NYCU DL
Lab1 - Backpropagation

TA 高宗霖

March 21, 2023

Outline

- Recap

- Lab Objective

- Lab Description

- Scoring Criteria

- Importance Date

- Demo format

Recap

Recap - Build a Network

● Model

```python
class simple(nn.Module):
    def __init__(self, num_hidden):
        self.layer1 = nn.Sequential(
            nn.Linear(2, num_hidden),
            nn.Sigmoid()
        )

        self.layer2 = nn.Sequential(
            nn.Linear(num_hidden, num_hidden),
            nn.Sigmoid()
        )

        self.flatten = nn.Linear(num_hidden, 1)

    def forward(self, x):
        h1 = self.layer1(x)
        h2 = self.layer2(h1)
        out = self.flatten(h2)
        return out
```
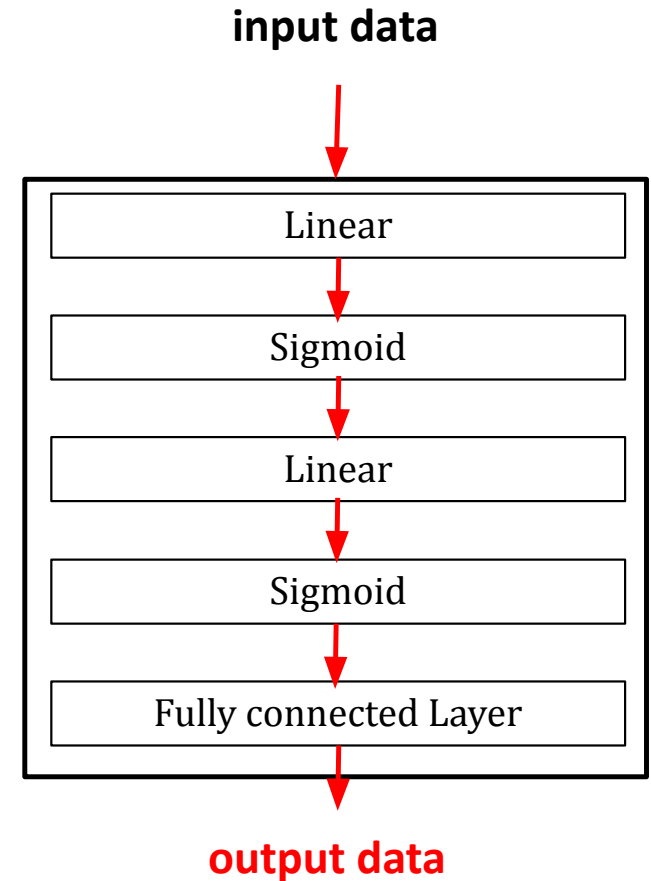
| Linear |
| Sigmoid |
| Linear |
| Sigmoid |
| Fully connected Layer |

Recap - Forward

● Model

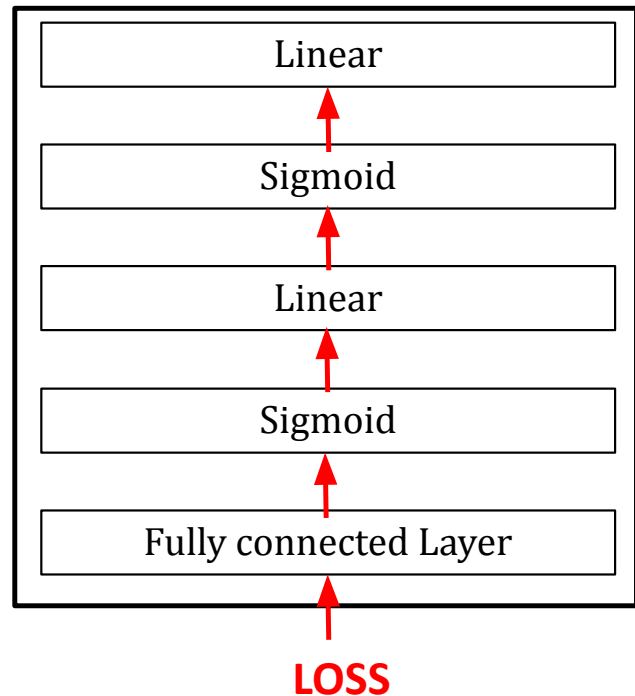```python
class simple(nn.Module):
    def __init__(self, num_hidden):
        self.layer1 = nn.Sequential(
            nn.Linear(2, num_hidden),
            nn.Sigmoid()
)

        self.layer2 = nn.Sequential(
            nn.Linear(num_hidden, num_hidden),
            nn.Sigmoid()
)

        self.flatten = nn.Linear(num_hidden, 1)

    def forward(self, x):
        h1 = self.layer1(x)
        h2 = self.layer2(h1)
        out = self.flatten(h2)
        return out
```

**input data**

| Linear |
| --- |

| Sigmoid |
| --- |

| Linear |
| --- |

| Sigmoid |
| --- |

| Fully connected Layer |
| --- |

**output data**

# Recap - Backpropagation

- Model

