Lab1 : back-propagation

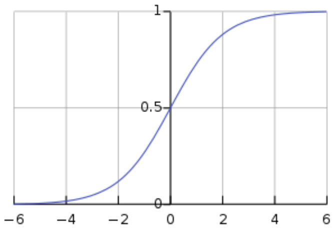
0516069 翁英傑

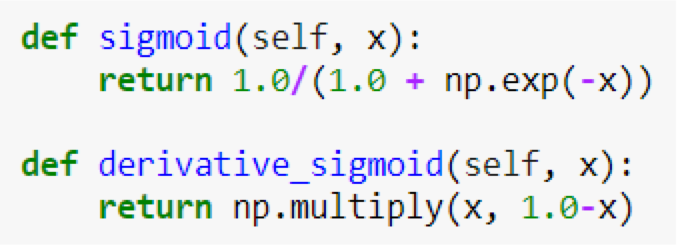
1. Introduction

In the training stage of a neural network, it relies on a technic called gradient decent. While calculating the gradient of the loss regarding to each weight, using back propagation will ease the computation, which the gradient is calculate from the output layer to the input layer.

2. Experiment setups

A. Sigmoid functions





```
def sigmoid(self, x):
    return 1.0/(1.0 + np.exp(-x))

def derivative_sigmoid(self, x):
    return np.multiply(x, 1.0-x)
```

Sigmoid function is used for making each layer none linear so that it’s meaningful for to create a multi-hidden layer network.

B. Neural network

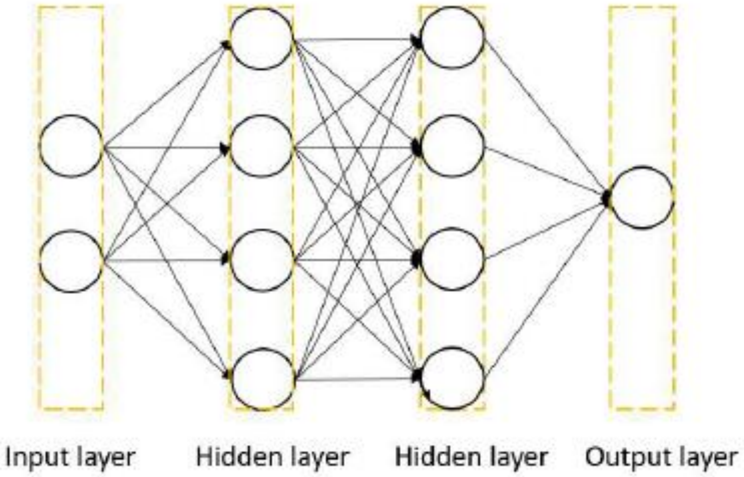
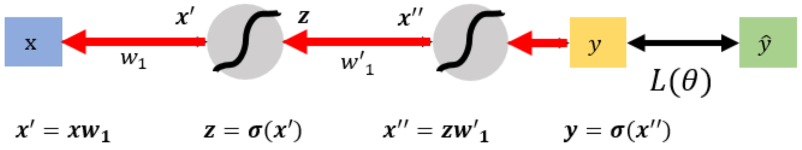


Figure 1. Two layers neural network

A picture of a network with two hidden layers, which each layer contains four nodes.

C. Backpropagation



Chain rule

$y = g(x) \quad z = h(y)$

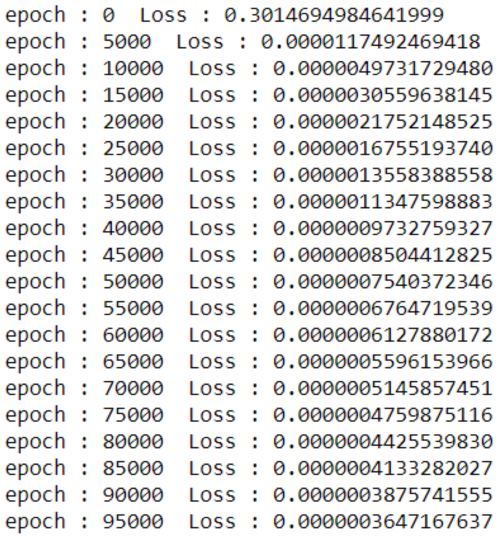
$\mathbf{x} \xrightarrow{g()} \boldsymbol{y} \xrightarrow{h()} \boldsymbol{z} \qquad \frac{dz}{dx} = \frac{dz}{\cancel{dy}}\frac{\cancel{dy}}{dx}$

