Reshape합니다.
with tf.name_scope('input_reshape'):
    x_image = tf.reshape(x, [-1, 28, 28, 1], name="x_image_reshape") # (Batch_size, width, height, Dim of Channel)
    tf.summary.image('input', x_image, 10)
```

CNN에 적합한 tensor 형태로 reshape

```
In [8]: # 첫번째 Convolutional layer의 Weight와 Bias
with tf.name_scope("Conv_Layer1"):
    W_conv1 = weight_variable([5, 5, 1, 32], name="W_conv1") # (가로길이, 세로길이, 채널 차원수, 필터 개수)
    b_conv1 = bias_variable([32], name="b_conv1")
    variable_summaries(W_conv1, 'W_conv1')
    variable_summaries(b_conv1, 'b_conv1')

    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1, name="h_conv1")
    variable_summaries(h_conv1, 'h_conv1')

# 첫번째 Convolutional layer와 Max pooling layer
with tf.name_scope("Pooling_Layer1"):
    h_pool1 = max_pool_2x2(h_conv1, name="h_pool1")
    variable_summaries(h_pool1, 'h_pool1')
```

첫번째와 두번째의 Conv layer와 max pooling layer의 Computational Graph

```
In [9]: # 두번째 Convolutional Layer의 Weight와 Bias
with tf.name_scope("Conv_Layer2"):
    W_conv2 = weight_variable([5, 5, 32, 64], name="W_conv2")
    b_conv2 = bias_variable([64], name="b_conv2")
    variable_summaries(W_conv2, 'W_conv2')
    variable_summaries(b_conv2, 'b_conv2')

    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2, name="h_conv2")
    variable_summaries(h_conv2, 'h_conv2')

# 두번째 Convolutional layer와 Max pooling layer
with tf.name_scope("Pooling_Layer2"):
    h_pool2 = max_pool_2x2(h_conv2, name="h_pool2")
    variable_summaries(h_pool2, 'h_pool2')
```

코드 설명 part3

```
In [10]: # 첫번째 FC Layer (Fully Connected Layer)          첫번째 FC Layer의 Computational Graph 정의
with tf.name_scope("FC_Layer1"):
    W_fc1 = weight_variable([7 * 7 * 64, 1024], name="W_fc1") # 현재까지의 이미지 크기는 7x7 (Pooling 2번!)
    b_fc1 = bias_variable([1024], name="b_fc1")
    variable_summaries(W_fc1, 'W_fc1')
    variable_summaries(b_fc1, 'b_fc1')
    h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64], name="h_pool2_flat")
    h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1, name="h_fc1")

In [11]: # Dropout을 FC Layer에 적용한다.                  첫번째 FC Layer의 결과값에 대한 Dropout의 Computational Graph 정의
with tf.name_scope('dropout'):
    keep_prob = tf.placeholder(tf.float32, name="keep_prob")
    h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob, name="h_fc1_drop")
    variable_summaries(keep_prob, 'dropout_keep_probability')
    variable_summaries(h_fc1_drop, 'h_fc1_drop')

In [12]: # 두번째 FC Layer                                두번째 FC Layer의 Computational Graph 정의
with tf.name_scope("FC_Layer2"):
    W_fc2 = weight_variable([1024, 10], name="W_fc2")
    b_fc2 = bias_variable([10], name="b_fc2")
    variable_summaries(W_fc2, 'W_fc2')
    variable_summaries(b_fc2, 'b_fc2')

    y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
    tf.summary.histogram('y_conv', y_conv)
```

코드 설명 part4

```
In [13]: # Loss(=cross entropy)를 정의한다                Training을 위한 Computational Graph 정의
with tf.name_scope("cross_entropy"):
    cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_conv, labels=y_), name="cross_entropy")
    tf.summary.scalar('cross_entropy', cross_entropy)

# Training operation을 정의한다.
# 본 예제에서는 AdamOptimizer를 사용한다.
with tf.name_scope('train'):
    learning_rate = 1e-4
    train_step = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropy)

with tf.name_scope('prediction'):                          Evaluation을 위한 Computational Graph 정의
    prediction = tf.argmax(y_conv, 1)
with tf.name_scope('target'):
    target = tf.argmax(y_, 1)

with tf.name_scope('correct_prediction'):
    correct_prediction = tf.equal(prediction, target, name="correct_prediction") # 라벨과 예측값이 맞으면 True
with tf.name_scope('incorrect_prediction'):
    incorrect_prediction = tf.not_equal(prediction, target, name="incorrect_prediction")

# Accuracy를 정의한다.
with tf.name_scope('accuracy'):
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32), name="accuracy")
    tf.summary.scalar('accuracy', accuracy)
```


코드 설명 part5

In [14]: `merged = tf.summary.merge_all()` TensorBoard를 위한 Summary merge

In [15]: `# Tensorflow에서 정의한 계산 과정(Computation Graph)을 실행(Operation)하기 위한 세션을 선언한다.
sess = tf.InteractiveSession()
Tensorflow에서 모든 변수는 초기화해야 합니다. 초기화 연산을 정의한다.
init = tf.global_variables_initializer()
sess.run() 통해 실제 실행(Run)을 수행한다.
sess.run(init)` Computational Graph를 실행하기 위한 세션 선언

In [16]: `train_writer = tf.summary.FileWriter('./train', sess.graph)
test_writer = tf.summary.FileWriter('./test')` TensorBoard를 위해 summary 데이터를 저장하기 위한 FileWriter 선언

In [19]: `for i in range(200):
 batch = mnist.train.next_batch(50)
 if i%100 == 0:
 train_accuracy = accuracy.eval(feed_dict={
 x:batch[0], y_: batch[1], keep_prob: 1.0})
 print("step %d, training accuracy %g"%(i, train_accuracy))

 summary, _ = sess.run([merged, train_step], feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
 train_writer.add_summary(summary, i)` Session에 Training에 대한 Computational Graph를 넣고 학습 진행

step 0, training accuracy 0.56
step 100, training accuracy 0.92

Session에 Evaluation에 대한 Computational Graph를 넣고 실행

In [20]: `acc, pred, incorrect_pred = sess.run([accuracy, prediction, incorrect_prediction], feed_dict={x: mnist.test.images,
print("test accuracy: ", acc)

print("Incorrect Prediction Case")
plot_mnist(mnist.test.images, classes=pred, incorrect=incorrect_pred)`

Tensorboard 가이드

DeepLearningTutorials/ii x main_NN_TF_Tensorboar x 192.168.99.100:8888/terr x

192.168.99.100:8888/terminals/1

jupyter