```python
class simple(nn.Module):
    def __init__(self, num_hidden):
        self.layer1 = nn.Sequential(
            nn.Linear(2, num_hidden),
            nn.Sigmoid()
        )

        self.layer2 = nn.Sequential(
            nn.Linear(num_hidden, num_hidden),
            nn.Sigmoid()
        )

        self.flatten = nn.Linear(num_hidden, 1)

    def forward(self, x):
        h1 = self.layer1(x)
        h2 = self.layer2(h1)
        out = self.flatten(h2)
        return out
```
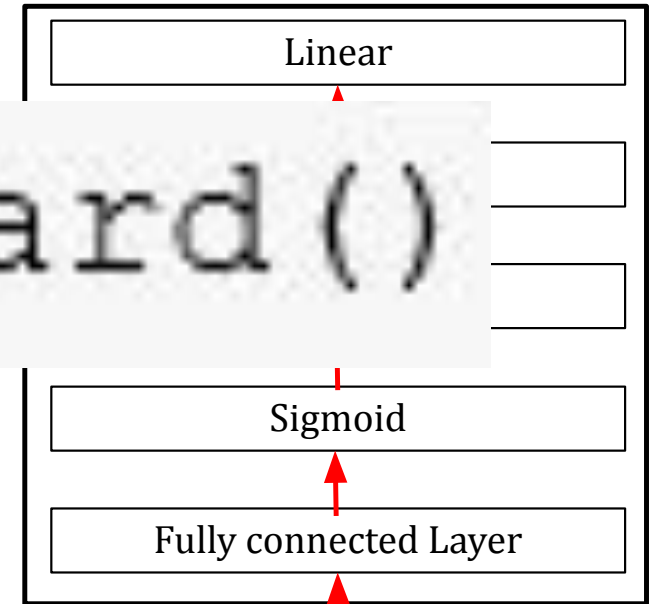
Linear
↑
Sigmoid
↑
Linear
↑
Sigmoid
↑
Fully connected Layer
↑

LOSS

Recap - Backpropagation

● Model

```python
class simple(nn.Module):
    def __init__(self, num_hidden):
        self.layer1 = nn.Sequential(
            nn.Linear(2, num_hidden),
            nn.Sigmoid()
        )
```

loss.backward()

