CSE 574: Introduction to Machine Learning (Fall 2017)

Saleem Ahmed
Person Number : 50247637
University at Buffalo

Problem Statement

1. Implement logistic regression, train it on the MNIST digit images and tune hyperparameters (Appendix 1).
2. Implement single hidden layer neural network, train it on the MNIST digit images and tune hyperparameters such as the number of units in the hidden layer (Appendix 2).
3. Use a publicly available convolutional neural network package, train it on the MNIST digit images and tune hyperparameters (Appendix 3).
4. Test your MNIST trained models on USPS test data and compare the performance with that of the MNIST data. Does your finding support the "No Free Lunch" theorem?

Methodology

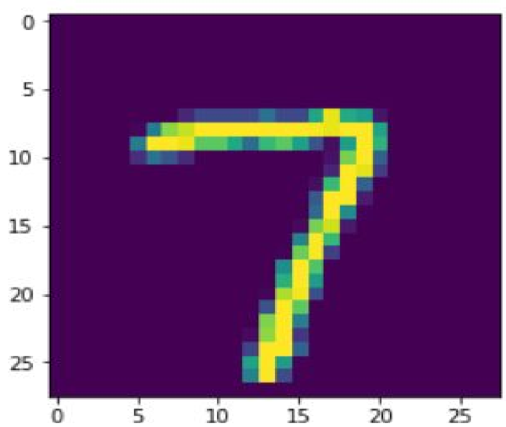
1. **MNIST data** - downloaded from tensorflow :

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

#Extract feature values and labels from the data
mnist_train_labels = np.array(mnist.train.labels)
mnist_train_images =  np.array(mnist.train.images)
mnist_valid_images =  np.array(mnist.validation.images)
mnist_valid_labels =  np.array(mnist.validation.labels)
mnist_test_labels =  np.array(mnist.test.labels)
mnist_test_images =  np.array(mnist.test.images)

```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

```
example = mnist_train_images[100]
mnist_train_images.shape
plt.imshow(np.reshape(example,[28,28]))
```

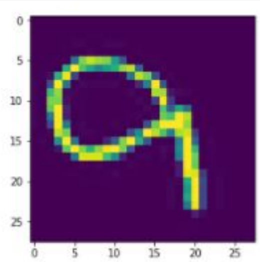
```
<matplotlib.image.AxesImage at 0x2789f5d1e80>
```



2. **USPS Data -**
    - Downloaded archive from UBLearns
    - Extract labels and images
    - Normalize image (values b/w 0-255 but MNIST values : 0-1)

```
def get_my_usps_data():
    import zipfile
    import os
    from PIL import Image
    import PIL.ImageOps
    import numpy as np
    import tensorflow  as tf
    import matplotlib.pyplot as plt

    filename="usps_dataset_handwritten.zip"

    #Defining height,width for resizing the images to 28x28 like MNIST digits
    height=28
    width=28

    #Defining path for extracting dataset zip file
    extract_path = "usps_data"

    #Defining image,label list
    images = []
    img_list = []
    labels = []

    #Extracting given dataset file
    with zipfile.ZipFile(filename, 'r') as zip:
        zip.extractall(extract_path)

    #Extracting labels,images array needed for training
    for root, dirs, files in os.walk("."):
        path = root.split(os.sep)

        if "Numerals" in path:
            image_files = [fname for fname in files if fname.find(".png") >= 0]
            for file in image_files:
                labels.append(int(path[-1]))
                images.append(os.path.join(*path, file))

    #Resizing images like MNIST dataset
    for idx, imgs in enumerate(images):
        img = Image.open(imgs).convert('L')
        img = img.resize((height, width), Image.ANTIALIAS)
        img_data = list(img.getdata())
        img_list.append(img_data)

    #Storing image and labels in arrays to be used for training
    USPS_img_array = np.array(img_list)
    USPS_img_array = np.subtract(255, USPS_img_array)
    USPS_label_array = np.array(labels)
    #print(USPS_label_array.shape)
    nb_classes = 10
    targets = np.array(USPS_label_array).reshape(-1)
    aa = np.eye(nb_classes)[targets]
    USPS_label_array = np.array(aa, dtype=np.int32)
    #print(USPS_label_array)