```
# cd iPythonNotebook
# ls
CNN NN
# cd NN
# tensorboard --logdir=./train
Starting tensorBoard b'28' on port 6006
(You can navigate to http://172.17.0.2:6006)
```

TensorBoard SCALARS IMAGES AUDIO GRAPHS DISTRIBUTIONS HISTOGRAMS EMBEDDINGS

Write a regex to create a tag group X

☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

Smoothing

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

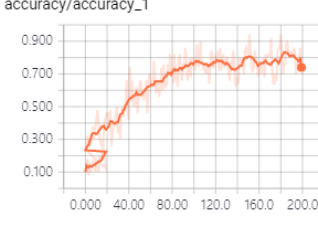
☒ ☐ ☐

TOGGLE ALL RUNS

./train

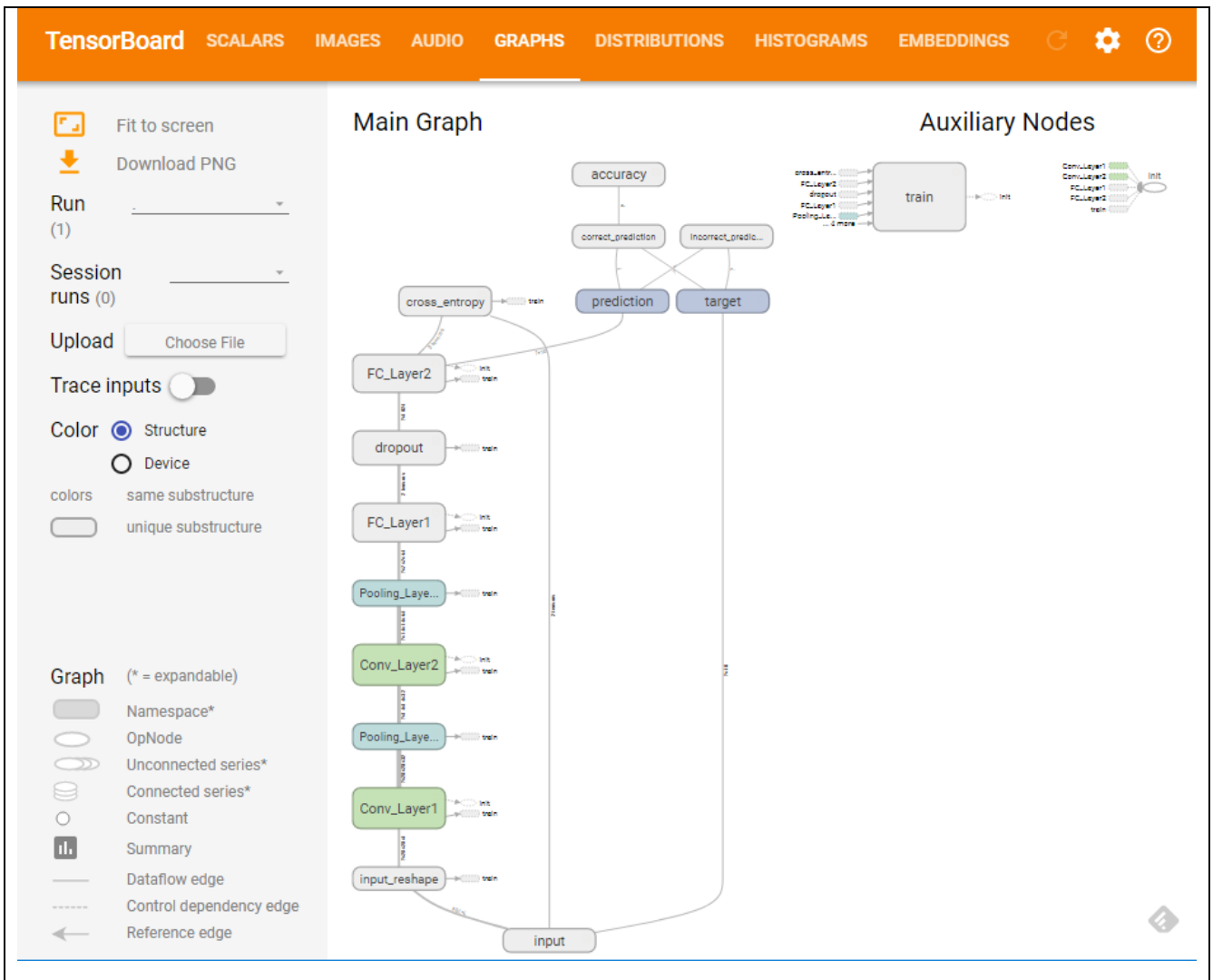
Conv_Layer1	12
Conv_Layer2	12
FC_Layer1	8
FC_Layer2	8
Pooling_Layer1	4
Pooling_Layer2	4
accuracy	1
cross_entropy	1
dropout	8

accuracy/accuracy_1



0.000 40.00 80.00 120.0 160.0 200.0

0.100 0.300 0.500 0.700 0.900



실행을 위한 포트 확인(Docker 실행시 포트포워딩 지정 안한 경우)

- 1.docker ps -a 로 컨테이너 리스트 확인
- 2.docker restart 아이디
3. 다시 docker ps -a 하면 포트포워딩이 어떻게 되어있는지 확인가능

(원래 노트북은 8888포트, 텐서보드는 6006포트를 씀)