$$\begin{aligned}\frac{\partial L(\theta)}{\partial w_1} &= \frac{\partial y}{\partial w_1}\frac{\partial L(\theta)}{\partial y}\\ &= \frac{\partial x''}{\partial w_1}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}\\ &= \frac{\partial z}{\partial w_1}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}\\ &= \frac{\partial x'}{\partial w_1}\frac{\partial z}{\partial x'}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}\end{aligned}$$

With the chain rule, if we calculate the gradient beginning from the output layer, the previous gradient will be used in the computation of the next gradient.

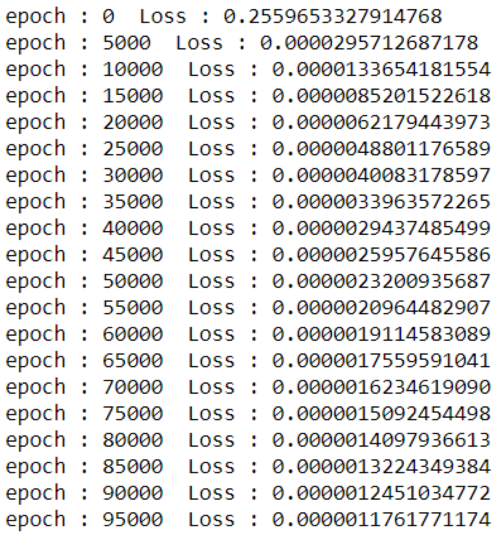
3. Results of your testing

1. Linear distribution

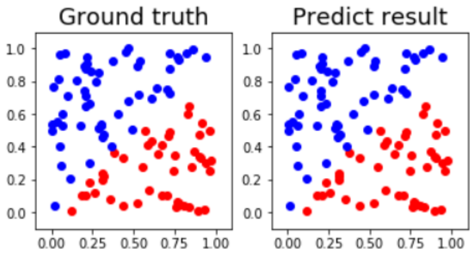


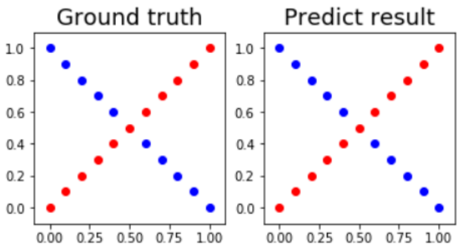
```
epoch : 0  Loss : 0.3014694984641999
epoch : 5000  Loss : 0.0000117492469418
epoch : 10000  Loss : 0.0000049731729480
epoch : 15000  Loss : 0.0000030559638145
epoch : 20000  Loss : 0.0000021752148525
epoch : 25000  Loss : 0.0000016755193740
epoch : 30000  Loss : 0.0000013558388558
epoch : 35000  Loss : 0.0000011347598883
epoch : 40000  Loss : 0.0000009732759327
epoch : 45000  Loss : 0.0000008504412825
epoch : 50000  Loss : 0.0000007540372346
epoch : 55000  Loss : 0.0000006764719539
epoch : 60000  Loss : 0.0000006127880172
epoch : 65000  Loss : 0.0000005596153966
epoch : 70000  Loss : 0.0000005145857451
epoch : 75000  Loss : 0.0000004759875116
epoch : 80000  Loss : 0.0000004425539830
epoch : 85000  Loss : 0.0000004133282027
epoch : 90000  Loss : 0.0000003875741555
epoch : 95000  Loss : 0.0000003647167637
```

