Assignment-2
제출자 : 강제순, 20170937

October 3, 2019

In [1]:
```
'''

Submitter : JESOON KANG, 20170937
Date : 2019. 10. 3


    Assignment 2.

- Binary classification based on logistic regression -


'''
import matplotlib.pyplot as plt
import math
import numpy as np
import random

import torch
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
import torchvision
import os
```

In [2]:
```
#### Section 1. #### This Section is bringed Data_import_ex.py file.

# Image Data import & resize

transform = transforms.Compose([#transforms.Resize((256,256)),
                                transforms.Grayscale(),                # the code tran
                                transforms.ToTensor(),])

#train_data_path = 'relative path of training data set'
train_data_path = './data/horse-or-human/train'
trainset = torchvision.datasets.ImageFolder(root=train_data_path, transform=transform)
# change the valuse of batch_size, num_workers for your program
# if shuffle=True, the data reshuffled at every epoch
trainloader = torch.utils.data.DataLoader(trainset, batch_size=1, shuffle=False, num_wo
```

```
validation_data_path = './data/horse-or-human/validation'
valset = torchvision.datasets.ImageFolder(root=validation_data_path, transform=transfor
# change the valuse of batch_size, num_workers for your program
valloader = torch.utils.data.DataLoader(valset, batch_size=1, shuffle=False, num_worke

#### Section 1 END ####
```

In [3]:
```
#### Section 2 START ####
# This Part includes Sigmoid, Hypothesis, Loss, Predict Functions

# sigmoid Function.
def sigmoid(z) :
    return 1 / (1 + math.e ** (-z))
```

In [4]:
```
# h(x) Function. hypothesis Func.
def hypothesis(weight,X) :
    theta = np.array(weight)
    z = np.dot(X,theta)
    h = sigmoid(z)
    return h
```

In [5]:
```
# Loss Function
def lossFunction(h,y) :
    # if i = 1, it occurs ZeroDivisionError ,
    # So, adjust value to 0.995
    for i in h :
        if i == 1 :
            i = 0.995
    try :
        ret = (-y * np.log(h) - (1-y) * np.log(1-h)).mean()
    except ZeroDivisionError :
        print("Error")
    finally :
        return (-y * np.log(h) - (1-y) * np.log(1-h)).mean()
```

In [6]:
```
# predict Function. if h(x) returns >=0.5, set to 1. other cases, set to 0.
def predict(h, labels) :
    mount = len(h)
    correct = 0

    for i in range(0,mount) :
        if h[i] >= 0.5 :
            if labels[i] == 1 :
                correct +=1
        else :
            if labels[i] == 0 :
                correct +=1
```

```
        return correct * (1/mount)

    #### Section 2 END ####
```

In [7]:
```
#### Section 3 START ####
# Section 3 includes Data Pre-processing. ReDesign datasets to easy calculate

# Data reconstruct. vectorize

# Each Image file will be stored shape of a row
training_vectorized = []
training_labels = []

validation_vectorized = []
validation_labels = []


# Training data vectorizing
for i, data in enumerate(trainloader) :
    train_data = []
    inputs, labels = data

    for u in inputs :
        for col in u[0] :
            train_data += list(col)
    training_vectorized.append(train_data)
    training_labels.append([labels])

training_vectorized = np.array(training_vectorized)
training_labels = np.array(training_labels)

# Validation data vectorizing
for i, data in enumerate(valloader) :
    val_data = []
    inputs, labels = data
    for u in inputs :
        for col in u[0] :
            val_data += list(col)
    validation_vectorized.append(val_data)
    validation_labels.append([labels])
validation_vectorized = np.array(validation_vectorized)
validation_labels = np.array(validation_labels)

#### Section 3 END ####
```