```
$ docker ps -a
CONTAINER ID   IMAGE                                NAMES      COMMAND                  CREATED        STATUS
7b5975a2afec   orroneousboat/tensorflow-pyhton3-jupyter:latest   "/run_jupyter.sh"  20 minutes ago   Up 42 seconds
0.0.0.0:32777->6006/tcp, 0.0.0.0:32778->8888/tcp   tensorflow-pyhton3-jupyter
8a878abe493b   gcr.io/tensorflow/tensorflow          "/run_jupyter.sh"  54 minutes ago   Exited (137) 24 minutes ago
af89bd468b39   imcomking/ttskc_cpu: latest           "/bin/bash"        5 weeks ago     Exited (0) 40 minutes ago
ttskc_cpu
```

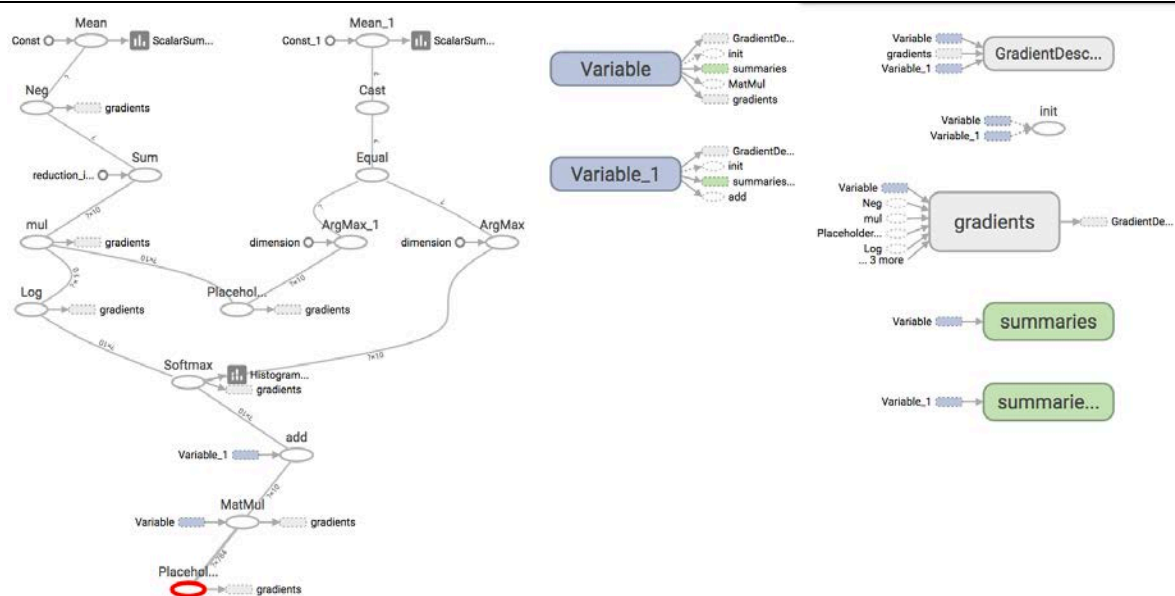
TensorBoard 예시

TensorFlow에서는

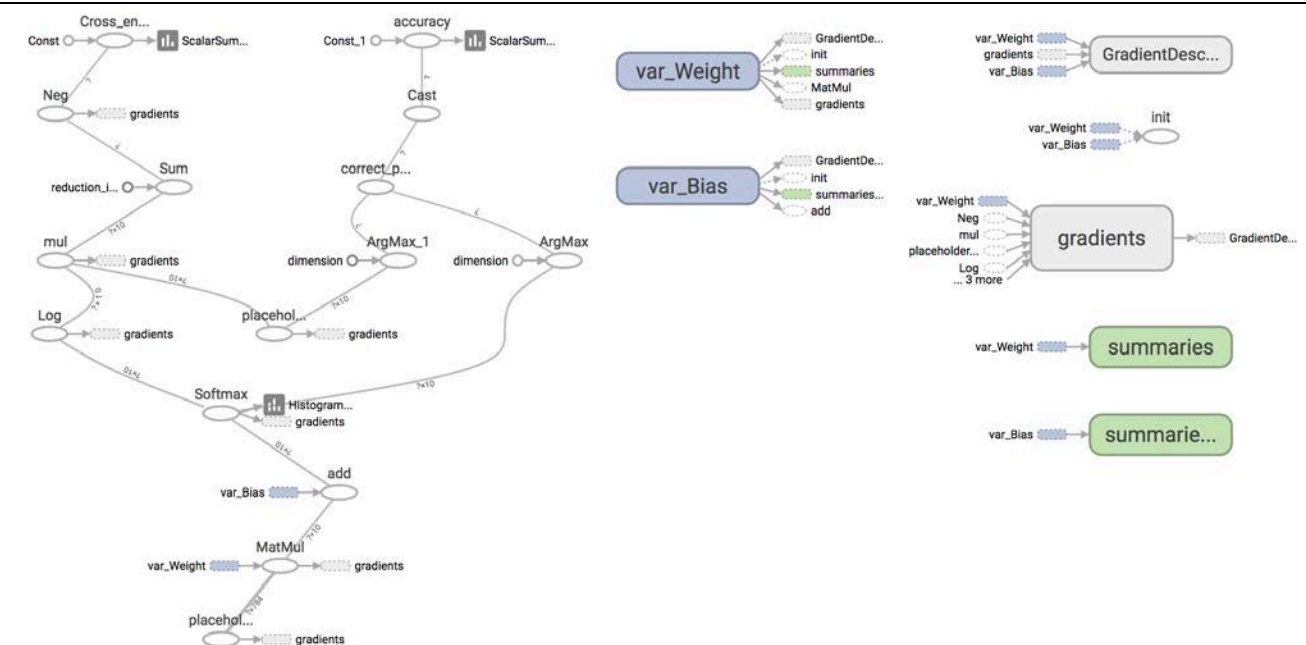
1. 각 Tensor에 Name을 할당하는 방법과
2. 각 구간에 Name Scope를 할당하는 방법이 있다.

TensorBoard 에서 디버깅할 때 매우 유용하니 습관적으로 지정해두는 것이 좋다.

Neural Network) Tensor에 Name 지정 전



Neural Network) Tensor에 Name 지정 후