```python
    def forward(self, x):
        h1 = self.layer1(x)
        h2 = self.layer2(h1)
        out = self.flatten(h2)
        return out
```

Linear

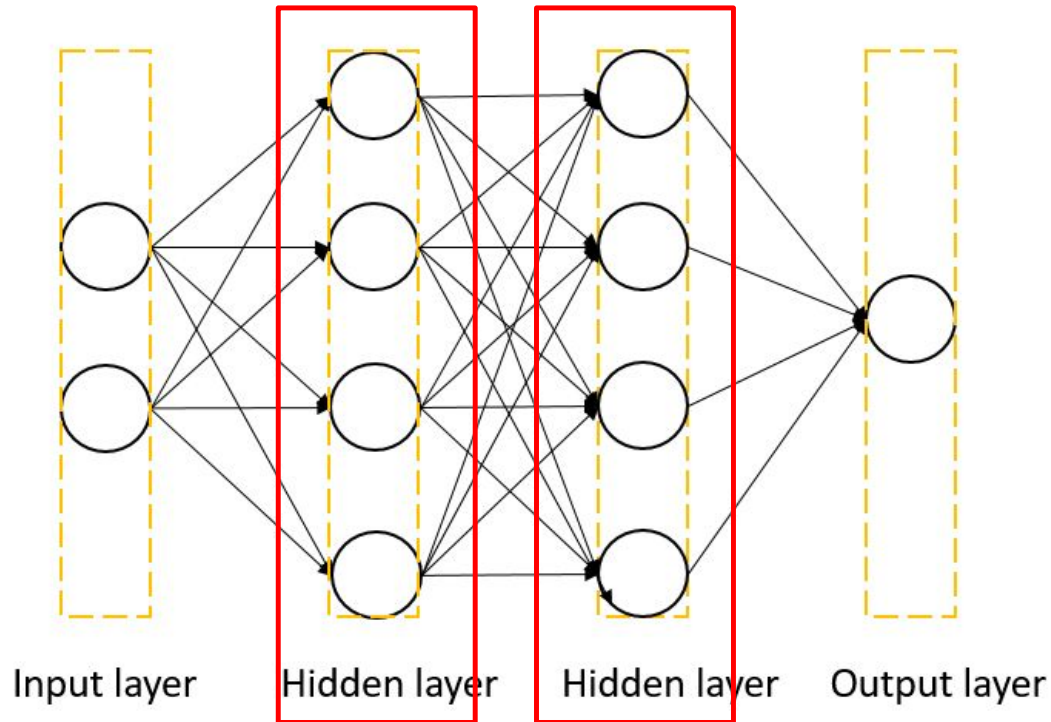| Sigmoid |
| --- |

Fully connected Layer

LOSS

Lab Objective

Lab Objective

● In this lab, you will need to understand and implement a simple neural network with forward and backward pass using two hidden layers

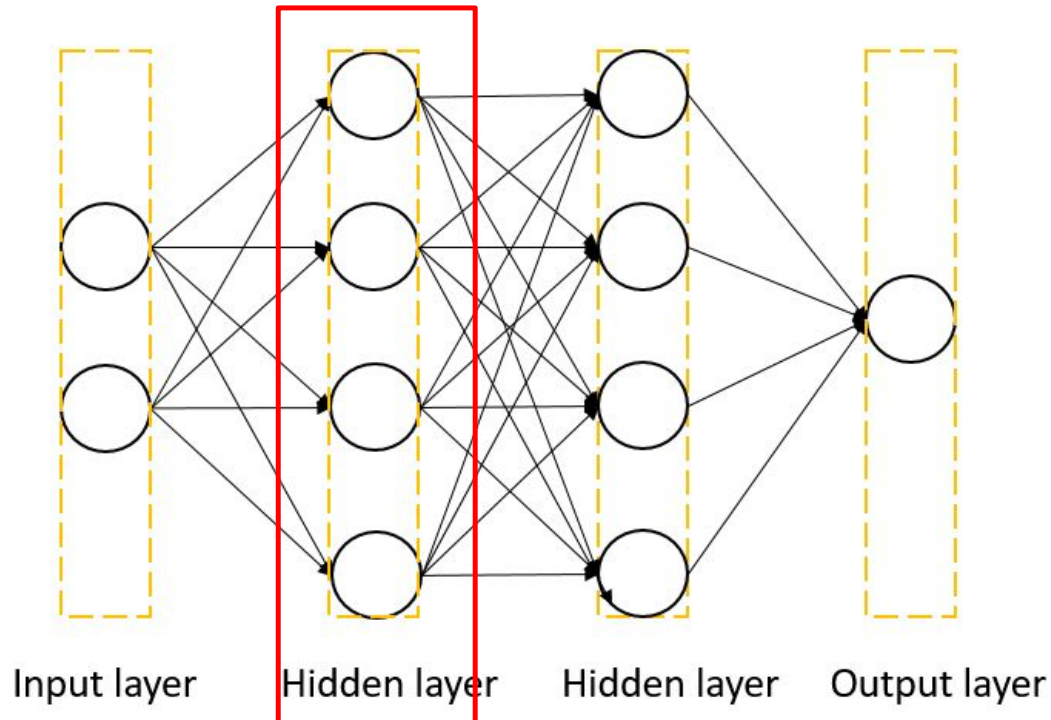Input layer   Hidden layer   Hidden layer   Output layer

Lab Description
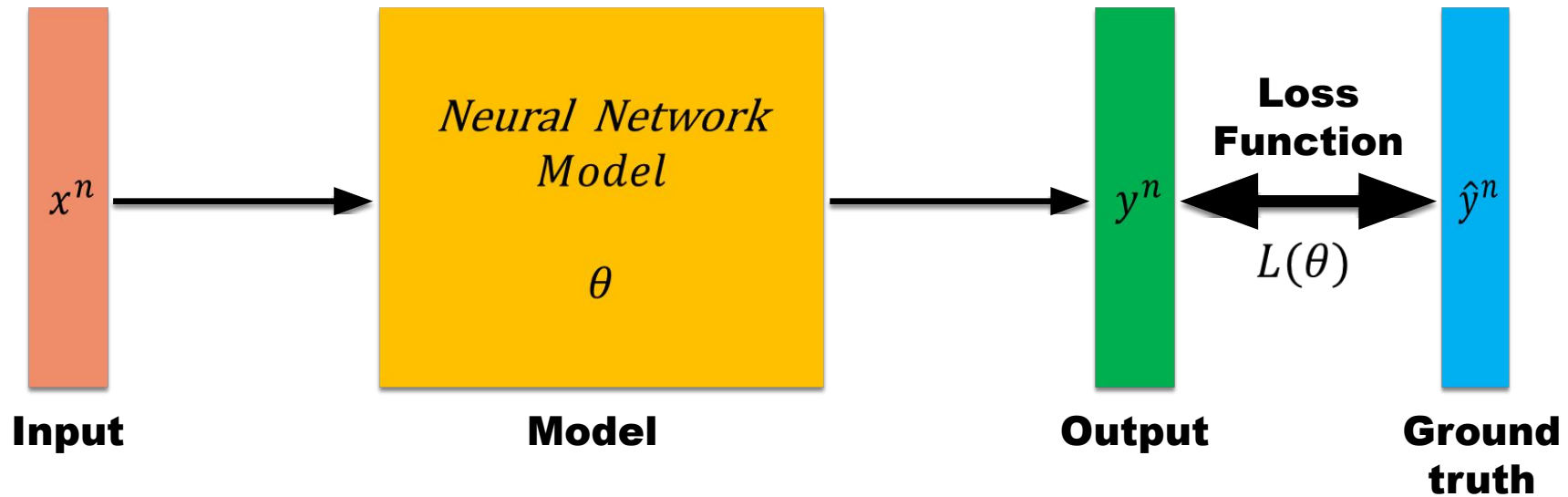
Lab Description

- Implement a simple neural network with two hidden layers

- You can only use **Numpy** and other **python standard libraries**.

  - **Pytorch, TensorFlow,..... are not allowed**

- Plot your comparison figure showing the predictions and ground truth.

- Plot your learning curve (loss, epoch).

- Print the accuracy of your prediction .

- Print the training Loss in the end of the few last epochs.

- **List above results in your report**

Lab Description

● Each Layer should contain at least **one transformation** (e.g. Linear, CNN,...) and **one activation function** (e.g. sigmoid, tanh….)

Input layer   Hidden layer   Hidden layer   Output layer

Lab Description

Input   **Model**   **Output**   **Ground truth**

