CS398 HW1: Logistic Regression

Yanjun Guo (yanjung2)

In this assignment, I used the method of stochastic gradient descent to train a logistic regression model and reached 90.01% accuracy. Firstly, I obtained training data randomly by generating random integers. Then I and constructed a distribution function with softmax function. I made approximation with the derived formula from lecture, and upgrade new theta each time I iterate. With a learning rate of 0.01 and 15000 iterations, I finally got 90.01% accuracy for test data.

(Code next page)

In [28]:

```
import numpy as np

import h5py
import time
import copy
import random
from random import randint


#load MNIST data
MNIST_data = h5py.File('MNISTdata.hdf5', 'r')
x_train = np.float32(MNIST_data['x_train'][:] )
y_train = np.int32(np.array(MNIST_data['y_train'][:,0]))
x_test = np.float32( MNIST_data['x_test'][:] )
y_test = np.int32( np.array( MNIST_data['y_test'][:,0] ) )
MNIST_data.close()

################################################################################
####
#Implementation of stochastic gradient descent algorithm
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)


#number of inputs
num_inputs = 28*28
#number of outputs
num_outputs = 10
model = {}
model['W1'] = np.random.randn(num_outputs,num_inputs) / np.sqrt(num_inputs)
model_grads = copy.deepcopy(model)


########################################################

l_rate = 0.01
iteration_num = 15000
theta = model['W1']

step = np.zeros(10*784).reshape(10,784)

i = 0
while (i < iteration_num):
    idx = random.randint(0, len(x_train)-1)
    x_t = x_train[idx]
    y_t = y_train[idx]

    fun_x = softmax(theta@x_t)

    for k in range(10):
        if k == y_t:
            step[k] = (1-fun_x[k])*x_t
        else:
            step[k] = (-fun_x[k])*x_t
    theta = theta + l_rate*step
    i += 1

def forward(x, y, model):
    return theta@x
```

```
#test data

total_correct = 0

for n in range( len(x_test)):
    y = y_test[n]
    x = x_test[n][:]
    p = forward(x, y, model)
    prediction = np.argmax(p)
    if (prediction == y):
        total_correct += 1

print(total_correct/np.float(len(x_test) ) )
```

```
0.9001
```

In [ ]:

In [ ]: