

Statistical learning: Second assignment

Ali Zamani(96123035)

November 11, 2018

0.1 Import packages

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Mon Nov  5 17:55:23 2018
5
6 @author: ali
7 """
8 #Start import packages
9 import tensorflow as tf
10 import numpy as np
11 from sklearn.metrics import accuracy_score as acc
12 import matplotlib.pyplot as plt
13 #End import packages
```

0.2 Load Data Set:

In this section we define function loadDataSet for loading data. Inputs of this function are:

- 1)directory: address of data set file.
- 2)trainNum: number of training data.
- 3)validNum: number of validation data.

```
1 #%%
2 #Start Function to load notMNist dataset
3 def loadDataSet(directory, trainNum, validNum):
4     with np.load(directory) as data:
5         Data, Target = data["images"], data["labels"]
6         posClass = 2
7         negClass = 9
8         dataIndx = (Target==posClass) + (Target==negClass)
9         Data = Data[dataIndx]/255.
10        Target = Target[dataIndx].reshape(-1, 1)
11        Target[Target==posClass] = 1
12        Target[Target==negClass] = 0
13        np.random.seed(521)
14        randIndx = np.arange(len(Data))
15        np.random.shuffle(randIndx)
16        Data, Target = Data[randIndx], Target[randIndx]
17        Data = Data.reshape([-1, 784])
18        Target = Target.reshape([-1, 1])
19        trainData, trainTarget = Data[:trainNum], Target[:trainNum]
20        validData, validTarget = Data[trainNum:validNum+trainNum],
21        Target[trainNum:validNum+trainNum]
22        testData, testTarget = Data[validNum+trainNum:], Target[
23        validNum+trainNum:]
24        return trainData, trainTarget, validData, validTarget,
25        testData, testTarget
26 #End Function to load notMNist dataset
```

0.3 Define variables , constants and placeholders of the logistic regression model

First we set epochNum=5000 , batchSize=500 , trainNum=3500 ,validNum=100 and learningRate=1e-6 for maximum likelihood and 1e-6 for binary cross-entropy and regularized binary cross-entropy.

Data set was loaded by calling the loadDataSet function.

we must define placeholders, one of them for input(named as x) and another one for output(named as y).

Two variables w and b are also defined.

```
1  ###
2  #Start main program
3  epochNum = 5000
4  batchSize = 500
5  trainNum=3500
6  validNum=100
7  learningRate = 1e-4
8  print ("dataset is loading...")
9  trainData, trainTarget, validData, validTarget, testData,
    testTarget=loadDataSet("DataSet/notMNIST.npz",trainNum ,
    validNum)
10 print ("dataset was loaded!!!")
11 # Define placeholder x for input
12 x = tf.placeholder(dtype=tf.float64, shape=[None, 784], name="x")
13 # Define placeholder y for output
14 y = tf.placeholder(dtype=tf.float64, shape=[None, 1], name="y")
15 # Define variable w and fill it with random number
16 w = tf.Variable(tf.random_normal(shape=[784, 1], stddev=0.1, dtype=
    tf.float64), name="weights", dtype=tf.float64)
17 # Define variable b and fill it with zero
18 b = tf.Variable(tf.zeros(1, dtype=tf.float64), name="bias", dtype=
    tf.float64)
```

0.4 Define logistic regression model, loss functions and optimizers

ypredicted was defined for modeling.

Three loss functions lossML, lossBCE, lossRBCE were also defined for maximum likelihood, binary cross-entropy and regularized binary cross-entropy.

Gradient Descent Optimizer was used for optimizing.

```
1  # Define logistic Regression
2  logit = tf.matmul(x, w) + b
3  yPredicted = 1.0 / (1.0 + tf.exp(-logit))
4  # Define maximum likelihood(ML) loss function
5  lossML = -1 * tf.reduce_sum(y * tf.log(yPredicted) + (1 - y) * tf.
    log(1 - yPredicted))
6  # Define binary cross-entropy(BCE) loss function
```

```

7 lossBCE = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(
    labels=y, logits=logit))
8 # Define Regularized Binary Cross-Entropy(RBCE) loss function
9 lossRBCE = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(
    labels=y, logits=logit)) +\
10     tf.constant(0.5 * learningRate, dtype=tf.float64) * tf.
    pow(tf.linalg.norm(w), 2)
11 # Define optimizer: GradientDescent
12 optimizer = tf.train.GradientDescentOptimizer(learning_rate=
    learningRate).minimize(lossBCE)

```

0.5 Run Session

We run the session for three different loss functions and report the results.