$$\theta = \{w_1, w_2, w_3, w_4, \cdots\}$$

$$\boldsymbol{\nabla L(\theta)} = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \partial L(\theta)/\partial w_3 \\ \vdots \\ \vdots \end{bmatrix}$$

Compute $\nabla L(\theta^0)$ $\theta^1 = \theta^0 - \rho \, \nabla L(\theta^0)$

Compute $\nabla L(\theta^1)$ $\theta^2 = \theta^1 - \rho \, \nabla L(\theta^1)$

Compute $\nabla L(\theta^2)$ $\theta^3 = \theta^2 - \rho \, \nabla L(\theta^2)$

ρ : Learning rate

Lab Description – Flowchart

Prepare data

↓

Create model

↓

Forward pass

↓

Backward pass

↓

Update weights

Training

Repeat n times
or
until convergence

14

Lab Description - Data

generate_linear()

generate_XOR_easy()
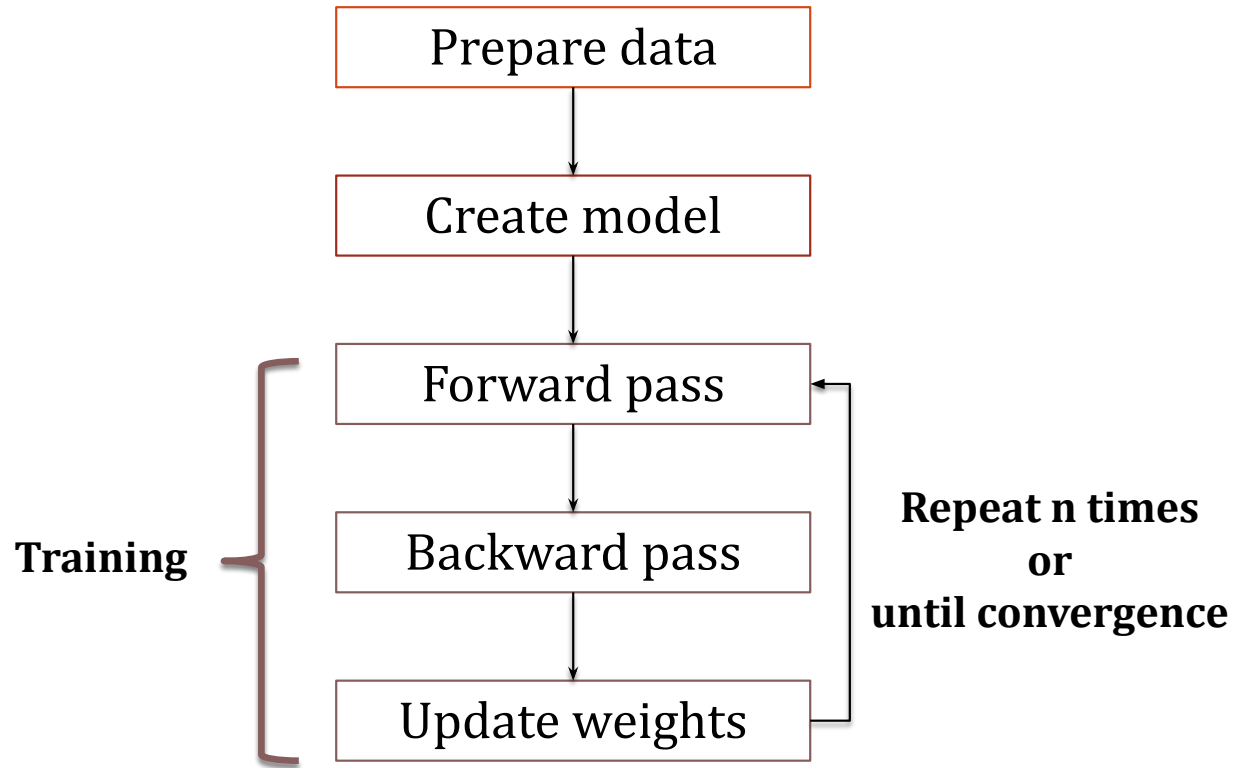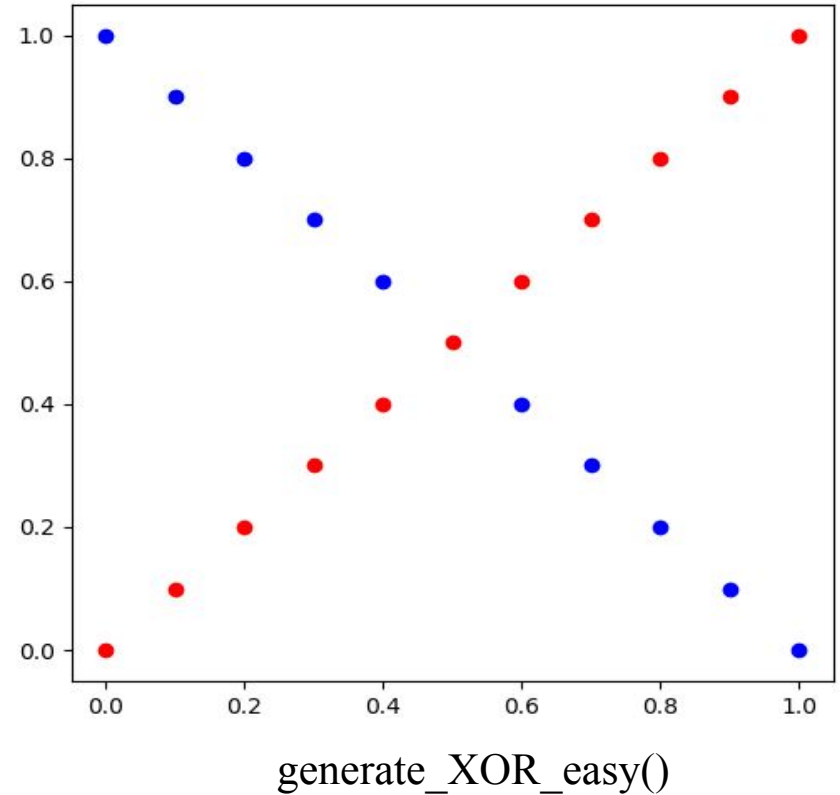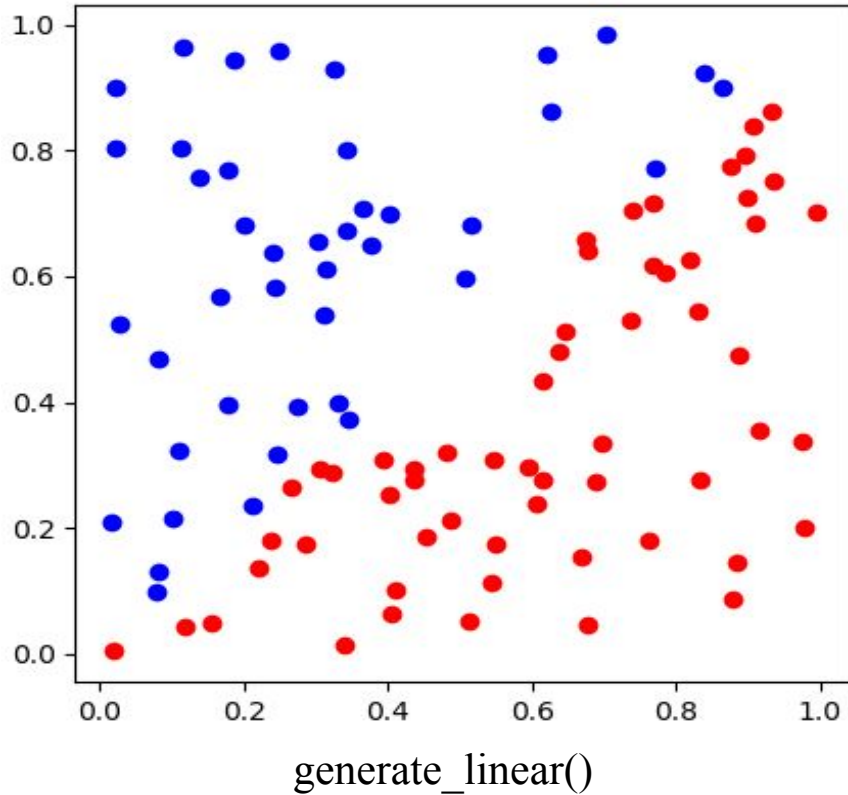
Lab Description – Architecture
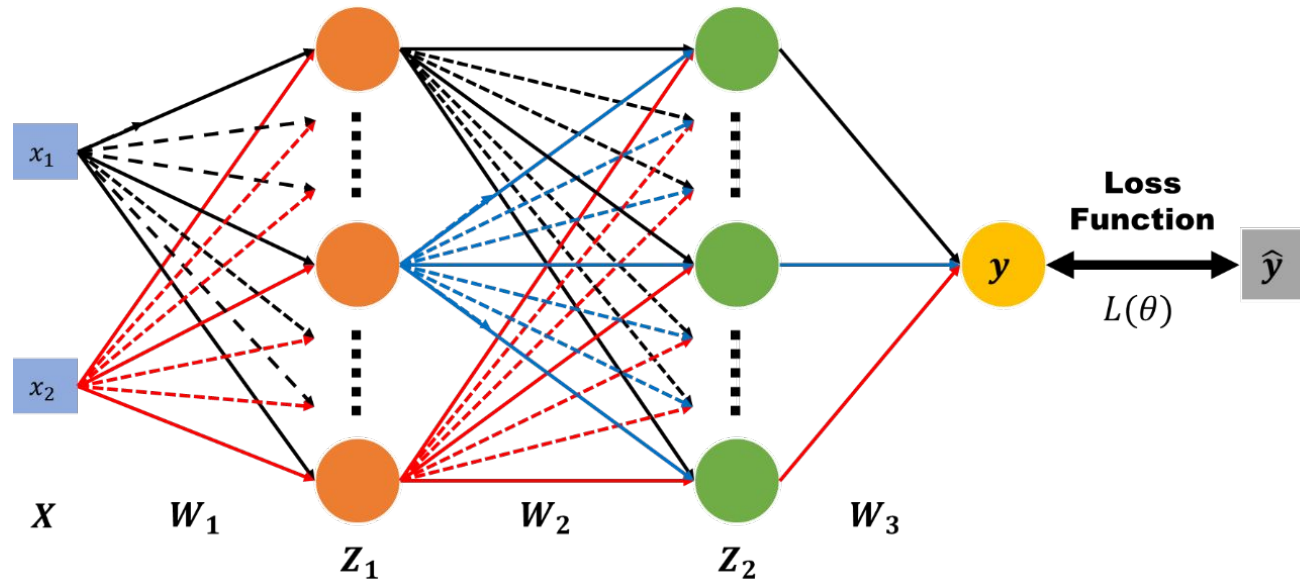
$$X : [x_1, x_2] \qquad y : outputs \qquad \widehat{y} : ground\ truth$$

$$W_1, W_2, W_3 : weight\ matrix\ of\ network\ layers$$

Lab Description – Forward
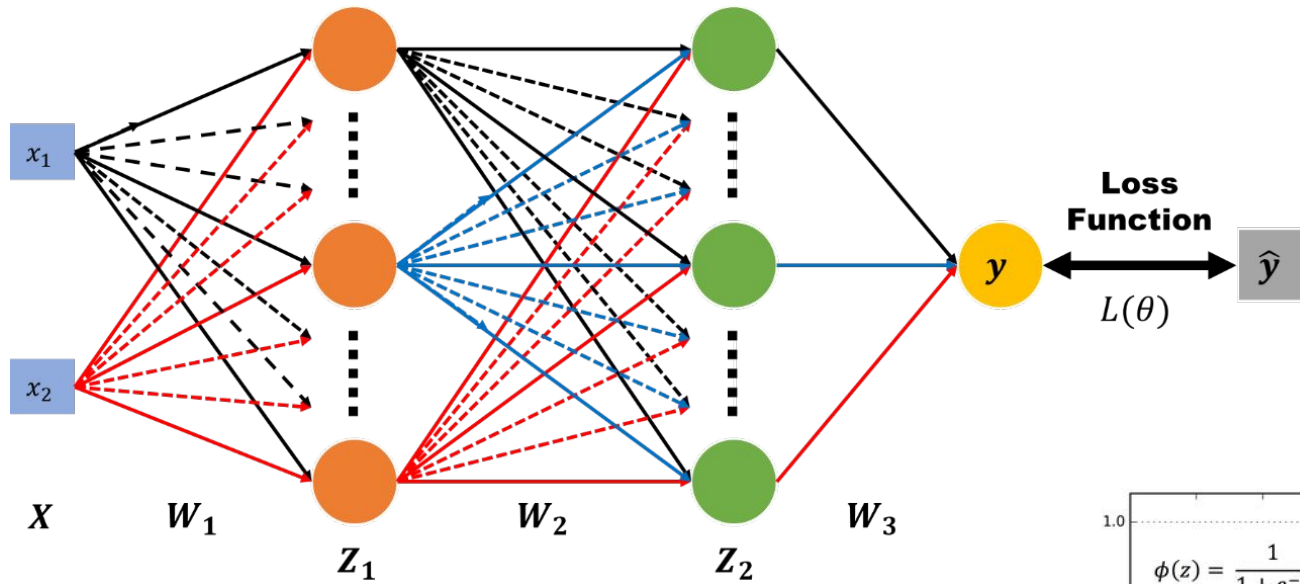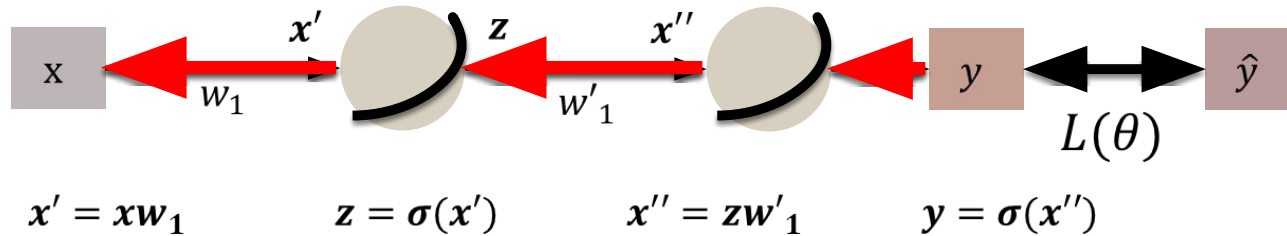

$$Z_1 = \sigma(XW_1) \qquad Z_2 = \sigma(Z_1W_2) \qquad y = \sigma(Z_2W_3)$$

$$\sigma(\text{x}) = \frac{1}{1 + e^{-x}}$$

