CS398-Deep Learning
Homework 2
Lingyi Xu (lingyix2)

1. **Description of Implementation**

First, I load the MNIST data from the dataset and preprocess the labels to one-hot format.

Fetch Data

```
# code from lecture notes
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]))
```

One_Hot Function for Data Pre-Process

```
def one_hot(Y):
    m = Y.shape[0]
    OHX = scipy.sparse.csr_matrix((np.ones(m), (Y, np.array(range(m)))
    OHX = np.array(OHX.todense()).T
    return OHX
```

```
one_hot_y_test = one_hot(y_test)
one_hot_y_train = one_hot(y_train)
```

Then I build my mini-batch softmax logistic regression model.
I define the nonlinearities σ(z) (relu) function I will use:

```
def relu(x):
    return np.maximum(0, x)
```

The forward propagation step:

Forward Propagation

```
def forward_prop(x, w1, w2):
    hidden_output = relu(np.dot(x,w1))
    output = relu(np.dot(hidden_output, w2))
    return(hidden_output, output)
```

And the gradient I will use for update weight matrix w:
When training, I first initialize w1 and w2 to random value, w1 is the matrix which get x as input and hidden layer as output and w2 is the matrix which get hidden layer as input and y as output. Then I use 14000 iterations to update the W by gradient decent.
For each iteration,we can update the parameter matrix w by

$$W^{(\ell+1)} = W^{(\ell)} - \alpha^{\ell}\left(\delta \odot \sigma'(Z)\right)X^{\top}$$

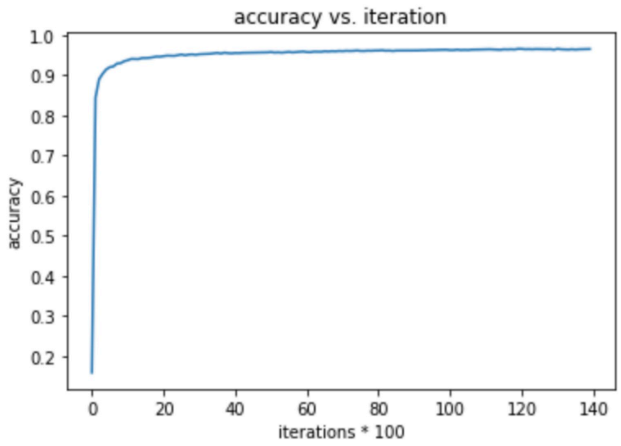
```
def train(self):
    #Randomly choose a batch from dataset
    n = [random.randint(0,len(self.X)-1) for i in range(self.batch_size)]
    batch_data = self.X[n]
    batch_label = self.y[n]
    #calculate the result of each layer
    hidden_output,output = forward_prop(batch_data, self.w1, self.w2)
    #calculate gradient of w2
    delta = output-batch_label
    grads_w2 = hidden_output.T.dot(delta)
    #calculate gradient of w1
    delta = (hidden_output > 0) * delta.dot(self.w2.T)
    grads_w1 = batch_data.T.dot(delta)
    #update w1 and w2
    self.w1-= lr*(grads_w1/(self.batch_size))
    self.w2-= lr*(grads_w2/(self.batch_size))
```

For this particular dataset, I use the following parameters:
Iteration = 14000
Batch size = 30
Learning rate = 0.25/（0.001*itr+1）

2. Final Test Accuracy:
96.5%

Plot the Accuracy vs. Iteration

```
1 plt.plot(accs)
2 plt.xlabel("iterations * 100")
3 plt.ylabel("accuracy")
4 plt.title("accuracy vs. iteration")
5 plt.show()
```



Print the Final Accuracy

```
1 print("final accuracy of test set is:", accs[-1])
```

```
final accuracy of test set is: 0.965
```