In [8]:
```
#### Section 4 START ####
# This Part includes declaring variables which will be used in train & predict & resul
```

```
# log variables setting : to record statements
log_training_loss = []
log_validation_loss = []
log_iter = []
log_training_acc = []
log_validation_acc = []

# Initial Weight Value
weight = np.zeros((10000,1),dtype=float)

# Learning rate
learning_rate = 0.0002

i = 0

log_training_loss.append(1)
# Training & Validation

while (True):
    i += 1
    h = hypothesis(weight,training_vectorized)

    gradient = np.dot(training_vectorized.T,h-training_labels) / len(training_vectorize

    # adjust weight values with gradient value
    weight-= learning_rate*gradient

    # get train loss value
    training_loss = lossFunction(h,training_labels)

    # get accuracy
    training_acc = predict(h,training_labels)

    h_validation = hypothesis(weight, validation_vectorized)
    validation_loss = lossFunction(h_validation, validation_labels)
    validation_acc = predict(h_validation, validation_labels)

    # add log data
    log_training_loss.append(training_loss)
    log_training_acc.append(training_acc)

    log_validation_loss.append(validation_loss)
    log_validation_acc.append(validation_acc)
    log_iter.append(i)


    # If loss value is nearly convergence, Stop training
```

```python
            if ( (abs(training_loss - log_training_loss[-2]) < 0.000001) ) :
                print("loss val is convergenced")
                break

            # But, Too many Epoch, Stop training.
            if (i == 10000) :
                print("EPOCH reached 10000. terminate process")
                break

            # Monitoring States
            if (i%100 == 0) :
                print("Current num_epoch : ",i)
                print("iter : ",i, "  t_loss : ",training_loss,"  t_acc : ", training_acc,"  v_


        print("Finished")
```

```
Current num_epoch :  100
iter :  100   t_loss :  0.6554004343762622   t_acc :  0.5851996105160663   v_loss:  0.63322684
Current num_epoch :  200
iter :  200   t_loss :  0.6291961804731352   t_acc :  0.6806231742940604   v_loss:  0.58146072
Current num_epoch :  300
iter :  300   t_loss :  0.6082936628510764   t_acc :  0.7108081791626095   v_loss:  0.54094719
Current num_epoch :  400
iter :  400   t_loss :  0.5911416794791897   t_acc :  0.7263875365141188   v_loss:  0.50864911
Current num_epoch :  500
iter :  500   t_loss :  0.5767399673308787   t_acc :  0.7380720545277507   v_loss:  0.48245171
Current num_epoch :  600
iter :  600   t_loss :  0.5644159074018921   t_acc :  0.747809152872444   v_loss:  0.46086909
Current num_epoch :  700
iter :  700   t_loss :  0.5537001473081345   t_acc :  0.7565725413826679   v_loss:  0.44284126
Current num_epoch :  800
iter :  800   t_loss :  0.5442549516954985   t_acc :  0.7633885102239533   v_loss:  0.42759876
Current num_epoch :  900
iter :  900   t_loss :  0.5358310731684737   t_acc :  0.7643622200584226   v_loss:  0.41457316
Current num_epoch :  1000
iter :  1000   t_loss :  0.5282406588913262   t_acc :  0.7643622200584226   v_loss:  0.40333737
Current num_epoch :  1100
iter :  1100   t_loss :  0.5213395803578329   t_acc :  0.7682570593963   v_loss:  0.39356518198
Current num_epoch :  1200
iter :  1200   t_loss :  0.5150155370694133   t_acc :  0.7682570593963   v_loss:  0.38500405816
Current num_epoch :  1300
iter :  1300   t_loss :  0.5091798308852055   t_acc :  0.7731256085686465   v_loss:  0.37745554
Current num_epoch :  1400
iter :  1400   t_loss :  0.5037615487921461   t_acc :  0.7779941577409932   v_loss:  0.37076186
Current num_epoch :  1500
iter :  1500   t_loss :  0.4987033689369433   t_acc :  0.7828627069133398   v_loss:  0.36479610
Current num_epoch :  1600
```

```
iter :  1600   t_loss :  0.4939584861357165   t_acc :  0.7848101265822786   v_loss:  0.3594551
Current num_epoch :  1700
iter :  1700   t_loss :  0.48948832485432503   t_acc :  0.7887049659201558   v_loss:  0.354654
Current num_epoch :  1800
iter :  1800   t_loss :  0.48526081574175584   t_acc :  0.7935735150925025   v_loss:  0.350323
Current num_epoch :  1900
iter :  1900   t_loss :  0.4812490816310464   t_acc :  0.7964946445959105   v_loss:  0.3464044
Current num_epoch :  2000
iter :  2000   t_loss :  0.4774304250897492   t_acc :  0.8033106134371958   v_loss:  0.3428477
Current num_epoch :  2100
iter :  2100   t_loss :  0.4737855407483774   t_acc :  0.801363193768257   v_loss:  0.33961172
Current num_epoch :  2200
iter :  2200   t_loss :  0.4702978970270941   t_acc :  0.8023369036027265   v_loss:  0.3366609
Current num_epoch :  2300
iter :  2300   t_loss :  0.46695324680881556   t_acc :  0.8033106134371958   v_loss:  0.333964
Current num_epoch :  2400
iter :  2400   t_loss :  0.46373923717369847   t_acc :  0.8042843232716651   v_loss:  0.331497