```

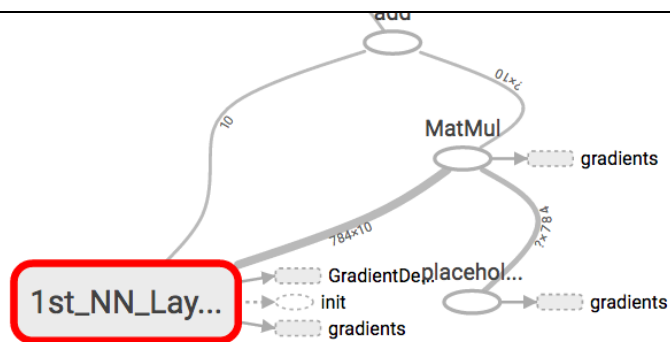
x = tf.placeholder(tf.float32, [None, 784], name="placeholder_mnist") # 데이터 담을 placeholder 선언

W = tf.Variable(tf.zeros([784, 10]), name="var_Weight") # 학습할 Weight Matrix
b = tf.Variable(tf.zeros([10]), name="var_Bias") # 학습할 bias
variable_summaries(W, 'weights')
variable_summaries(b, 'biases')

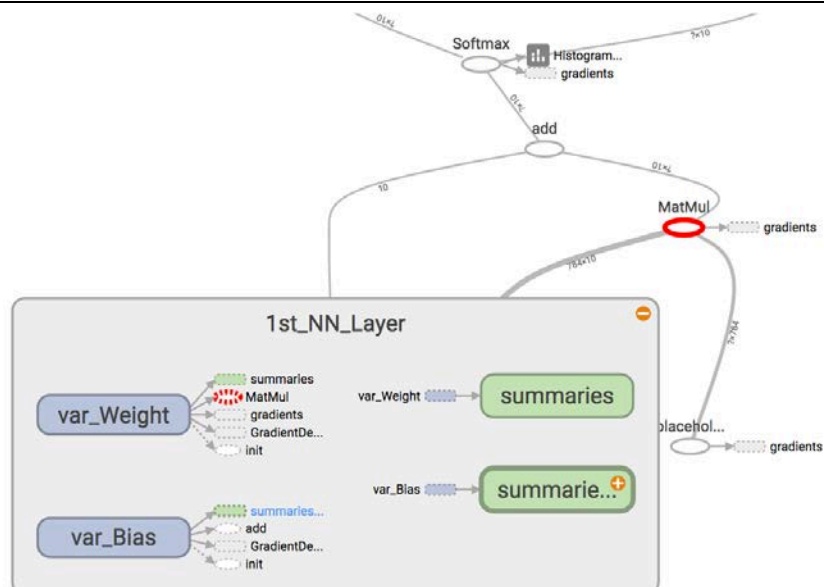
y = tf.nn.softmax(tf.matmul(x, W) + b) # 예측 Label 값
tf.histogram_summary('pre_activations', y)
y_ = tf.placeholder(tf.float32, [None, 10], name="placeholder_TrueLabel") # True Label 값

```

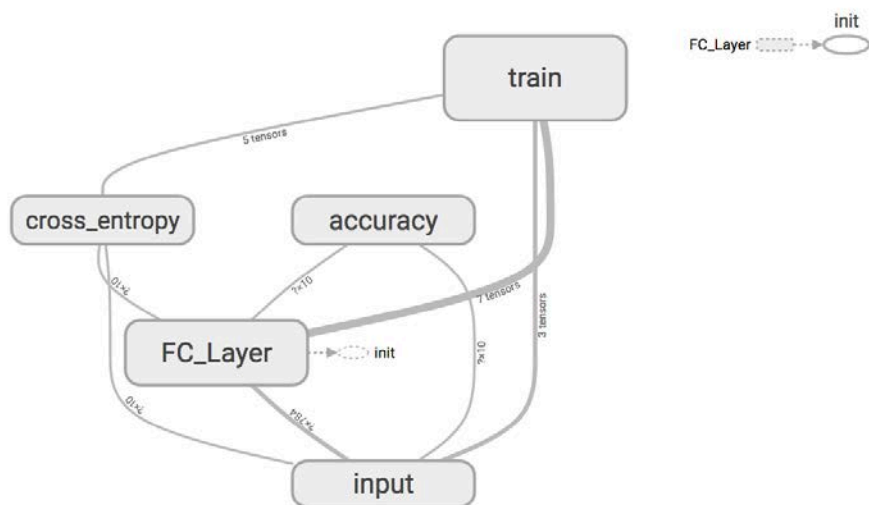
Neural Network) Name Scope 지정 후 1



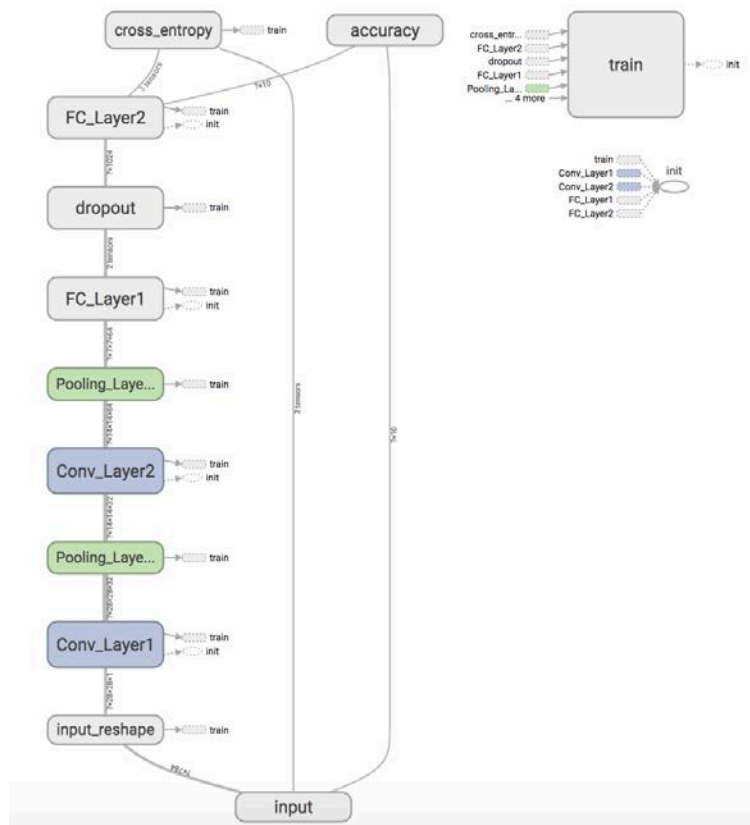
Neural Network) Name Scope 지정 후 2



Neural Network) Name Scope 전체 지정

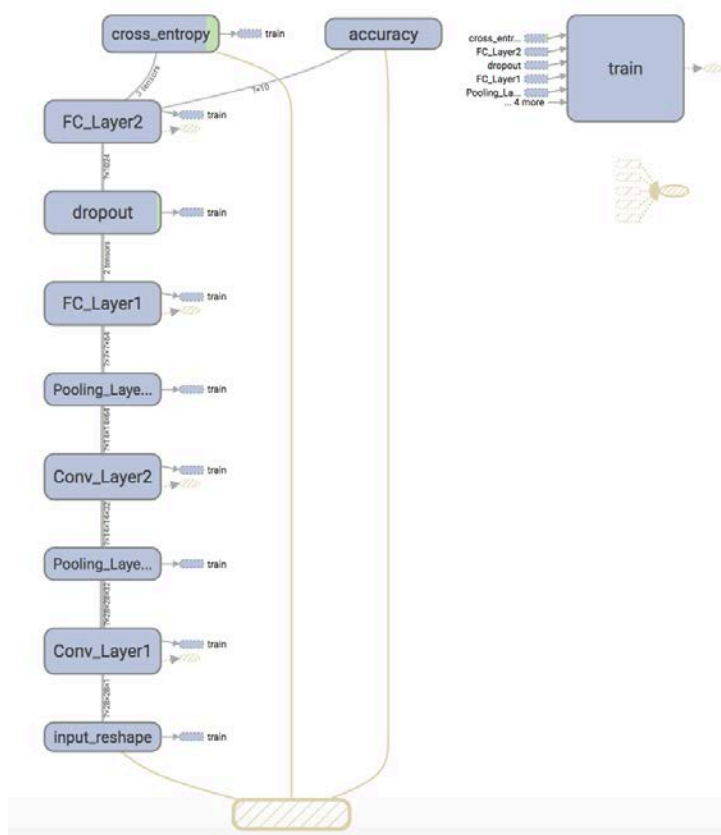


Convolutional Neural Network) Name Scope

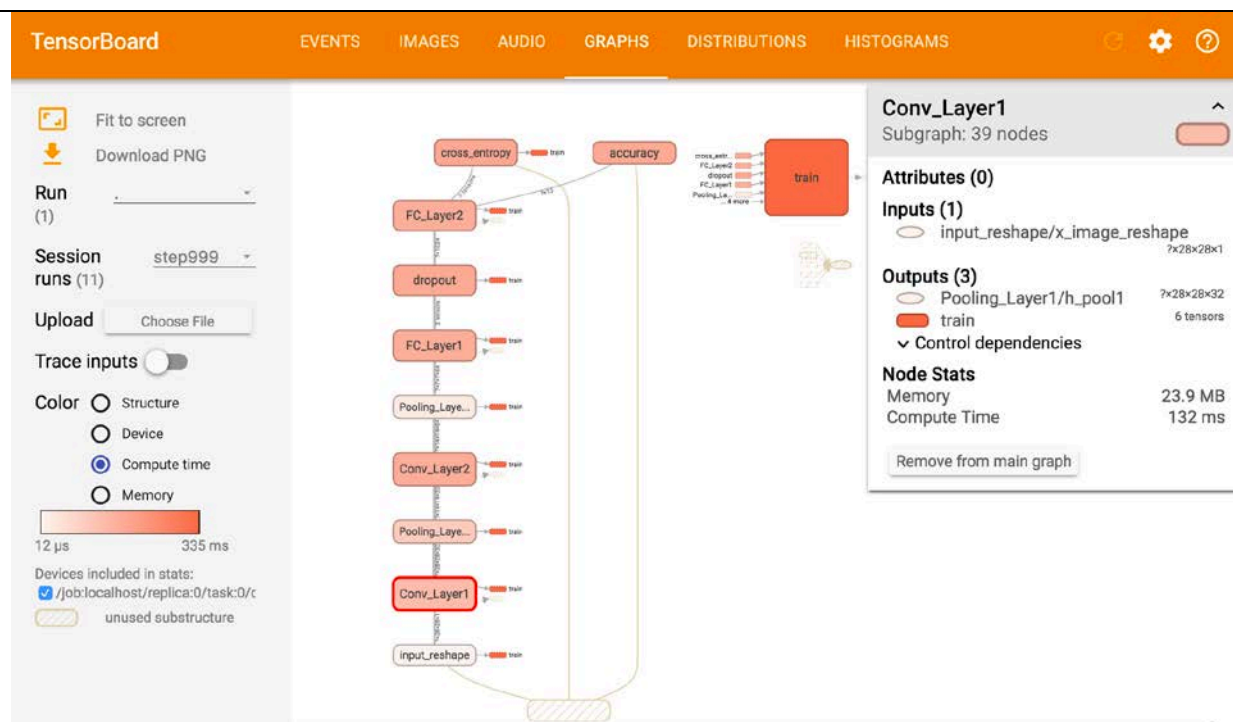


TensorBoard에서는 Run Statistics(e.g CPU/GPU 사용량) 정보를 제공합니다.

Convolutional Neural Network) CPU(Green) vs GPU(Blue)



Convolutional Neural Network) Runtime Statistics



부록

4. 실습환경 설정

4.1. Weka

windows / mac 설치