```

1  ###
2  print("Parameters were initialized, Session is running ...")
3  trainErrorList = []
4  validErrorList = []
5  trainAccList = []
6  validAccList = []
7  with tf.Session() as sess:
8      sess.run(tf.global_variables_initializer())
9      for i in range(epochNum):
10         trainLoss = 0
11         for idx in range(trainNum//batchSize):
12             InputList = {x: trainData[idx*batchSize:(idx+1)*
batchSize],
13                         y: trainTarget[idx*batchSize:(idx+1)*
batchSize]}
14             _, trainL = sess.run([optimizer, lossBCE], feed_dict=
InputList)
15             trainLoss += trainL
16             trainAccList.append(acc(trainTarget, np.round(sess.run(
yPredicted, feed_dict={x: trainData}))))
17             validAccList.append(acc(validTarget, np.round(sess.run(
yPredicted, feed_dict={x: validData}))))
18             trainErrorList.append(trainLoss/trainNum)#number should be
used as constant
19             validErrorList.append(sess.run(lossML, feed_dict={x:
validData, y: validTarget})/100)
20
21             print("train accuracy =", format(i), trainAccList[i])
22             w_value, b_value = sess.run([w, b])
23             testAcc = acc(testTarget, np.round(sess.run(yPredicted,
feed_dict={x: testData})))
24             print("accuracy =", testAcc)
25 #titles=["Maximum likelihood", "Cross Entropy", "Regularized Cross
Entropy"]

```

0.6 Results