2. XOR distribution



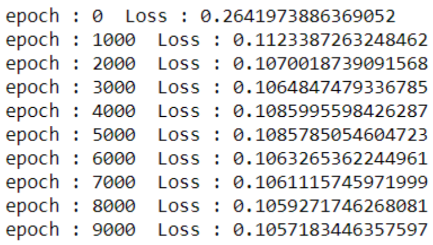
```
epoch : 0  Loss : 0.2559653327914768
epoch : 5000  Loss : 0.0000295712687178
epoch : 10000  Loss : 0.0000133654181554
epoch : 15000  Loss : 0.0000085201522618
epoch : 20000  Loss : 0.0000062179443973
epoch : 25000  Loss : 0.0000048801176589
epoch : 30000  Loss : 0.0000040083178597
epoch : 35000  Loss : 0.0000033963572265
epoch : 40000  Loss : 0.0000029437485499
epoch : 45000  Loss : 0.0000025957645586
epoch : 50000  Loss : 0.0000023200935687
epoch : 55000  Loss : 0.0000020964482907
epoch : 60000  Loss : 0.0000019114583089
epoch : 65000  Loss : 0.0000017559591041
epoch : 70000  Loss : 0.0000016234619090
epoch : 75000  Loss : 0.0000015092454498
epoch : 80000  Loss : 0.0000014097936613
epoch : 85000  Loss : 0.0000013224349384
epoch : 90000  Loss : 0.0000012451034772
epoch : 95000  Loss : 0.0000011761771174
```



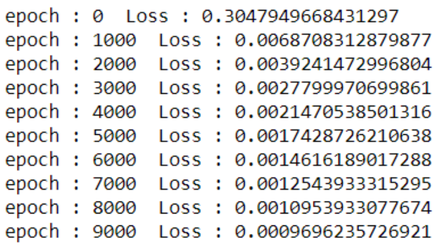


4. Discussion

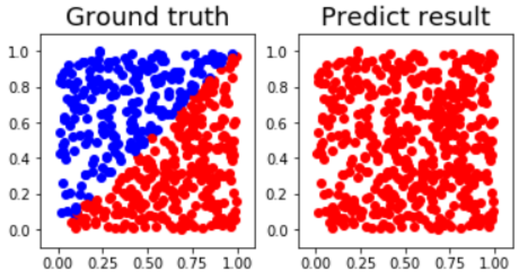
A common tip in training is to add a learning rate to avoid the overshooting problem. In the basic setting of this LAB, which generates 100 data samples by default setting, there is no problem with gradient decent. But if we generate over 200 data samples, the gradient will be too large that it crosses the lowest point and the loss does not decrease. Adding a learning rate solves the problem.



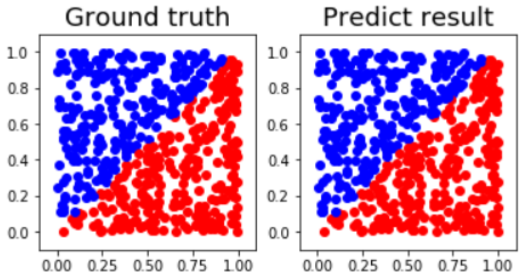
```
epoch : 0  Loss : 0.2641973886369052
epoch : 1000  Loss : 0.1123387263248462
epoch : 2000  Loss : 0.1070018739091568
epoch : 3000  Loss : 0.1064847479336785
epoch : 4000  Loss : 0.1085995598426287
epoch : 5000  Loss : 0.1085785054604723
epoch : 6000  Loss : 0.1063265362244961
epoch : 7000  Loss : 0.1061115745971999
epoch : 8000  Loss : 0.1059271746268081
epoch : 9000  Loss : 0.1057183446357597
```



```
epoch : 0  Loss : 0.3047949668431297
epoch : 1000  Loss : 0.0068708312879877
epoch : 2000  Loss : 0.0039241472996804
epoch : 3000  Loss : 0.0027799970699861
epoch : 4000  Loss : 0.0021470538501316
epoch : 5000  Loss : 0.0017428726210638
epoch : 6000  Loss : 0.0014616189017288
epoch : 7000  Loss : 0.0012543933315295
epoch : 8000  Loss : 0.0010953933077674
epoch : 9000  Loss : 0.0009696235726921
```



Learnig rate = 1



Learning rate = 0.01