<http://www.cs.waikato.ac.nz/ml/weka/downloading.html> 에 접속하여 설치파일(105.5MB)을 다운 받아 설치한다. 아래와 같은 화면이 나오면 Next를 눌러 설치하면 된다.



ubuntu 설치

- ① 사이트 접속 후 설치 파일 다운로드
terminal 에서 설치 파일 위치로 이동 후
- ② \$ java -jar weka.jar 입력

unofficial package 다운로드

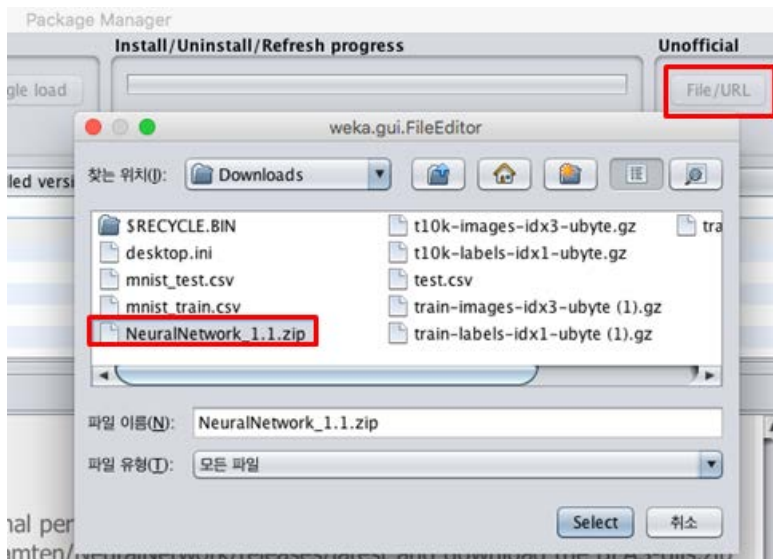
- ③ 사이트 접속 후 필요한 패키지 다운로드
<http://weka.wikispaces.com/Unofficial+packages+for+WEKA>
cf) 공식 사이트에 없는 패키지 - Convolutional Neural Network package

<https://github.com/mtten/NeuralNetwork>

④ Package Manager



⑤ 패키지 적용



4.2. TensorFlow

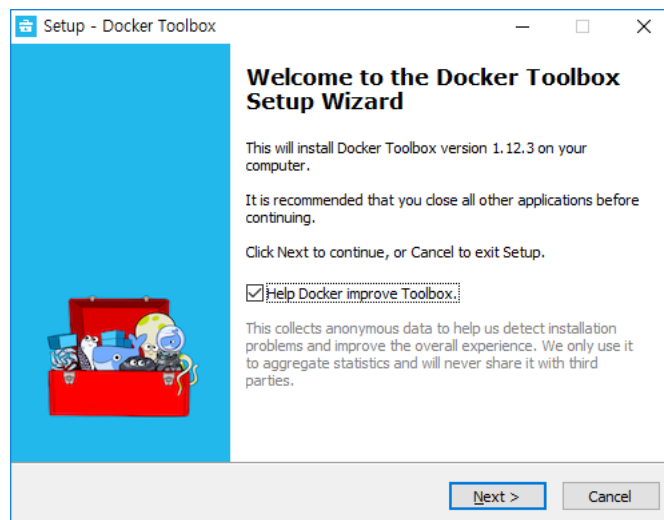
공식문서 참조

https://www.tensorflow.org/install/install_windows

Windows 설치

A. Docker installation

- ①. <https://www.docker.com/products/docker-toolbox> 에 접속하여 설치파일 (206MB)을 받고 실행



- ② Docker terminal을 실행 후, 아래 명령어를 Docker Quickstart Terminal을 실행하여 입력하여 TensorFlow를 설치한다.

<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

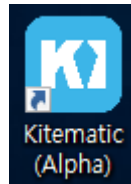
<pre>docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow</pre>
<pre>docker run -d -p 8888:8888 -p 6006:6006 eagle705/tensorflow1.2-python3-jupyter</pre>
