CSE 250A HW5.5

November 6, 2017

```
In [1]: '''
        Performing logistic regression via GRADIENT ASCENT to classify handwritten digits
        '''

Out[1]: '\nPerforming logistic regression via GRADIENT ASCENT to classify handwritten digits\n'

In [75]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

In [76]: # load files
         train3_fh = 'hw5_train3.txt'
         test3_fh = 'hw5_test3.txt'
         train5_fh = 'hw5_train5.txt'
         test5_fh = 'hw5_test5.txt'

         train3 = np.loadtxt(train3_fh, dtype=int)
         test3 = np.loadtxt(test3_fh, dtype=int)
         train5 = np.loadtxt(train5_fh, dtype=int)
         test5 = np.loadtxt(test5_fh, dtype=int)

         # collect together
         train = np.append(train3, train5, axis=0)
         test = np.append(test3, test5, axis=0)
         train_labs = [0] * train3.shape[0] + [1] * train5.shape[0]
         test_labs = [0] * test3.shape[0] + [1] * test5.shape[0]

         # CONSTANTS
         STEPS = 5000

In [83]: # functions
         def sigmoid(w,xt):
             z = np.dot(w,xt)
             return(1/(1+np.exp(-z)))

         def log_likelihood(yt, w, xt):
             L = yt*np.log(sigmoid(w,xt)) + (1-yt)*np.log((1-sigmoid(w,xt)))
             return(L)
```

```
def gradient(yt,w,xt):
    dL = np.multiply(yt-sigmoid(w, xt), xt)
    return(dL)

def learn(x_data, y_data):
    T = x_data.shape[0]
    eta = 0.02/T
    w = np.random.randint(2, size=x_data.shape[1])
    Lw_list = []
    percent_err = []
    for i in range(STEPS):
        Lw = 0
        dL_sum = [0] * x_data.shape[1]
        correct = 0
        for j in range(T):
            Lw += log_likelihood(y_data[j], w, x_data[j])
            dL_sum += gradient(y_data[j], w, x_data[j])
            if (y_data[j]==1 and sigmoid(w, x_data[j]) > 0.5) or (y_data[j]==0 and sigm
                correct += 1
        Lw_list.append(Lw)
        w = w + eta*dL_sum
        err = (T-correct)/T*1.0
        percent_err.append(err)
        if i%100 == 0 :
            print('iteration %d' % i)
            print('percent error=%f' % err)
    return Lw_list, w, percent_err

def predict(x_data, y_data, w):
    correct = 0
    for i in range(x_data.shape[0]):
        s = sigmoid(w,x_data[i])
        if (y_data[i]==1 and s>0.5) or (y_data[i]==0 and s<0.5):
                correct += 1
    err = (x_data.shape[0]-correct)/x_data.shape[0]*1.0
    return err
```

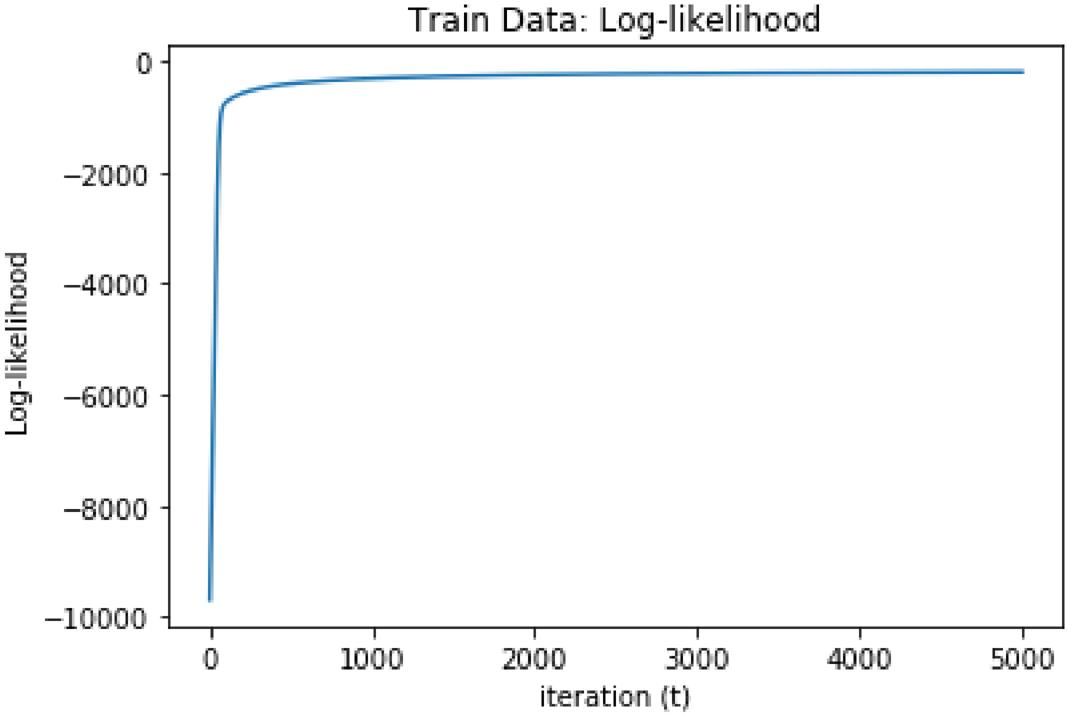
```
In [42]: Lw_train, w_train, err_train = learn(train, train_labs)
```

```
iteration 0
err=0.500000
iteration 100
err=0.246429
iteration 200
err=0.177143
iteration 300
err=0.140000
```