Current num_epoch :  2500
iter :  2500   t_loss :  0.4606450958865904   t_acc :  0.8062317429406037   v_loss:  0.3292355
Current num_epoch :  2600
iter :  2600   t_loss :  0.4576613778254603   t_acc :  0.8081791626095424   v_loss:  0.3271596
Current num_epoch :  2700
iter :  2700   t_loss :  0.45477975856903524   t_acc :  0.8091528724440117   v_loss:  0.325252
Current num_epoch :  2800
iter :  2800   t_loss :  0.45199286534612865   t_acc :  0.8081791626095424   v_loss:  0.323497
Current num_epoch :  2900
iter :  2900   t_loss :  0.4492941377790934   t_acc :  0.8111002921129503   v_loss:  0.3218832
Current num_epoch :  3000
iter :  3000   t_loss :  0.44667771253422767   t_acc :  0.8149951314508277   v_loss:  0.320396
Current num_epoch :  3100
iter :  3100   t_loss :  0.44413832726827174   t_acc :  0.8169425511197663   v_loss:  0.319026
Current num_epoch :  3200
iter :  3200   t_loss :  0.441671240236702   t_acc :  0.818889970788705   v_loss:  0.317764465
Current num_epoch :  3300
iter :  3300   t_loss :  0.4392721626819629   t_acc :  0.8198636806231743   v_loss:  0.3166013
Current num_epoch :  3400
iter :  3400   t_loss :  0.436937201703324   t_acc :  0.8198636806231743   v_loss:  0.31552973
Current num_epoch :  3500
iter :  3500   t_loss :  0.43466281176544175   t_acc :  0.821811100292113   v_loss:  0.3145426
Current num_epoch :  3600
iter :  3600   t_loss :  0.4324457533601719   t_acc :  0.8237585199610516   v_loss:  0.3136338
Current num_epoch :  3700
iter :  3700   t_loss :  0.4302830576183598   t_acc :  0.8257059396299903   v_loss:  0.3127977
Current num_epoch :  3800
iter :  3800   t_loss :  0.428171995892273   t_acc :  0.8276533592989289   v_loss:  0.31202926
Current num_epoch :  3900
iter :  3900   t_loss :  0.42611005350795905   t_acc :  0.830574488802337   v_loss:  0.3113236
Current num_epoch :  4000
```

```
iter :  4000   t_loss :  0.42409490702998504   t_acc :  0.8315481986368063   v_loss:  0.310676
Current num_epoch :  4100
iter :  4100   t_loss :  0.42212440449630245   t_acc :  0.8315481986368063   v_loss:  0.310084
Current num_epoch :  4200
iter :  4200   t_loss :  0.4201965481742384   t_acc :  0.8325219084712756   v_loss:  0.30954335
Current num_epoch :  4300
iter :  4300   t_loss :  0.41830947946436703   t_acc :  0.8364167478091529   v_loss:  0.309050
Current num_epoch :  4400
iter :  4400   t_loss :  0.4164614656408119   t_acc :  0.8383641674780915   v_loss:  0.3086019
Current num_epoch :  4500
iter :  4500   t_loss :  0.4146508881671356   t_acc :  0.8383641674780915   v_loss:  0.3081957
Current num_epoch :  4600
iter :  4600   t_loss :  0.4128762323685769   t_acc :  0.8393378773125609   v_loss:  0.3078291
Current num_epoch :  4700
iter :  4700   t_loss :  0.4111360782757266   t_acc :  0.8412852969814996   v_loss:  0.3074997
Current num_epoch :  4800
iter :  4800   t_loss :  0.4094290924831613   t_acc :  0.8422590068159689   v_loss:  0.3072052
Current num_epoch :  4900
iter :  4900   t_loss :  0.40775402089018087   t_acc :  0.8451801363193768   v_loss:  0.306943
Current num_epoch :  5000
iter :  5000   t_loss :  0.4061096822104926   t_acc :  0.8451801363193768   v_loss:  0.3067132
Current num_epoch :  5100
iter :  5100   t_loss :  0.40449496215416275   t_acc :  0.8451801363193768   v_loss:  0.306512
Current num_epoch :  5200
iter :  5200   t_loss :  0.4029088081989881   t_acc :  0.8471275559883155   v_loss:  0.3063385
Current num_epoch :  5300
iter :  5300   t_loss :  0.40135022488007843   t_acc :  0.8500486854917235   v_loss:  0.306191
Current num_epoch :  5400
iter :  5400   t_loss :  0.3998182695362704   t_acc :  0.8510223953261928   v_loss:  0.3060683
Current num_epoch :  5500
iter :  5500   t_loss :  0.39831204846031976   t_acc :  0.8510223953261928   v_loss:  0.305969
Current num_epoch :  5600
iter :  5600   t_loss :  0.39683071340688336   t_acc :  0.8510223953261928   v_loss:  0.305891
Current num_epoch :  5700
iter :  5700   t_loss :  0.39537345841832855   t_acc :  0.8510223953261928   v_loss:  0.305835
Current num_epoch :  5800
iter :  5800   t_loss :  0.39393951693354173   t_acc :  0.8510223953261928   v_loss:  0.305798
Current num_epoch :  5900
iter :  5900   t_loss :  0.3925281591493192   t_acc :  0.8510223953261928   v_loss:  0.3057812
Current num_epoch :  6000
iter :  6000   t_loss :  0.3911386896076942   t_acc :  0.8519961051606622   v_loss:  0.3057813
Current num_epoch :  6100
iter :  6100   t_loss :  0.389770444985814   t_acc :  0.8529698149951315   v_loss:  0.30579848
Current num_epoch :  6200
iter :  6200   t_loss :  0.3884227920677853   t_acc :  0.8529698149951315   v_loss:  0.3058316
Current num_epoch :  6300
iter :  6300   t_loss :  0.38709512588033457   t_acc :  0.8539435248296008   v_loss:  0.305880
Current num_epoch :  6400
```