```

1  ###

```

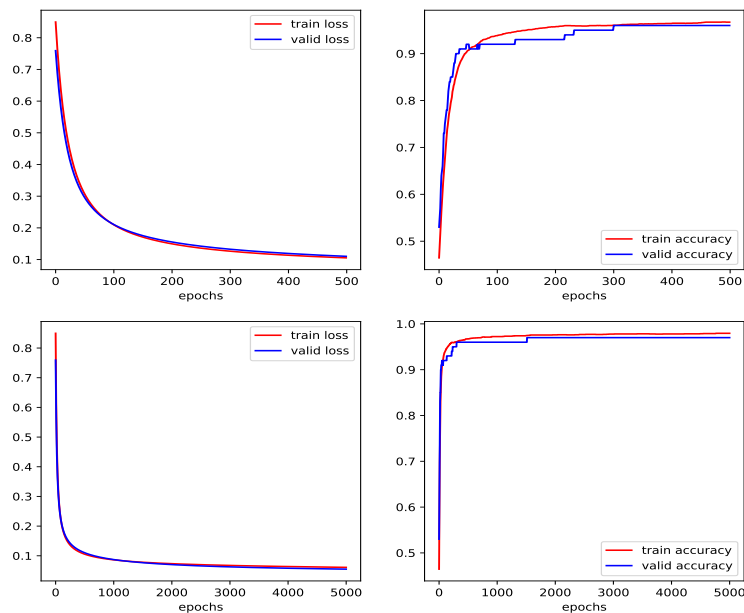
```

2 fig, ax = plt.subplots(2, 2, figsize=(10, 10))
3 fig.suptitle("BCE "+"test accuracy = " + str(testAcc))
4 for a in ax.reshape(-1,1):
5     a[0].set_xlabel("epochs")
6 ax[0][0].plot(trainErrorList[:500], color='red', label='train loss',
7 )
8 ax[0][0].plot(validErrorList[:500], color='blue', label='valid loss',
9 )
10 ax[0][0].legend()
11 ax[1][0].plot(trainErrorList, color='red', label='train loss')
12 ax[1][0].plot(validErrorList, color='blue', label='valid loss')
13 ax[1][0].legend()
14 ax[0][1].plot(trainAccList[:500], color='red', label='train
15 accuracy')
16 ax[0][1].plot(validAccList[:500], color='blue', label='valid
17 accuracy')
18 ax[0][1].legend()
19 ax[1][1].plot(trainAccList, color='red', label='train accuracy')
20 ax[1][1].plot(validAccList, color='blue', label='valid accuracy')
21 ax[1][1].legend()
22 plt.savefig("BCE"+" ".pdf")
23 #End main program

```

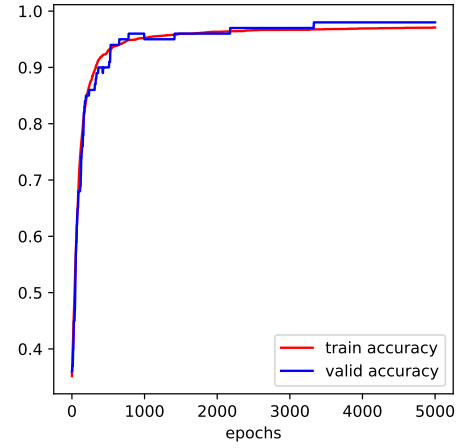
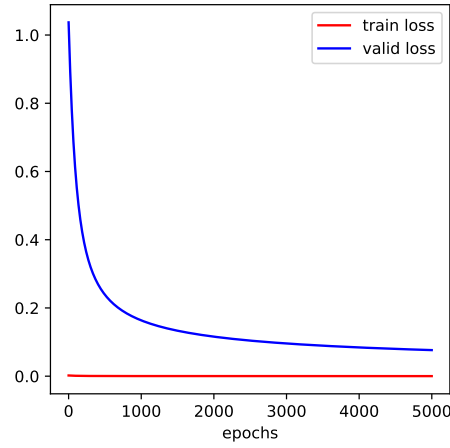
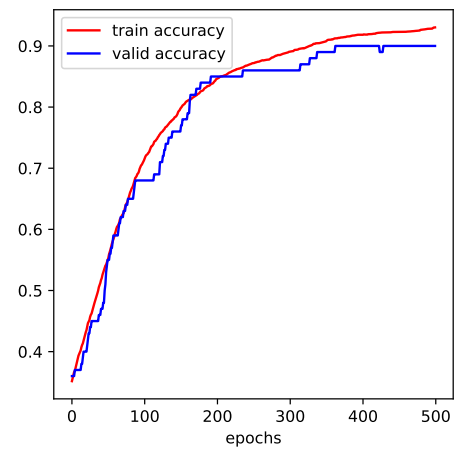
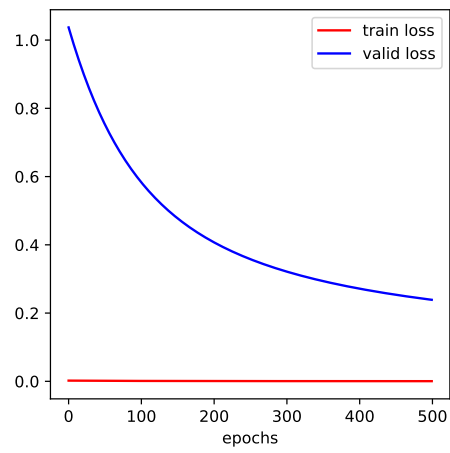
maximum likelihood:

Maximum likelihood test accuracy = 0.9655172413793104



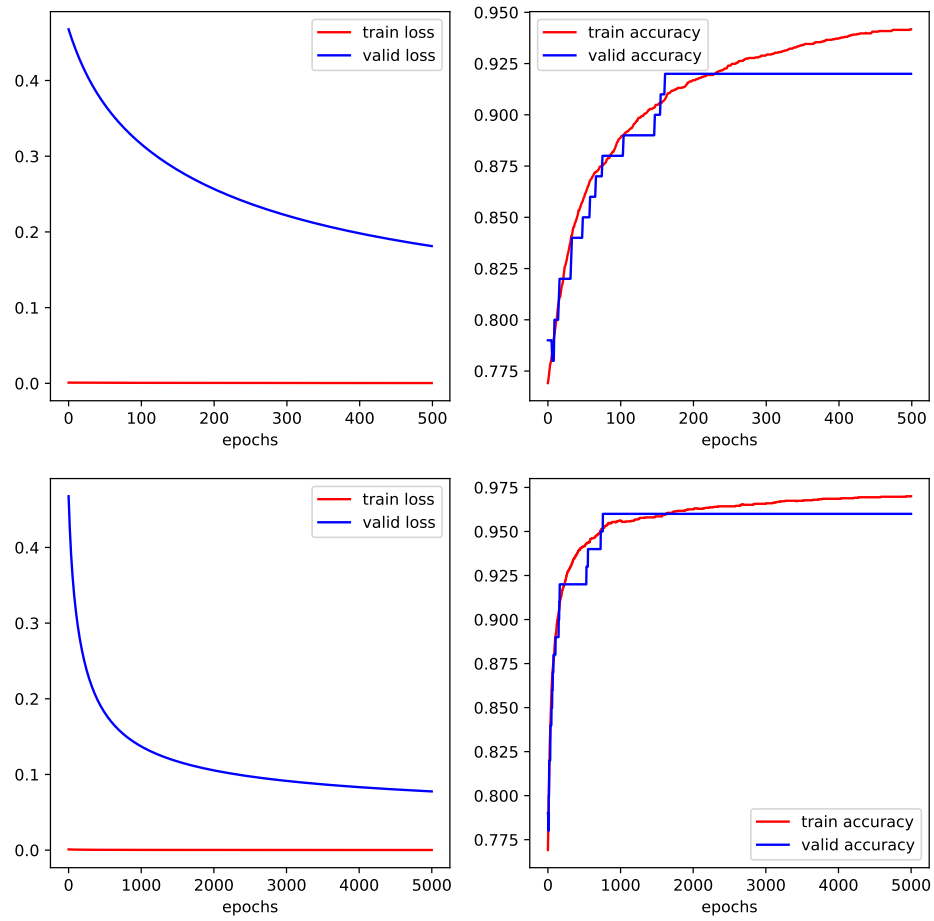
binary cross-entropy:

BCE test accuracy = 0.9655172413793104



regularized binary cross-entropy:

BCE test accuracy = 0.9793103448275862



[Source Code](#)