$$\phi(z) = \frac{1}{1 + e^{-z}}$$
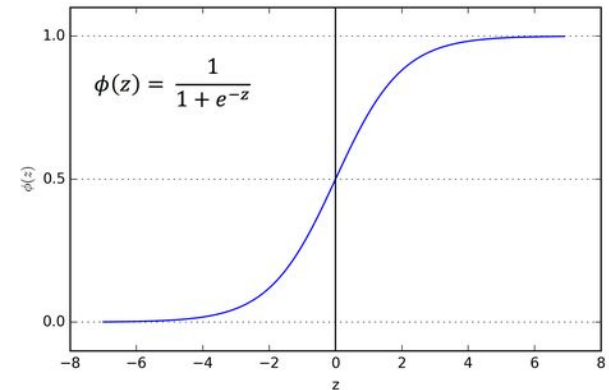
Lab Description – Backward

$$x' = xw_1 \qquad z = \sigma(x') \qquad x'' = zw'_1 \qquad y = \sigma(x'')$$
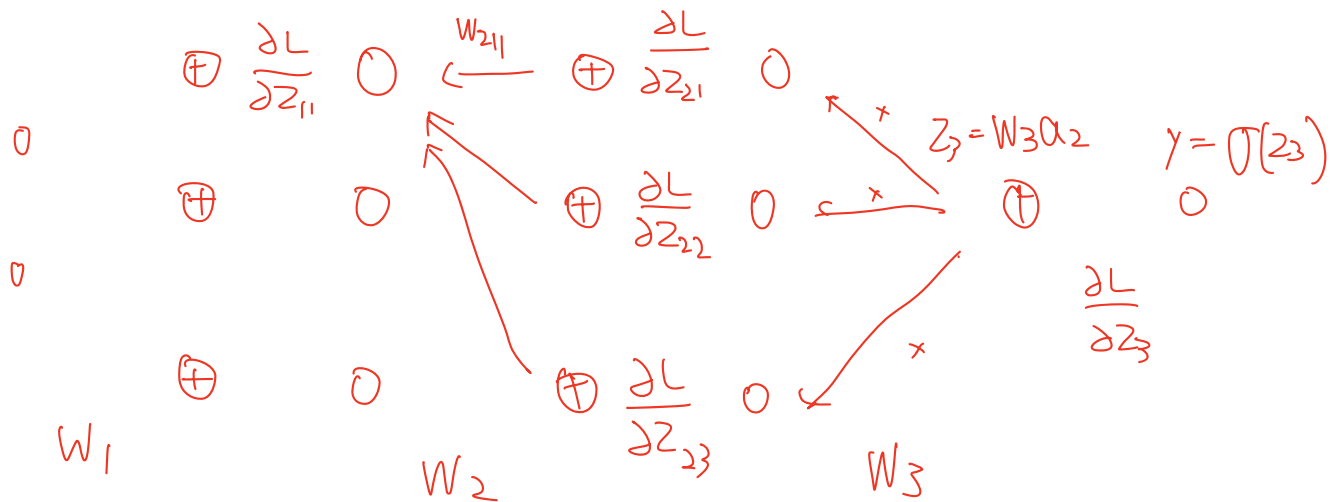
Chain rule

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

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

$$
\begin{aligned}
\frac{\partial L(\theta)}{\partial w_1} &= \frac{\partial y}{\partial w_1}\frac{\partial L(\theta)}{\partial y} \\[4pt]
&= \frac{\partial x''}{\partial w_1}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y} \\[4pt]
&= \frac{\partial z}{\partial w_1}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y} \\[4pt]
&= \frac{\partial x'}{\partial w_1}\frac{\partial z}{\partial x'}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}
\end{aligned}
$$

$$z_1 = (\quad) \quad a_1 = \sigma(z_1) \quad z_2 = W_2 a_1 \quad a_2 = \sigma(z_2)$$

$$\oplus \frac{\partial L}{\partial z_{11}} \quad O \quad \xleftarrow{W_{211}} \quad \oplus \frac{\partial L}{\partial z_{21}} \quad O$$

$$O \qquad\qquad z_3 = W_3 a_2 \quad y = \sigma(z_3)$$

$$\oplus \qquad O \qquad \oplus \frac{\partial L}{\partial z_{22}} \quad O \quad \xrightarrow{\quad\times\quad} \oplus \qquad O$$

$$\frac{\partial L}{\partial z_3}$$

$$O$$

