Lab1
back propagation

Outline

- back propagation
- dataset
- code

Outline

- back propagation
- dataset
- code

back propagation

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad j = 1,\ldots,M$$
$$z_j = h(a_j)$$

Hidden layer

$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}^{(1)}} = \delta_j z_i$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \quad k = 1,\ldots,K$$
$$y_k = a_k$$

Output layer

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

$$E_n(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2$$

Error function

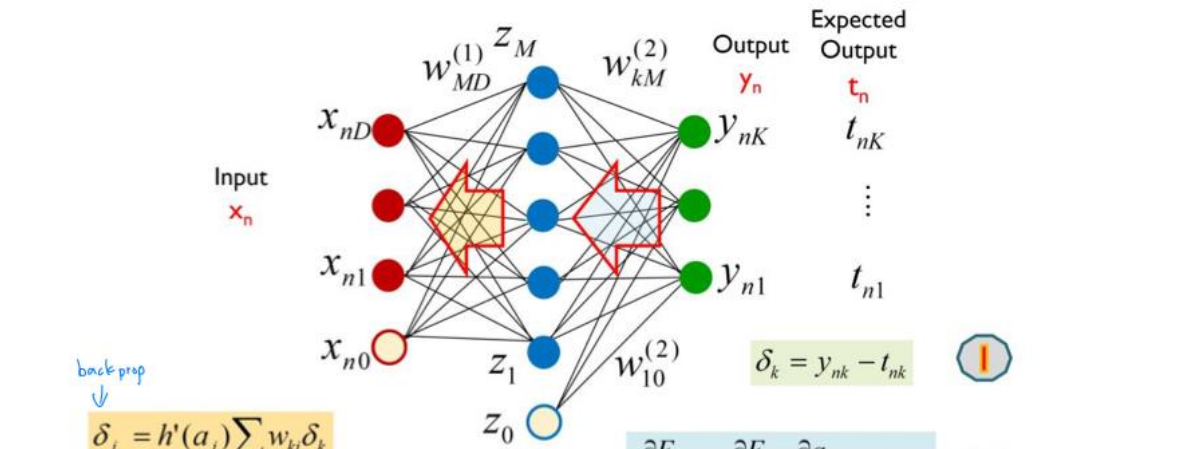
$$\delta_k = y_{nk} - t_{nk}$$

$$\theta = \theta - \eta\frac{\partial L}{\partial \theta}$$

沿著 gradient 的反方向走

Learning rate

More visually, to see the *backpropgation*, we have



back prop

$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$

3

$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}^{(1)}} = \delta_j z_i$

要算第1層所有 weight 之 gradient

$\delta_k = y_{nk} - t_{nk}$ 1

$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}^{(2)}} = \delta_k z_j$ 2

forward 時算出來的

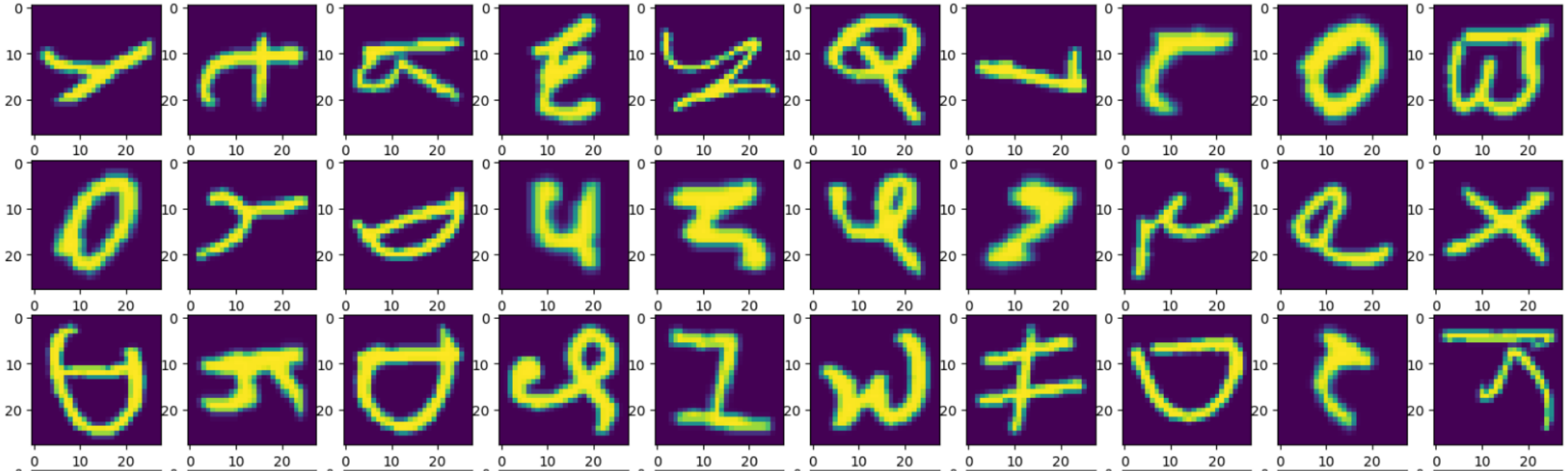
要算第2層所有 weight 之 gradient

Outline

- back propagation
- dataset
- code

dataset

- Classification
- Classes: 47, image size =28*28
- Training: 100000, Testing: 12800