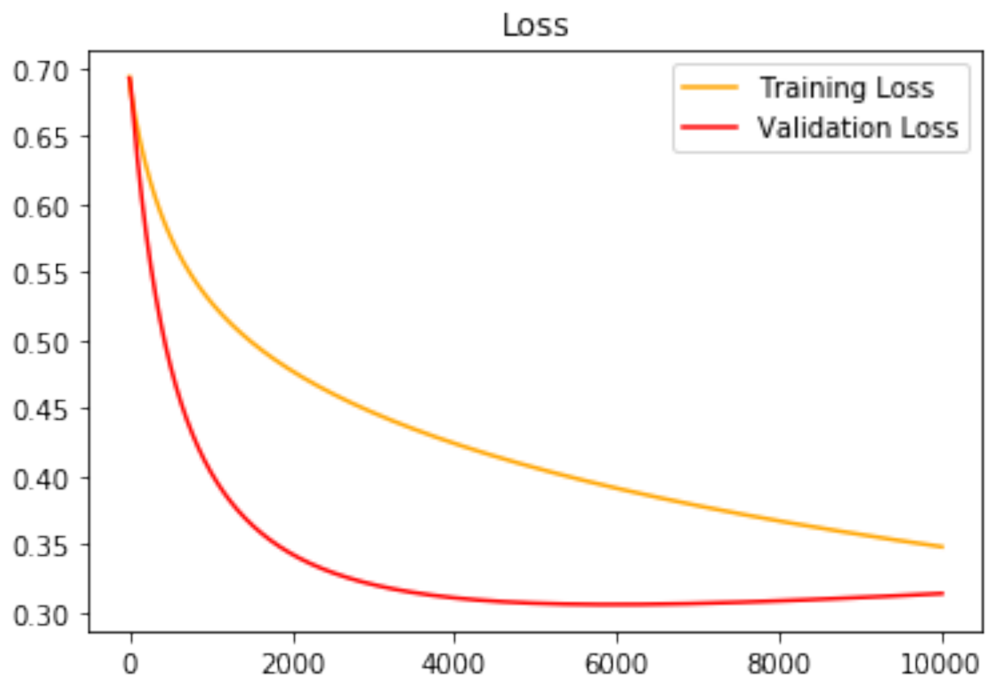
```
iter :  6400   t_loss :  0.38578686797623624   t_acc :  0.8539435248296008   v_loss:  0.3059430
Current num_epoch :  6500
iter :  6500   t_loss :  0.38449746485129316   t_acc :  0.8539435248296008   v_loss:  0.3060197
Current num_epoch :  6600
iter :  6600   t_loss :  0.38322638648224455   t_acc :  0.8539435248296008   v_loss:  0.3061096
Current num_epoch :  6700
iter :  6700   t_loss :  0.3819731249743732   t_acc :  0.8539435248296008   v_loss:  0.30621198
Current num_epoch :  6800
iter :  6800   t_loss :  0.3807371933087993   t_acc :  0.8539435248296008   v_loss:  0.30632618
Current num_epoch :  6900
iter :  6900   t_loss :  0.37951812418051734   t_acc :  0.8539435248296008   v_loss:  0.3064516
Current num_epoch :  7000
iter :  7000   t_loss :  0.3783154689191711   t_acc :  0.8539435248296008   v_loss:  0.30658795
Current num_epoch :  7100
iter :  7100   t_loss :  0.377128796485391   t_acc :  0.8549172346640701   v_loss:  0.306734453
Current num_epoch :  7200
iter :  7200   t_loss :  0.37595769253624267   t_acc :  0.8549172346640701   v_loss:  0.3068907
Current num_epoch :  7300
iter :  7300   t_loss :  0.3748017585539889   t_acc :  0.8558909444985394   v_loss:  0.30705626
Current num_epoch :  7400
iter :  7400   t_loss :  0.37366061103292975   t_acc :  0.8588120740019475   v_loss:  0.3072306
Current num_epoch :  7500
iter :  7500   t_loss :  0.37253388071960025   t_acc :  0.8597857838364168   v_loss:  0.3074135
Current num_epoch :  7600
iter :  7600   t_loss :  0.37142121190205274   t_acc :  0.8588120740019475   v_loss:  0.3076044
Current num_epoch :  7700
iter :  7700   t_loss :  0.37032226174435345   t_acc :  0.8597857838364168   v_loss:  0.3078029
Current num_epoch :  7800
iter :  7800   t_loss :  0.369236699662783   t_acc :  0.8597857838364168   v_loss:  0.308008880
Current num_epoch :  7900
iter :  7900   t_loss :  0.36816420674054895   t_acc :  0.8607594936708861   v_loss:  0.3082217
Current num_epoch :  8000
iter :  8000   t_loss :  0.3671044751781086   t_acc :  0.8617332035053554   v_loss:  0.30844124
Current num_epoch :  8100
iter :  8100   t_loss :  0.36605720777645534   t_acc :  0.8617332035053554   v_loss:  0.3086670
Current num_epoch :  8200