<p>-p 8888:8888 -p 6006:6006 옵션은 Docker 컨테이너 내에서 포트 포워딩을 하기 위한 옵션으로, 8888은 <code>ipython notebook</code>을 위해서,</p> <p>6006은 <code>Tensorboard</code>를 위해서 사용</p> <p>(공식 이미지는 https://hub.docker.com/r/tensorflow/tensorflow/)</p>

[illegible]

③. <http://192.168.99.100:8888/tree> 를 통해 Jupyter Notebook으로 TensorFlow 실습환경에 접근할 수 있다.

+) Kitematic(alpha)로도 해당 컨테이너를 실행할 수 있다.

(만약 Virtual Box와 관련해 문제가 생길 경우,
BIOS setting에서 CPU 가상화 가속 옵션을 enable로 바꾼 후 다시 설치해본다.
단, 32bit OS인 경우 Docker Toolbox 설치가 불가능하다.)



Docker 명령어 정리

docker ps -a : 현재 실행중인 컨테이너 정보를 알 수 있음

docker restart 아이디 : 컨테이너를 활성화 함 (아이디는 **docker ps -a** 에서 확인가능, 일부만 입력해도 됨)

docker attach 아이디 : 해당 컨테이너에 접속

docker run -d -p 8888:8888 -p 6006:6006 dockerhub계정/이미지이름 : docker이미지를 다운받고 포트포워딩 설정 후 실행

docker images : image 목록을 볼 수 있음

docker login : docker hub계정으로 로그인 가능

docker commit -m “메시지” 아이디 **dockerhub계정/이미지이름:태그** : **docker** 이미지 커밋

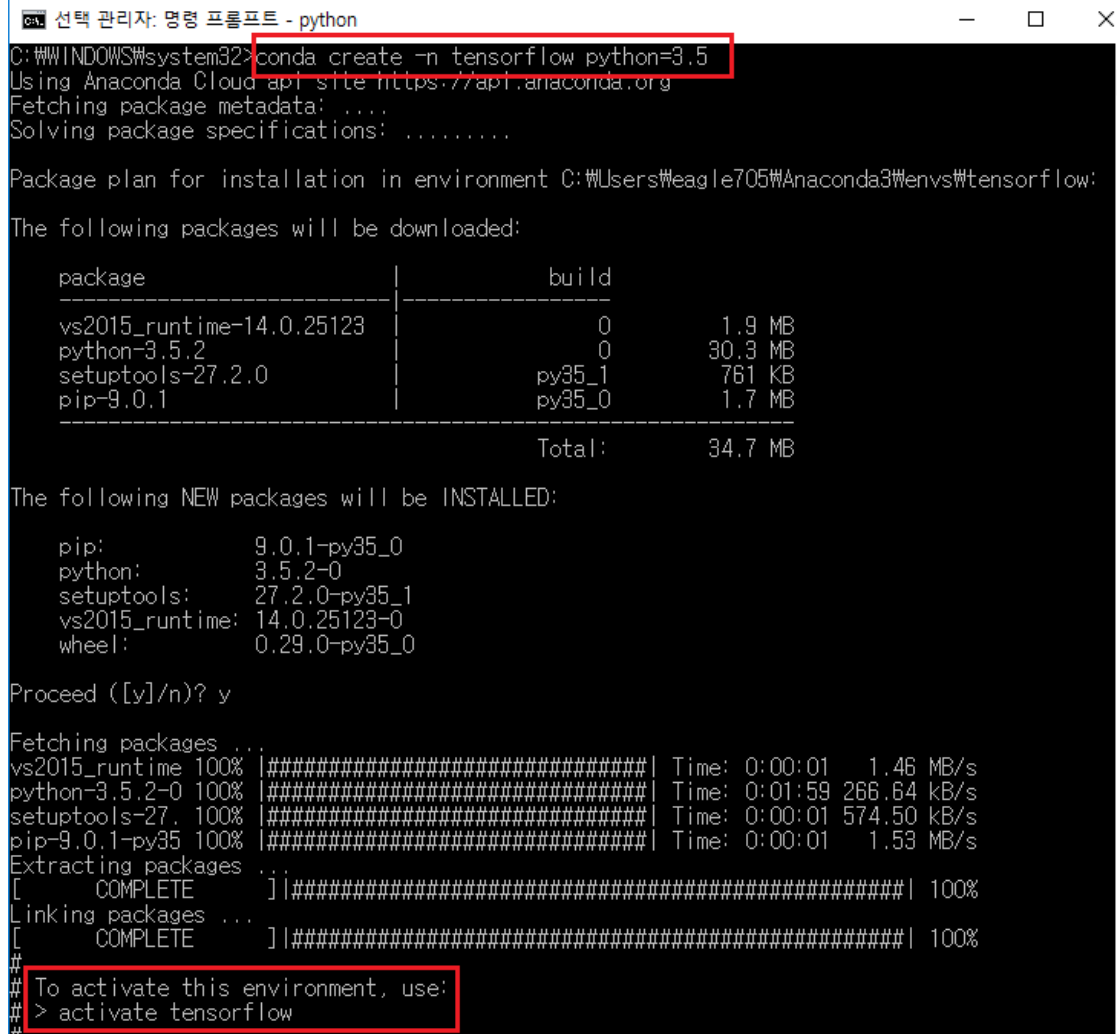
docker push dockerhub계정/이미지이름:태그 : **docker hub**에 이미지 업로드

B. Anaconda installation 이용

공식문서 참조 (https://www.tensorflow.org/install/install_windows)

① 환경설정

