hw8 - Neural Network

1. load data

```
data = loadmat('hw8data1.mat')
data
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: GLNXA64, Created on: Sun Oct 16 13:09:09 2011',
 '__version__': '1.0',
 '__globals__': [],
 'X': array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 'y': array([[10],
        [10],
        [10],
        ...,
        [ 9],
        [ 9],
        [ 9]], dtype=uint8)}
```

```
X = data['X']
y = data['y']

X.shape, y.shape    #看一下维度信息
```

```
((5000, 400), (5000, 1))
```

2. training

```
from scipy.optimize import minimize

# minimize the objective function
fmin = minimize(fun=backprop, x0=params, args=(input_size, hidden_size, num_labels, X, y_onehot, learning_rate),
                method='TNC', jac=True, options={'maxiter': 500})
fmin
```

```
<ipython-input-90-409996d00ef2>:21: RuntimeWarning: divide by zero encountered in log
  second_term = np.multiply((1 - y[i,:]), np.log(1 - h[i,:]))
<ipython-input-90-409996d00ef2>:21: RuntimeWarning: invalid value encountered in multiply
  second_term = np.multiply((1 - y[i,:]), np.log(1 - h[i,:]))
```

```
     fun: 0.07253714493629515
     jac: array([ 1.72365272e-04, -5.45188557e-07,  2.20550419e-07, ...,
       -5.49461179e-06, -5.57995587e-05,  4.73175080e-06])
 message: 'Linear search failed'
    nfev: 409
     nit: 24
  status: 4
 success: False
       x: array([-0.95792615, -0.02725943,  0.01102752, ..., -1.64164294,
       -3.01865411,  2.58786893])
```

3. predict

```
X = np.matrix(X)
theta1 = np.matrix(np.reshape(fmin.x[:hidden_size * (input_size + 1)], (hidden_size, (input_size + 1))))
theta2 = np.matrix(np.reshape(fmin.x[hidden_size * (input_size + 1):], (num_labels, (hidden_size + 1))))

a1, z2, a2, z3, h = forward_propagate(X, theta1, theta2)
y_pred = np.array(np.argmax(h, axis=1) + 1)
y_pred
```

```
array([[10],
       [10],
       [10],
       ...,
       [ 9],
       [ 9],
       [ 9]], dtype=int64)
```

准确率

```
correct = [1 if a == b else 0 for (a, b) in zip(y_pred, y)]
accuracy = (sum(map(int, correct)) / float(len(correct)))
print ('accuracy = {0}%'.format(accuracy * 100))
```

```
accuracy = 99.98%
```

4. tuning

4.1 调整学习率

```
learning_rate = 0.1
```

把学习率从 1 调到 0.1, 步长更小会收敛到一个更准确的位置。

4.2 调整 hidden layer 的神经元个数

```
hidden_size = 30
```

增加了参数个数会提高拟合效果。

4.3 提高 max_iter

```
options={'maxiter': 500})
```

增加训练的最大轮数有机会收敛到一个更准确的位置。