Deep Learning - Assignment 1 - Solution

USPS collection of handwritten digits

Produced by : 2016110648 金融实验班 李怡婷

Part 1

```
def eval_obj_grad(params, data, wd):
    W_hid, W_out = params['W_hid'], params['W_out']

    loss = 0
    z_hid = W_hid.dot(data['X'])
    a_hid = sigmoid(z_hid)
    z_out = W_out.dot(a_hid)
    y_predict = np.exp(z_out - log_sum_exp(z_out))

    cross_entropy = 0
    cross_entropy += - np.log(y_predict) * data['y']

    weight_decay=0
    weight_decay=wd/2*
(np.square(np.linalg.norm(W_out))+np.square(np.linalg.norm(W_hid)))

    loss =  np.sum(cross_entropy)/data['y'].shape[1] + weight_decay  /
data['X'].shape[1]

    # todo implement the backward prapagation
    n_hidden=100
    grad_W_out = np.zeros((10, n_hidden))
    grad_W_hid = np.zeros((n_hidden, 256))
```

```
    error_out = (y_predict - data['y']) / data['y'].shape[1]
    dw2 = np.dot(error_out, a_hid.T)
    grad_W_out = dw2 + wd * W_out / data['X'].shape[1]

    error_hid = a_hid * (1 - a_hid) * np.dot(W_out.T, error_out)
    dw1 = np.dot(error_hid, data['X'].T)
    grad_W_hid = dw1 + wd * W_hid / data['X'].shape[1]

    grad = {'W_out': grad_W_out,
            'W_hid': grad_W_hid,}

    return loss, grad
```

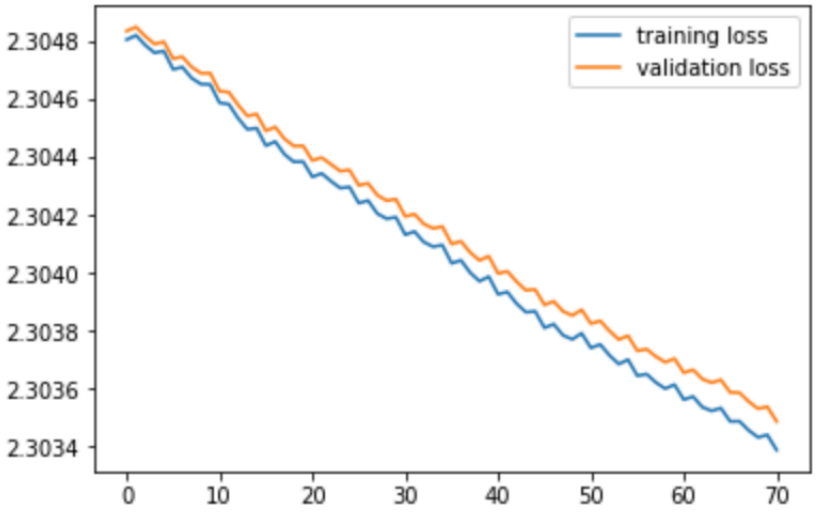
```
def eval_obj(params, data, wd):
    W_hid, W_out = params['W_hid'], params['W_out']

    loss = 0
    z_hid = W_hid.dot(data['X'])
    a_hid = sigmoid(z_hid)
    z_out = W_out.dot(a_hid)
    y_predict = np.exp(z_out - log_sum_exp(z_out))

    cross_entropy = 0
    cross_entropy += - np.log(y_predict) * data['y']

    weight_decay=0
    weight_decay=wd/2*
(np.square(np.linalg.norm(W_out))+np.square(np.linalg.norm(W_hid)))

    loss =  np.sum(cross_entropy)/data['y'].shape[1] + weight_decay /
data['X'].shape[1]

    return loss
```

Part2

1. Learning rate

Run train(0, 10, 70, 0.005, 0, False, 4) and what is the total loss?

!




```
{'loss_train': 2.3033889920196877,
 'loss_valid': 2.3034877917584526,
 'loss_test': 2.303475700379494}
```

Now let learning rate be chosen from {0.002, 0.01, 0.05, 0.2, 1.0, 5.0}. Find out the best learning rate on validation performance.