```
conda create -n tensorflow python=3.5
```



```
C:\WINDOWS\system32>conda create -n tensorflow python=3.5
Using Anaconda Cloud api site https://api.anaconda.org
Fetching package metadata: ....
Solving package specifications: .....

Package plan for installation in environment C:\Users\Weagle705\Anaconda3\envs\tensorflow:

The following packages will be downloaded:

package                        | build                | size
-----
vs2015_runtime-14.0.25123     | 0                    | 1.9 MB
python-3.5.2                  | 0                    | 30.3 MB
setuptools-27.2.0             | py35_1               | 761 KB
pip-9.0.1                    | py35_0               | 1.7 MB
-----
Total:                        |                      | 34.7 MB

The following NEW packages will be INSTALLED:

pip: 9.0.1-py35_0
python: 3.5.2-0
setuptools: 27.2.0-py35_1
vs2015_runtime: 14.0.25123-0
wheel: 0.29.0-py35_0

Proceed ([y]/n)? y

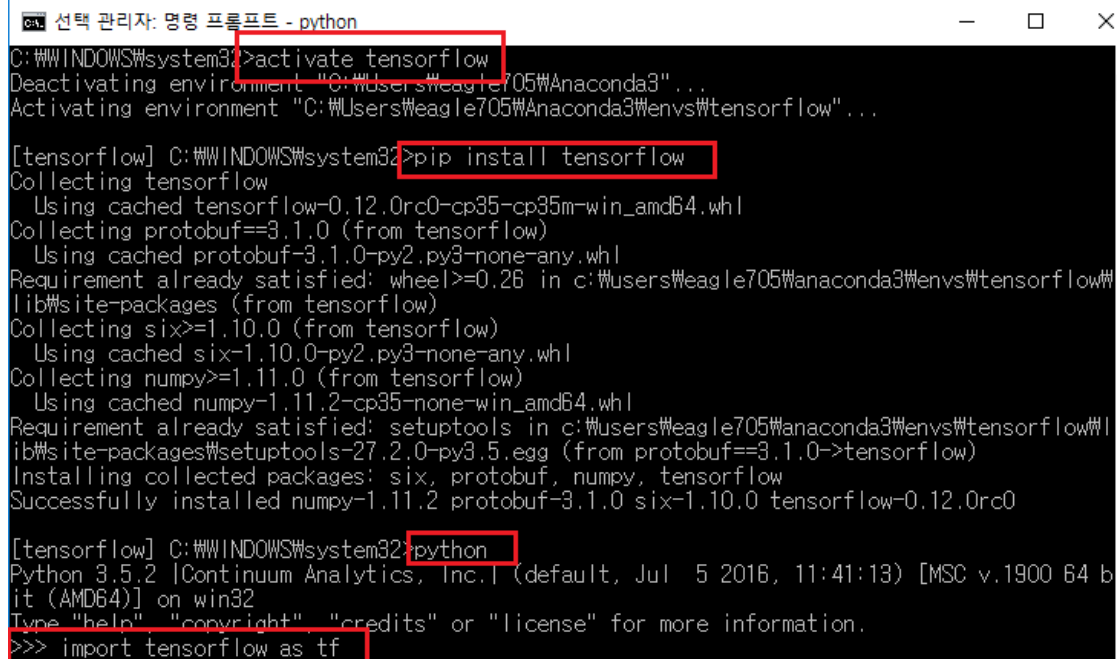
Fetching packages ...
vs2015_runtime 100% |#####| Time: 0:00:01 1.46 MB/s
python-3.5.2-0 100% |#####| Time: 0:01:59 266.64 kB/s
setuptools-27. 100% |#####| Time: 0:00:01 574.50 kB/s
pip-9.0.1-py35 100% |#####| Time: 0:00:01 1.53 MB/s
Extracting packages ...
[ COMPLETE ]|#####| 100%
Linking packages ...
[ COMPLETE ]|#####| 100%
#
# To activate this environment, use:
# > activate tensorflow
#
```

② conda 환경에 접속

```
activate tensorflow
```

③ TensorFlow 설치

pip install tensorflow



```
C:\WINDOWS\system32>activate tensorflow
Deactivating environment "C:\Users\weagle705\Anaconda3"...
Activating environment "C:\Users\weagle705\Anaconda3\envs\tensorflow"...