$$\oplus \qquad O \qquad \oplus \frac{\partial L}{\partial z_{23}} \quad O$$

$$W_1 \qquad\qquad W_2 \qquad\qquad W_3$$

$$\frac{\partial L}{\partial z_3} = \frac{\partial y}{\partial z_3}\frac{\partial L}{\partial y} = \sigma'(z_3)\frac{\partial L}{\partial y}$$

$$[W_{31} \; W_{32} \; W_{33}]$$

$$\frac{\partial L}{\partial z_{21}} = \frac{\partial a_{21}}{\partial z_{21}}\frac{\partial L}{\partial a_{21}} = \frac{\partial a_{21}}{\partial z_{21}}\left(\frac{\partial z_3}{\partial a_{21}}\frac{\partial L}{\partial z_3}\right)$$

$$(W_3)_{1\times 3} \; (z_3\_der)_{1\times batchsize}$$

$$\frac{\partial}{\partial y} -\left(\hat{y}\log y + (1-\hat{y})\log(1-y)\right)$$

$$\begin{bmatrix} W_{31} \\ W_{32} \\ W_{33} \end{bmatrix} [z_3\_der$$

$$= -\left(\frac{\hat{y}}{y} + (1-\hat{y})\frac{-1}{1-y}\right) = \frac{-\hat{y}}{y} + \frac{1-\hat{y}}{1-y}$$

$$(W_2)_{\ell_2 \times \ell_1}$$

$$\ell_1\begin{bmatrix} z_1\_der \\ \text{batchsize} \end{bmatrix} = \ell_1\begin{bmatrix} W_2 \\ \ell_2 \end{bmatrix} \; \ell_2\begin{bmatrix} z_2\_der \\ \text{batchsize} \end{bmatrix}$$

$$\frac{\partial L}{\partial W_{31}} = \frac{\partial z_3}{\partial W_{31}}\frac{\partial L}{\partial z_3} = a_{21}\frac{\partial L}{\partial z} \quad \ell_2\begin{bmatrix} a_{21} \\ a \\ \end{bmatrix} 1\begin{bmatrix} \quad \end{bmatrix}$$