iter :  8200   t_loss :  0.3650221174509539   t_acc :  0.8617332035053554   v_loss:  0.30889899
Current num_epoch :  8300
iter :  8300   t_loss :  0.3639989267735207   t_acc :  0.8636806231742941   v_loss:  0.30913667
Current num_epoch :  8400
iter :  8400   t_loss :  0.3629873675411284   t_acc :  0.8636806231742941   v_loss:  0.30937984
Current num_epoch :  8500
iter :  8500   t_loss :  0.3619871803687868   t_acc :  0.8646543330087634   v_loss:  0.30962827
Current num_epoch :  8600
iter :  8600   t_loss :  0.3609981143053059   t_acc :  0.8646543330087634   v_loss:  0.30988170
Current num_epoch :  8700
iter :  8700   t_loss :  0.3600199264702801   t_acc :  0.8656280428432327   v_loss:  0.31013991
Current num_epoch :  8800
```

```
iter :  8800   t_loss :  0.3590523817108661   t_acc :  0.8656280428432327   v_loss:  0.3104026
Current num_epoch :  8900
iter :  8900   t_loss :  0.35809525227703354   t_acc :  0.8675754625121714   v_loss:  0.310669
Current num_epoch :  9000
iter :  9000   t_loss :  0.35714831751407505   t_acc :  0.8675754625121714   v_loss:  0.310941
Current num_epoch :  9100
iter :  9100   t_loss :  0.35621136357125716   t_acc :  0.8675754625121714   v_loss:  0.311216
Current num_epoch :  9200
iter :  9200   t_loss :  0.3552841831255751   t_acc :  0.8666017526777021   v_loss:  0.311495
Current num_epoch :  9300
iter :  9300   t_loss :  0.35436657511965824   t_acc :  0.8656280428432327   v_loss:  0.311777
Current num_epoch :  9400
iter :  9400   t_loss :  0.35345834451293967   t_acc :  0.8656280428432327   v_loss:  0.312063
Current num_epoch :  9500
iter :  9500   t_loss :  0.35255930204527186   t_acc :  0.8675754625121714   v_loss:  0.312353
Current num_epoch :  9600
iter :  9600   t_loss :  0.351669264012229   t_acc :  0.8685491723466408   v_loss:  0.31264537
Current num_epoch :  9700
iter :  9700   t_loss :  0.3507880520513897   t_acc :  0.8704965920155794   v_loss:  0.3129406
Current num_epoch :  9800
iter :  9800   t_loss :  0.3499154929389495   t_acc :  0.8704965920155794   v_loss:  0.3132388
Current num_epoch :  9900
iter :  9900   t_loss :  0.3490514183960496   t_acc :  0.8704965920155794   v_loss:  0.3135396
EPOCH reached 10000. terminate process
Finished
```