    USPS_img_array = np.float_(np.array(USPS_img_array))
    for z in range(len(USPS_img_array)):
        USPS_img_array[z] /= 255.0

    plt.imshow(USPS_img_array[19998].reshape(28,28))
    plt.show()


    return USPS_img_array, USPS_label_array
```

```
USPS_img_array, USPS_label_array = get_my_usps_data()
```



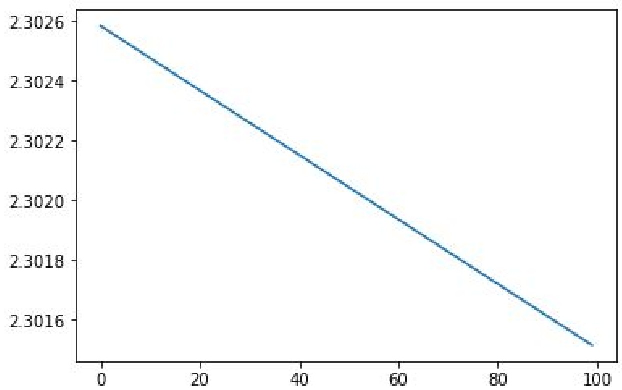
Results

1. Logistic Regression from Scratch

```
Loss for iteration :  99 is :  2.30151515596
2.30151515596
```

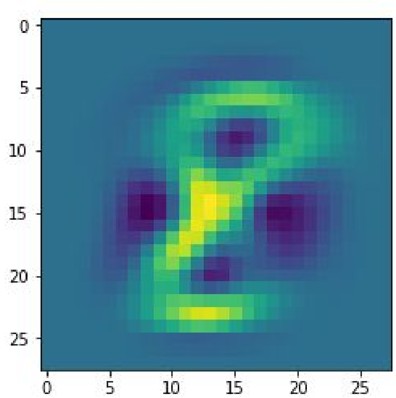
```
plt.plot(losses)
```

```
[<matplotlib.lines.Line2D at 0x278b6681c88>]
```




```
classWeightsToVisualize = 8
plt.imshow(scipy.reshape(w[:,classWeightsToVisualize],[28,28]))
```

```
<matplotlib.image.AxesImage at 0x278b674dda0>
```




```
print ('Training Accuracy: ', getAccuracy(x,y))
testX = mnist_test_images
testY = mnist_test_labels
print ( 'Test Accuracy: ', getAccuracy(testX,testY))
```

```
Training Accuracy:  0.6625272727272727
Test Accuracy:  0.671
```

Training Accuracy: 0.6625272727272727

Test Accuracy: 0.671

2. Logistic Regression Using Tensorflow

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
Epoch: 0001 cost= 1.183795122
Epoch: 0002 cost= 0.665195716
Epoch: 0003 cost= 0.552762509
Epoch: 0004 cost= 0.498667078
Epoch: 0005 cost= 0.465457584
Epoch: 0006 cost= 0.442547556
Epoch: 0007 cost= 0.425525734
Epoch: 0008 cost= 0.412158962
Epoch: 0009 cost= 0.401384600
Epoch: 0010 cost= 0.392403505
Epoch: 0011 cost= 0.384773200
Epoch: 0012 cost= 0.378174692
Epoch: 0013 cost= 0.372425060
Epoch: 0014 cost= 0.367293216
Epoch: 0015 cost= 0.362725902
Epoch: 0016 cost= 0.358642988
Epoch: 0017 cost= 0.354847474
Epoch: 0018 cost= 0.351436241
Epoch: 0019 cost= 0.348350410
Epoch: 0020 cost= 0.345441545
Epoch: 0021 cost= 0.342750276
Epoch: 0022 cost= 0.340259032
Epoch: 0023 cost= 0.337914452
Epoch: 0024 cost= 0.335685684
Epoch: 0025 cost= 0.333686076
Optimization Finished!
Accuracy of MNIST Training SET: 0.907691
Accuracy of MNIST VAlidation SET: 0.9146
Accuracy of MNIST TESTING SET: 0.914
Accuracy of USPS Numeral SET: 0.379719
```