$$A_2 \quad batchsize \qquad z_3\_der \quad batchsize$$

$$\frac{\partial L}{\partial W_{211}} = \frac{\partial Z_{21}}{\partial W_{211}} \frac{\partial L}{\partial Z_{21}} = a_{11} \frac{\partial L}{\partial Z_{21}} \qquad |W_2 =$$

$$\frac{\partial L}{\partial b_3} = \frac{\partial Z_3}{\partial b_3} \frac{\partial L}{\partial Z_3}$$
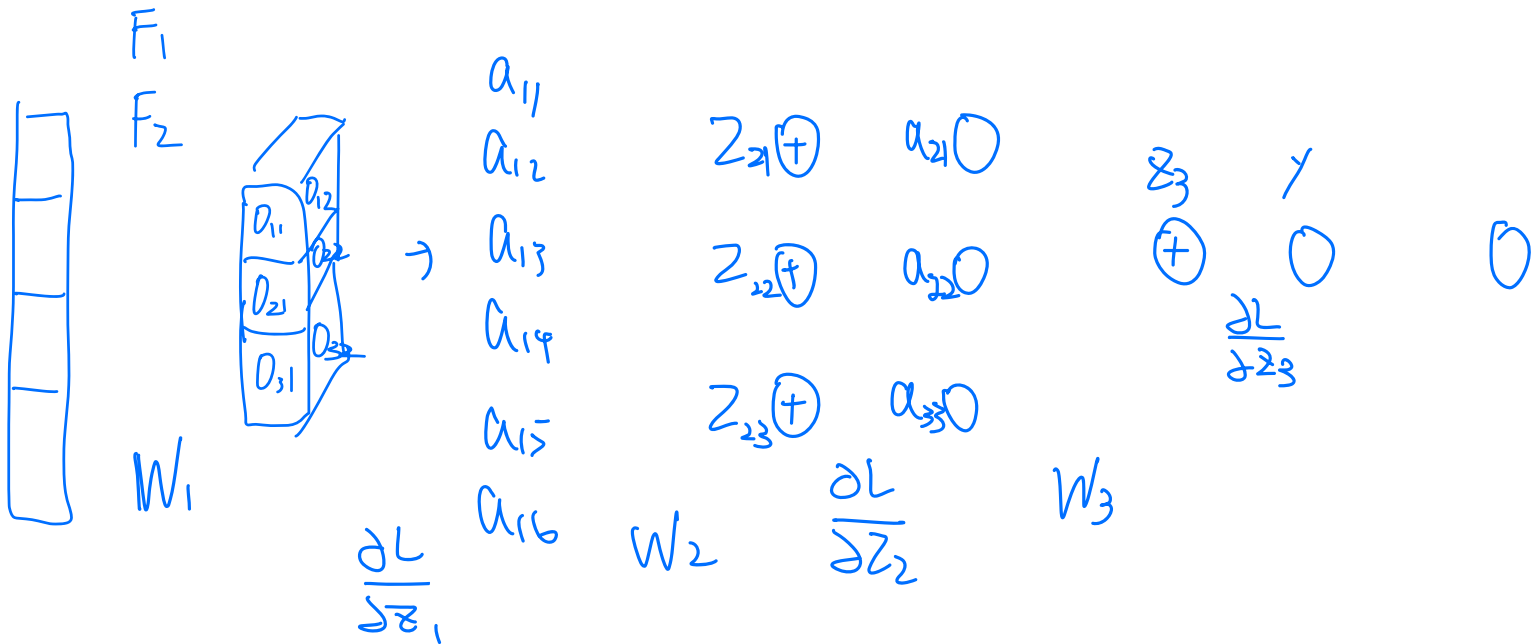
$$A_1 \text{ batch-size} \qquad Z2\_der \text{ batch-size}$$

$$l_1 \begin{bmatrix} a_{11} \\ a_{12} \\ \vdots \end{bmatrix} \qquad l_2 \begin{bmatrix} \end{bmatrix} \qquad \frac{\partial L}{\partial b_{21}} = \frac{\partial Z_{21}}{\partial b_{21}} \frac{\partial L}{\partial Z_{21}}$$

F_1

F_2

a_{11}

a_{12}

a_{13}

a_{14}

a_{15}

a_{16}

$Z_{21} (+) \quad a_{21} \bigcirc$

$Z_{22} (+) \quad a_{22} \bigcirc$

$Z_{23} (+) \quad a_{33} \bigcirc$

$Z_3 \qquad Y$

$(+) \qquad \bigcirc \qquad \bigcirc$

$\frac{\partial L}{\partial Z_3}$

W_1

$W_2 \qquad \frac{\partial L}{\partial Z_2}$

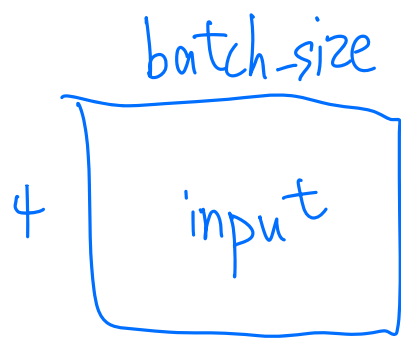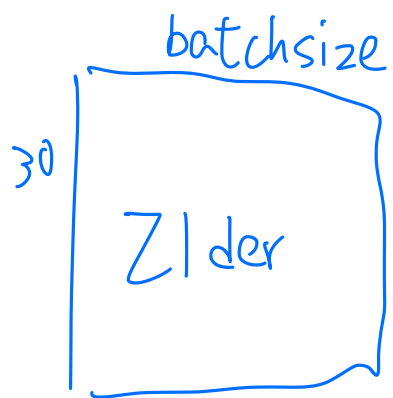$\frac{\partial L}{\partial Z_1}$

W_3

$$\sum_i \frac{\partial O_i}{\partial F} \frac{\partial L}{\partial O_i} \qquad \frac{\partial L}{\partial O_i} = \frac{\partial a_i}{\partial O_i} \frac{\partial L}{\partial a_i} = \frac{\partial a_i}{\partial O_i} \frac{\partial Z}{\partial a} \frac{\partial L}{\partial Z}$$