Outline

- back propagation
- dataset
- Code

Code

Just an example.You can alter sample code anywhere.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import model
```

In [2]:
```
#Fix the random seed
np.random.seed(0)
```

Load the training data and label

In [3]:
```
train_load = np.loadtxt('./data/train.csv',delimiter=',',dtype="int")
test_data = np.loadtxt('./data/test.csv',delimiter=',',dtype="int")

train_data=train_load[:,1:]
train_label=train_load[:,0]

print("shape of train_data: {}".format(train_data.shape))
print("shape of train_label: {}".format(train_label.shape))
print("shape of test_data: {}".format(test_data.shape))
```

```
shape of train_data: (100000, 784)
shape of train_label: (100000,)
shape of test_data: (12800, 784)
```

Code

In [6]:
```
val_image_num=8000
```

Convert the training labels to one hot vector

In [7]:
```
label_temp = np.zeros((train_image_num, 47), dtype = np.float32)
for i in range(train_image_num):
    label_temp[i][train_label[i]] = 1
train_label_onehot = np.copy(label_temp)
print("One-hot training labels shape:",train_label_onehot.shape)
```

```
One-hot training labels shape: (100000, 47)
```

Hyperparameters

In [ ]:
```
EPOCH =
Batch_size =
Learning_rate = 6e-4
```

Code

Training

In [15]:
```
net = model.Network()

train_batch_num = (train_image_num  -  val_image_num  )//Batch_size
val_batch_num = (val_image_num)//Batch_size
# test_batch_num = test_image_num//Batch_size

for epoch in range(1, EPOCH+1):
    train_hit = 0
    val_hit = 0
    total_train_loss = 0
    total_val_loss = 0
    for it in range(train_batch_num):
        pred, train_loss = net.forward(train_data[it*Batch_size:(it+1)*Batch_size], train_label_onehot[it*Batch_size:(it+1)*Batch
        pred_index = np.argmax(pred, axis=1)
        train_hit += (pred_index==train_label[it*Batch_size:(it+1)*Batch_size]).sum()
        total_train_loss += train_loss

        net.backward()
        net.update(Learning_rate)

    for titt in range(val_batch_num):
        tit=train_batch_num+titt
        pred, val_loss = net.forward(train_data[tit*Batch_size:(tit+1)*Batch_size], train_label_onehot[tit*Batch_size:(tit+1)*Bat
        pred_index = np.argmax(pred, axis=1)
        val_hit += (pred_index==train_label[tit*Batch_size:(tit+1)*Batch_size]).sum()
        total_val_loss += val_loss

    print('Epoch:%3d'%epoch, '|Train Loss:%8.4f'%(total_train_loss/train_batch_num), '|Train Acc:%3.4f'%(train_hit/(train_image_n
          , '|Val Loss:%8.4f'%(total_val_loss/val_batch_num), '|Val Acc:%3.4f'%(val_hit/val_image_num*100.0))
```

```
Epoch:  1 |Train Loss:  1.4883 |Train Acc:49.7200 |Val Loss:  0.8185 |Val Acc:69.9333
Epoch:  2 |Train Loss:  0.5583 |Train Acc:80.3900 |Val Loss:  0.4845 |Val Acc:80.8667
```

Code

Dump for evaluation (upload your DLAD-test-predict.csv to kaggle)

```
test_pred_list = []

for tit in range(test_image_num//Batch_size):
    pred, test_loss = net.forward(test_data[tit*Batch_size:(tit+1)*Batch_size], train_label_onehot[tit*Batch_size:(tit+1)*Ba
    pred_index = np.argmax(pred, axis=1)
    test_pred_list += pred_index.tolist()


print('Dump file...')
df = pd.DataFrame(test_pred_list, columns=["Category"])
df.to_csv('DLAD-test-predict.csv', index=True, index_label="Id")
```

```
Dump file...
```

Make sure your prediction type is int not float

Code network.py