Accuracy of MNIST Training SET: 0.907691
Accuracy of MNIST VAlidation SET: 0.9146
Accuracy of MNIST TESTING SET: 0.914
Accuracy of USPS Numeral SET: 0.379719

The same algorithm degrades over a different dataset, thus no free lunch is supported

3. Single Layer Neural Network With One Hidden Layer

```
            _, c = sess.run([train_op, loss_op], feed_dict={X: batch_x,
                                                            Y: batch_y})
            # Compute average loss
            avg_cost += c / total_batch
        # Display logs per epoch step
        if epoch % display_step == 0:
            print("Epoch:", '%04d' % (epoch+1), "cost={:.9f}".format(avg_cost))
    print("Optimization Finished!")

    # Test model
    pred = tf.nn.softmax(logits)  # Apply softmax to logits
    correct_prediction = tf.equal(tf.argmax(pred, 1), tf.argmax(Y, 1))
    # Calculate accuracy
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print("Accuracy of MNIST Train Data:", accuracy.eval({X: mnist.train.images, Y: mnist.train.labels}))
    print("Accuracy of MNIST Validation Data:", accuracy.eval({X: mnist.validation.images, Y: mnist.validation.labels}))
    print("Accuracy of MNIST Test Data:", accuracy.eval({X: mnist.test.images, Y: mnist.test.labels}))

    print("Accuracy for USPS Numerals:", accuracy.eval({X: USPS_img_array, Y: USPS_label_array}))
```

```
Epoch: 0001 cost=28.324336378
Epoch: 0002 cost=7.932121539
Epoch: 0003 cost=5.924172065
Epoch: 0004 cost=4.858874294
Epoch: 0005 cost=4.193946858
Epoch: 0006 cost=3.678970504
Epoch: 0007 cost=3.307560816
Epoch: 0008 cost=3.022878025
Epoch: 0009 cost=2.757018665
Epoch: 0010 cost=2.590802059
Epoch: 0011 cost=2.385162545
Epoch: 0012 cost=2.282708634
Epoch: 0013 cost=2.107260968
Epoch: 0014 cost=2.006777644
Epoch: 0015 cost=1.888892937
Optimization Finished!
Accuracy of MNIST Train Data: 0.900818
Accuracy of MNIST Validation Data: 0.8888
Accuracy of MNIST Test Data: 0.8879
Accuracy for USPS Numerals: 0.321866
```

```
def get_my_usps_data():
    import zipfile
    import os
    from PIL import Image
    import PIL.ImageOps
    import numpy as np
    import tensorflow  as tf
```

Accuracy of MNIST Train Data: 0.900818
Accuracy of MNIST Validation Data: 0.8888
Accuracy of MNIST Test Data: 0.8879
Accuracy for USPS Numerals: 0.321866

\

The same algorithm degrades over a different dataset, thus no free lunch is supported

4. Convolutional Neural Network

```
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  for i in range(20000):
    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))
    train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

  print('test accuracy %g' % accuracy.eval(feed_dict={
      x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))

  print('USPS accuracy %g' % accuracy.eval(feed_dict={
      x: USPS_img_array, y_: USPS_label_array, keep_prob: 1.0}))
```

```
step 0, training accuracy 0.1
step 100, training accuracy 0.82
step 200, training accuracy 0.86
step 300, training accuracy 0.92
step 400, training accuracy 0.92
step 500, training accuracy 0.98
step 600, training accuracy 1
step 700, training accuracy 0.96
step 800, training accuracy 0.98
```

Accuracy of MNIST Training SET: 0.98
Accuracy of MNIST Testing SET: 0.99
Accuracy of USPS : 0.44

The same algorithm degrades over a different dataset, thus no free lunch is supported