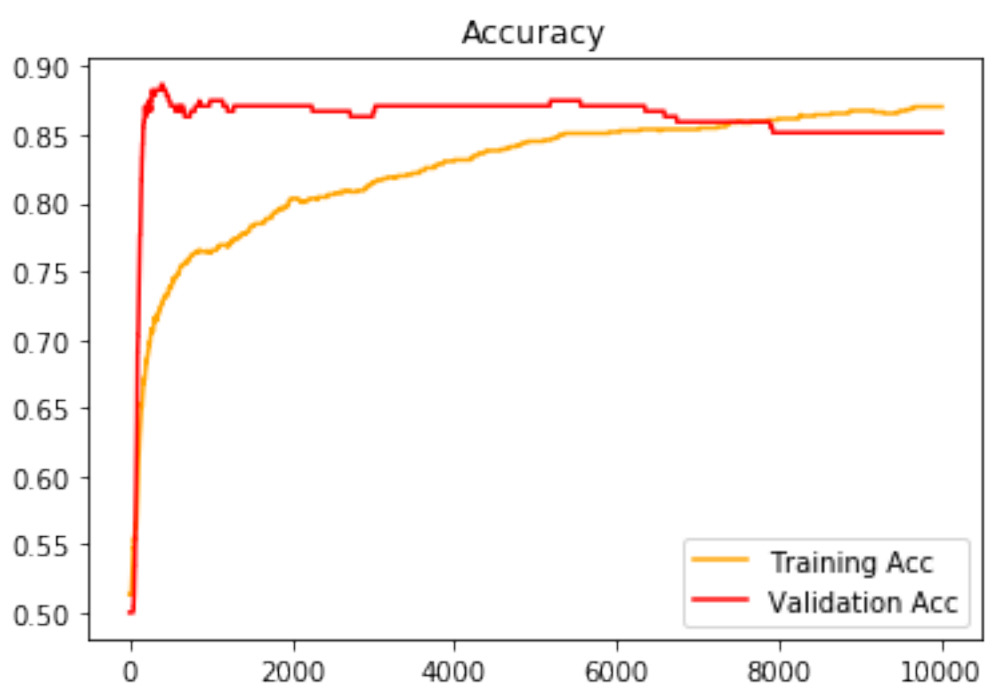
In [9]:
```
log_training_loss = log_training_loss[1:]
```

In [10]:
```
#### Section 5 ####
# Data Visualization


t1 = plt.plot(log_iter,log_training_loss, color='orange',label='Training Loss')
t2 = plt.plot(log_iter,log_validation_loss, color= 'red',label='Validation Loss')
plt.title("Loss")
plt.legend(['Training Loss','Validation Loss'])
plt.show()
```




```
In [11]: t1 = plt.plot(log_iter,log_training_acc, color='orange',label='Training Acc')
         t2 = plt.plot(log_iter,log_validation_acc, color= 'red',label='Validation Acc')
         plt.title("Accuracy")
         plt.legend(['Training Acc','Validation Acc'])
         plt.show()
```



```
In [12]: # commit Point. Assignment 2 Implement is END
```