```
1  from .layer import *
2
3  class Network(object):
4      def __init__(self):
5
6          ## by yourself .Finish your own NN framework
7          ## Just an example.You can alter sample code anywhere.
8          self.fc1 = FullyConnected(28*28, 60) ## Just an example.You can alter sample code anywhere.
9          self.act1 =
10         self.loss
11
12
13
14     def forward(self, input, target):
15         h1 = self.fc1.forward(input)
16         ## by yourself .Finish your own NN framework
17
18         pred, loss =
19
20         return pred, loss
21
22     def backward(self):
23         ## by yourself .Finish your own NN framework
24         h1_grad =
25         _ = self.fc1.backward(h1_grad)
26
27     def update(self, lr):
28         ## by yourself .Finish your own NN framework
29
30         self.fc1.weight -=
31         self.fc1.bias -=
```

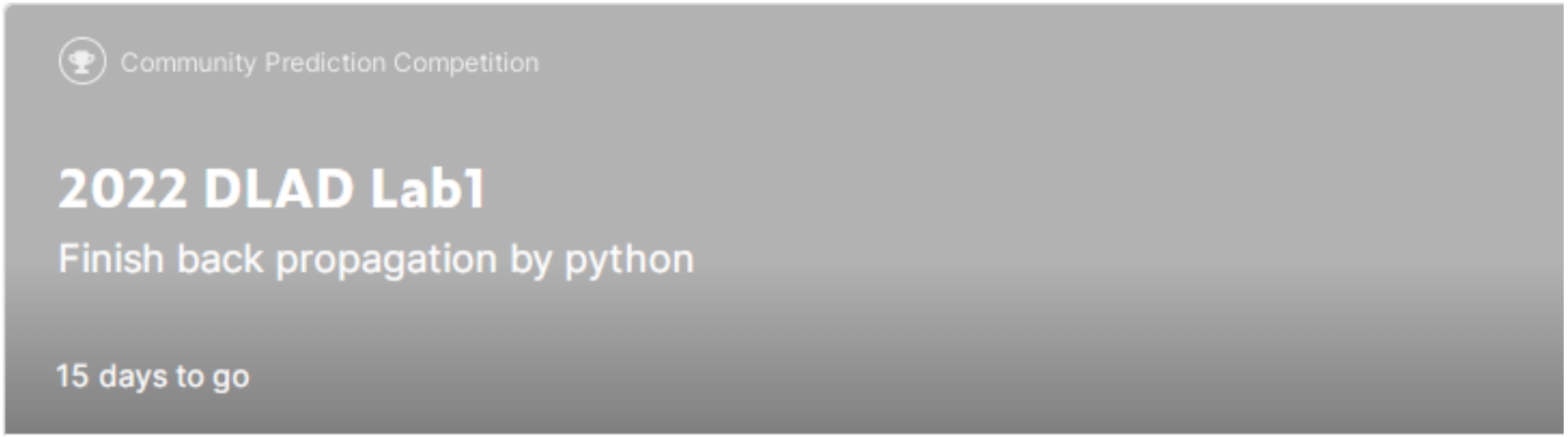
Code layer.py

```python
import numpy as np

## by yourself .Finish your own NN framework
## Just an example.You can alter sample code anywhere.


class _Layer(object):
    def __init__(self):
        pass

    def forward(self, *input):
        r"""Define the forward propagation of this layer.

        Should be overridden by all subclasses.
        """
        raise NotImplementedError

    def backward(self, *output_grad):
        r"""Define the backward propagation of this layer.

        Should be overridden by all subclasses.
        """
        raise NotImplementedError

## by yourself .Finish your own NN framework
class FullyConnected(_Layer):
    def __init__(self, in_features, out_features):
        self.weight = np.random.randn(in_features, out_features) * 0.01
        self.bias = np.zeros((1, out_features))

        self.weight_grad =
        self.bias_grad =

    def forward(self, input):


        return output

    def backward(self, output_grad):
        input_grad =
        self.weight_grad =
        self.bias_grad =

        return input_grad
```

```python
## by yourself .Finish your own NN framework
class ACTIVITY1(_Layer):
    def __init__(self):
        pass

    def forward(self, input):




        return output

    def backward(self, ):



        return

class SoftmaxWithloss(_Layer):
    def __init__(self):
        pass

    def forward(self, input, target):


        '''Softmax'''




        '''Average loss'''