Learning_rate	loss_train	loss_valid	loss_test
0.002	2.304260	2.304316	2.304330
0.01	2.302024	2.302194	2.302139
0.05	2.292717	2.293463	2.293053
0.2	2.226031	2.229733	2.227115
1.0	1.572740	**1.578626**	1.567040
5.0	2.301322	2.301741	2.302015

Maximum performance: Learningrate 1.0

2. Classical momentum

Combination	loss_train	loss_valid	loss_test
l1m1	2.301054	2.301254	2.301191
l1m2	2.297801	2.298167	2.298013
l1m3	2.271529	2.273230	2.272309
l2m1	2.287880	2.288793	2.288329
l2m2	2.263925	2.265962	2.264805
l2m3	1.711966	1.723489	1.713227
l3m1	2.127231	2.133296	2.128699
l3m2	1.730742	1.741502	1.731620
l3m3	0.637584	0.730685	0.685974
l4m1	0.959017	1.002423	0.982117
l4m2	0.492569	**0.583290**	0.555243
l4m3	1.624751	1.741678	1.701367
l5m1	0.485624	0.583969	0.592234
l5m2	1.516542	1.562454	1.520241
l5m3	2.170323	2.269300	2.189903

Maximum performance: Learningrate 1.0 momemtummul 0.5

3. Nesterov Accelerated Method

Combination	loss_train	loss_valid	loss_test
l1m1	2.301054	2.301254	2.301191
l1m2	2.297867	2.298234	2.298078
l1m3	2.208845	2.211843	2.209502
l2m1	2.287880	2.288793	2.288329
l2m2	2.257571	2.259803	2.258473
l2m3	1.577063	1.593422	1.582482
l3m1	2.127231	2.133296	2.128699
l3m2	1.705942	1.717178	1.707245
l3m3	0.540118	0.635027	0.596546
l4m1	0.959017	1.002423	0.982117
l4m2	0.496829	**0.579388**	0.552242
l4m3	0.980447	1.101497	1.062799
l5m1	0.485624	0.583969	0.592234
l5m2	1.103199	1.122893	1.116112
l5m3	10.930663	10.863314	11.050260

As for Nesterov Accelerated Method, the best performance on validation data is under the learning rate 1.0 and the momentum mul 0.5. The result shows that the best parameter is the same and Nesterov gives a better performance.

Part 3

Part3 is based on the best setting given by Nesterov method:

```
learning_rate = 0.2
momentum_mul = 0.9
```

1. Weight dacay

Weight_decay	loss_train	loss_valid	loss_test
0	0.063827	0.286145	0.315128
1e-4	0.064251	0.286155	0.314878
1e-3	0.063514	0.282262	0.311668
1e-2	0.059174	**0.260867**	0.282776
1e-1	0.415202	0.475569	0.478152
1	2.207311	2.209992	2.207342
10	2.418840	2.418310	2.418074

The best performance is under weight decay 1e-2.

2. Hidden units, early stopping

Combination	loss_train	loss_valid	loss_test
10_on	0.031078	0.342888	0.357664
10_off	0.029501	0.357989	0.372058
50_on	0.028977	0.263903	0.294212
50_off	0.005364	0.282035	0.297399
100_on	0.090184	**0.253990**	0.298894
100_off	0.003760	0.277113	0.318276
200_on	0.063827	0.286145	0.315128
200_off	0.003890	0.317051	0.353517
300_on	0.107780	0.288532	0.317306
300_off	0.003652	0.342651	0.382134

The best performance is when using 100 hidden units and early stopping is on.

3. Testing

For a given sgd method, I want to choose a combination of parameters from weight decay, number of hidden units and early stopping.

For classical momentum, the result is le-1, 50 and True.

For Nesterov, the result is le-1, 50 and True.

And Nesterov has an overall better performance than classical momentum.

```
{'loss_train': 0.490760787258209,
 'loss_valid': 0.5490871550592348,
 'loss_test': 0.5484979732461879}
```