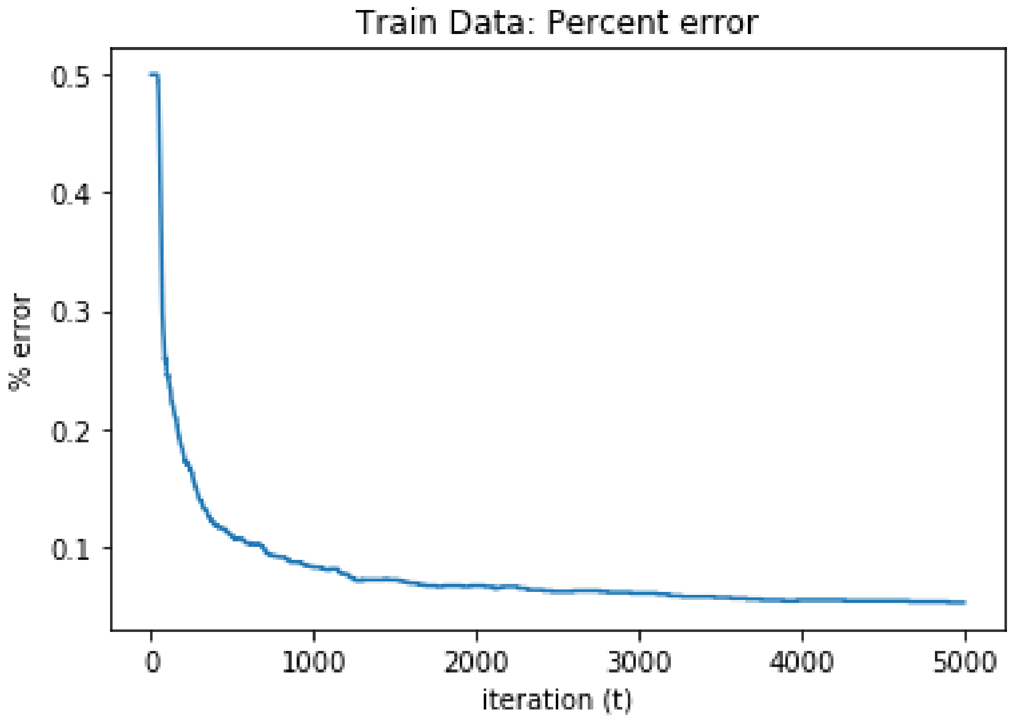
```
iteration 400
err=0.117857
iteration 500
err=0.109286
iteration 600
err=0.102857
iteration 700
err=0.097143
iteration 800
err=0.091429
iteration 900
err=0.087143
iteration 1000
err=0.082857
iteration 1100
err=0.080714
iteration 1200
err=0.077143
iteration 1300
err=0.072143
iteration 1400
err=0.072143
iteration 1500
err=0.072143
iteration 1600
err=0.069286
iteration 1700
err=0.067143
iteration 1800
err=0.067143
iteration 1900
err=0.067143
iteration 2000
err=0.067143
iteration 2100
err=0.065714
iteration 2200
err=0.066429
iteration 2300
err=0.065000
iteration 2400
err=0.063571
iteration 2500
err=0.062143
iteration 2600
err=0.062857
iteration 2700
err=0.062857
```

```
iteration 2800
err=0.062143
iteration 2900
err=0.061429
iteration 3000
err=0.060714
iteration 3100
err=0.060714
iteration 3200
err=0.058571
iteration 3300
err=0.057857
iteration 3400
err=0.057857
iteration 3500
err=0.057143
iteration 3600
err=0.056429
iteration 3700
err=0.055714
iteration 3800
err=0.055000
iteration 3900
err=0.054286
iteration 4000
err=0.055000
iteration 4100
err=0.055000
iteration 4200
err=0.055000
iteration 4300
err=0.054286
iteration 4400
err=0.054286
iteration 4500
err=0.054286
iteration 4600
err=0.054286
iteration 4700
err=0.053571
iteration 4800
err=0.053571
iteration 4900
err=0.053571
```

```
In [48]: plt.plot(Lw_train)
         plt.title("Train Data: Log-likelihood")
         plt.xlabel("iteration (t)")
         plt.ylabel("Log-likelihood")
         plt.savefig('log_likelihood.png')
```



```
In [53]: plt.plot(err_train)
         plt.title("Train Data: Percent error")
         plt.xlabel("iteration (t)")
         plt.ylabel("% error")
         plt.savefig('error.png')
```



```
In [82]: # '\n'.join([''.join(str(w_train[i:i+8])) for i in range(0,len(w_train),8)])
         with open('trained_weight_vector.txt', 'w') as f:
             count = 0
             for i in range(len(w_train)):
                 f.write(str(w_train[i]) + ' ')
                 count += 1
                 if count%8 == 0 and count>0:
                     count = 0
                     f.write('\n')
```

```
In [84]: test_err = predict(test, test_labs, w_train)
         test_err
```

```
Out[84]: 0.05375
```