        return predict, your_loss

    def backward(self):
        input_grad

        return input_grad
```

Join the Kaggle competition

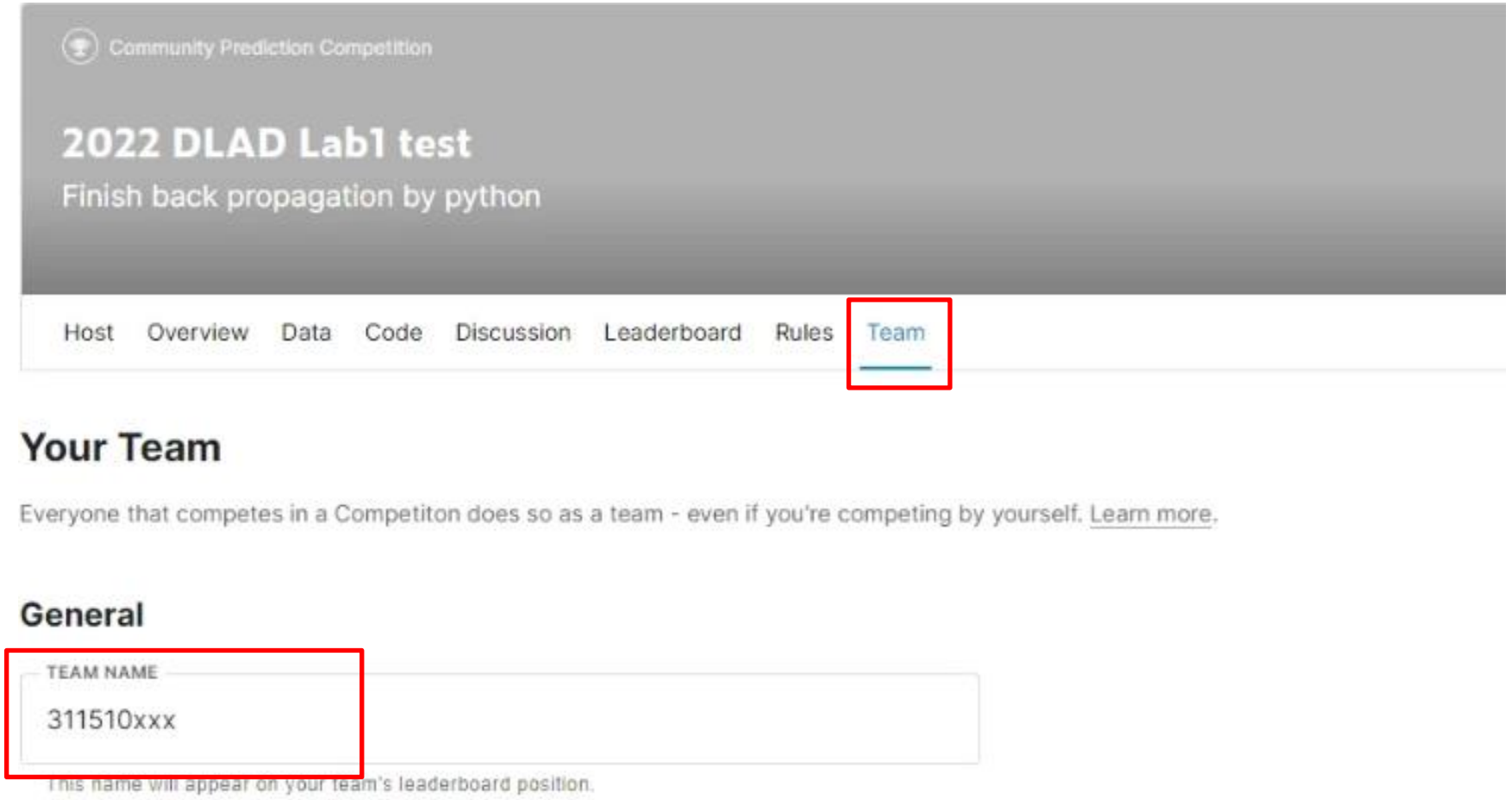
- Competition URL:
  - https://www.kaggle.com/t/ae53e3f2d84b45acb2f1ec95e5b0fcee



- Upload your test prediction
- Check your team name

Notice!!!

- Change your team name to your ID
  - If TA can not find your ID, you will loss 40% of score in this lab directly



Assignment Regulation

- Please use jupyter notebook to finish this lab (.ipynb)
- Only Python3.6 available
- You don’t need to use GPU in this lab
- Package available
  - numpy
  - pandas
  - ~~Scikit-learn~~
  - ~~Pytorch ,tensorflow , keras~~

Reminder

- Submit Deadline : 2 weeks (06日/10月/2022 11:59 PM)
- You need to submit your **code and result** to New E3

- Hand in your code and in the following format(4 files)
  - Lab1_studentid.ipynb
  - network.py
  - layer.py
  - Lab1_report_studentid.pdf

Lab1 Grading policy

- Lab1
  - Submit your homework to E3 and kaggle(40%),test accuracy must >80%
  - performance (40%)
  - report(20%)
- 請勿抄襲 (抓到以0分計算)

Lab1 report

- How to improve the accuracy (list your method)
  - Loss function ?
  - Your network?
  - Activation function?
  - Etc…

Shut down your kernel !!!!!

- Remember to SHUTDOWN the kernel
- To release resources for others.

Lab0 Overview

- You need to be familiar with jupyter python and useful library
  - numpy, pandas, matplot, etc

- If you are not familiar with python, I strongly recommend you to see 莫凡's python tutorial. (or other online materials)
  - https://morvanzhou.github.io/