[tensorflow] C:\WINDOWS\system32>pip install tensorflow
Collecting tensorflow
  Using cached tensorflow-0.12.0rc0-cp35-cp35m-win_amd64.whl
Collecting protobuf==3.1.0 (from tensorflow)
  Using cached protobuf-3.1.0-py2.py3-none-any.whl
Requirement already satisfied: wheel>=0.26 in c:\Users\weagle705\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow)
Collecting six>=1.10.0 (from tensorflow)
  Using cached six-1.10.0-py2.py3-none-any.whl
Collecting numpy>=1.11.0 (from tensorflow)
  Using cached numpy-1.11.2-cp35-none-win_amd64.whl
Requirement already satisfied: setuptools in c:\Users\weagle705\anaconda3\envs\tensorflow\lib\site-packages\setuptools-27.2.0-py3.5.egg (from protobuf==3.1.0->tensorflow)
Installing collected packages: six, protobuf, numpy, tensorflow
Successfully installed numpy-1.11.2 protobuf-3.1.0 six-1.10.0 tensorflow-0.12.0rc0

[tensorflow] C:\WINDOWS\system32>python
Python 3.5.2 |Continuum Analytics, Inc.| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
```

④ 결과확인

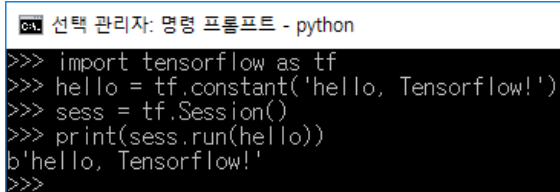
```
>>> import tensorflow as tf

>>> hello = tf.constant('Hello, TensorFlow!')

>>> sess = tf.Session()

>>> print(sess.run(hello))

Hello, TensorFlow!
```



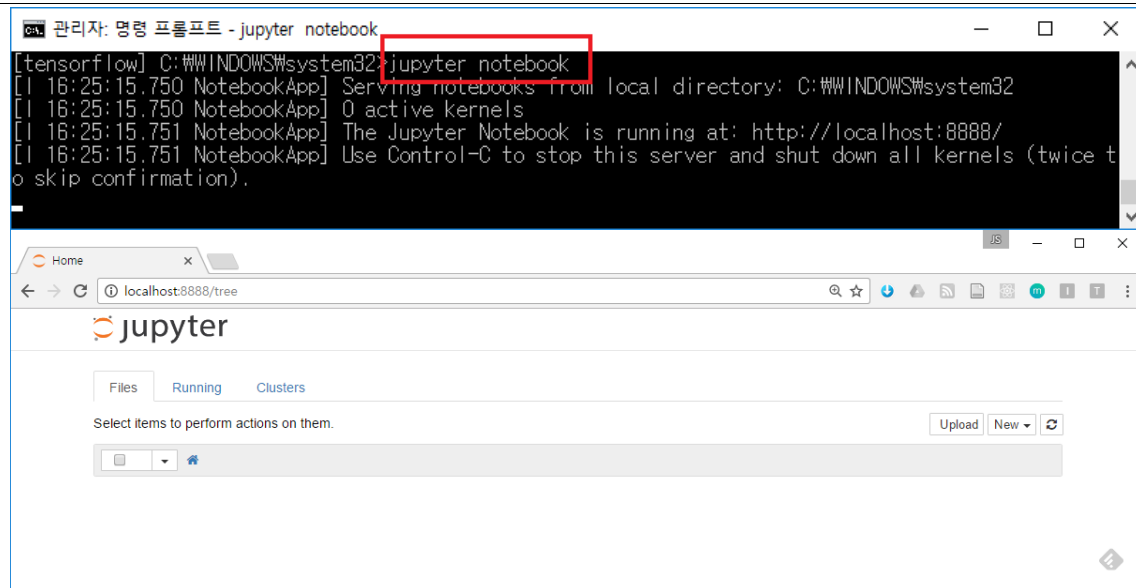
```
>>> import tensorflow as tf
>>> hello = tf.constant('hello, Tensorflow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
b'hello, Tensorflow!'
>>>
```

⑤Jupyter notebook 설치

```
conda install jupyter
```

⑥Jupyter notebook 실행 (파일이 있는 디렉토리에서 실행)

jupyter notebook



⑦Source Code Download (cmd / Docker container / Jupyter notebook terminal에서 실행)

```
git clone https://github.com/inikoreaackr/DeepLearningTutorials.git
```

No description, website, or topics provided.

[Edit](#)[Add topics](#)

18 commits

1 branch

0 releases

2 contributors

Branch: master ▾

[New pull request](#)[Create new file](#)[Upload files](#)[Find file](#)[Clone or download ▾](#)

inikoreaackr committed on GitHub Merge pull request #6 from eagle705/master ... Latest commit 9c1de2e a day ago

📁 (준비중)PyTorch	add more examples	a day ago
📁 .idea	add more examples	a day ago
📁 TF_version_upgrade_for_1.0_util	add more examples	a day ago
📁 TensorFlow/iPythonNotebook	add more examples	a day ago
📄 README.md	add more examples	a day ago

```
# git clone https://github.com/inikoreaackr/DeepLearningTutorials.git
Cloning into 'DeepLearningTutorials' ...
remote: Counting objects: 60, done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 60 (delta 27), reused 59 (delta 26), pack-reused 0
Unpacking objects: 100% (60/60), done.
Checking connectivity... done.
#
```