$$\frac{\partial L}{\partial F_{11}} = \sum_i \frac{\partial O_{1i}}{\partial F_{11}} \frac{\partial L}{\partial O_{1i}} \quad , \quad \frac{\partial L}{\partial O_{1i}} = \frac{\partial a_{1i}}{\partial O_{1i}} \frac{\partial L}{\partial a_{1i}} = \frac{\partial a_{1i}}{\partial O_{1i}}$$

$$= \frac{\partial O_{11}}{\partial F_{11}} \frac{\partial L}{\partial Z_{11}} + \frac{\partial O_{21}}{\partial F_{11}} \frac{\partial L}{\partial Z_{12}} + \frac{\partial O_{31}}{\partial F_{11}} \frac{\partial L}{\partial Z_{13}}$$
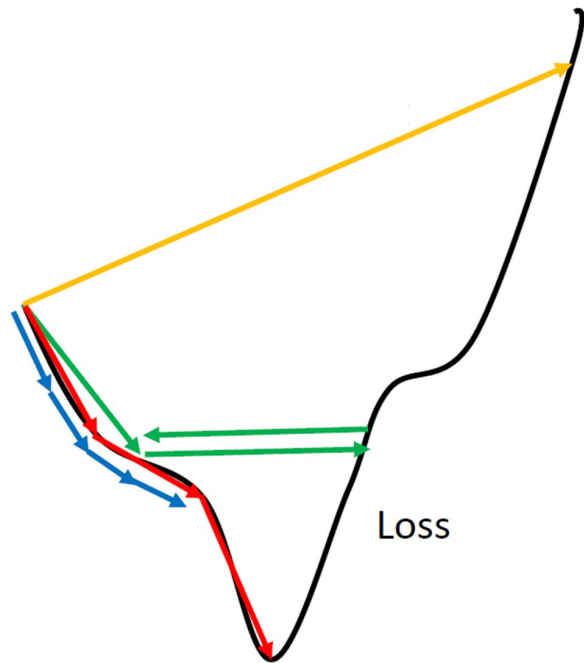
$$= x_1 \frac{\partial L}{\partial Z_{11}} + x_2 \frac{\partial L}{\partial Z_{12}} + x_3 \frac{\partial L}{\partial Z_{13}}$$

batchsize

30 | Z1der

batch-size

4 | input

10 W_1
1_1
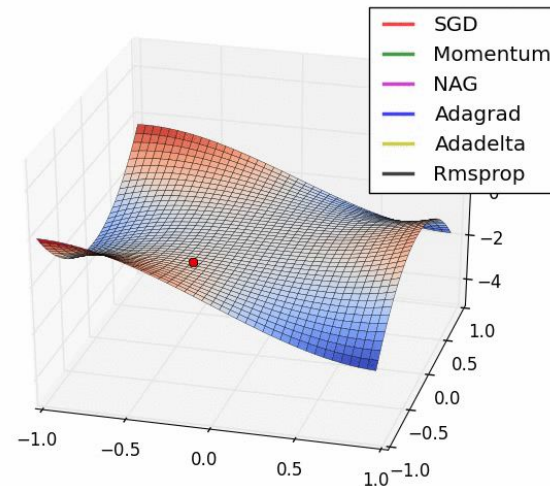
$$\frac{\partial L}{\partial b_{11}} =$$

∂ | ∂

Lab Description – Gradient descent

Network Parameters $\theta = \{w_1, w_2, w_3, w_4, \cdots\}$

$$\theta^1 = \theta^0 - \rho\, \nabla L(\theta^0)$$

$$\theta^2 = \theta^1 - \rho\, \nabla L(\theta^1)$$

$$\theta^3 = \theta^2 - \rho\, \nabla L(\theta^2)$$

ρ : Learning rate

Loss

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

19

Lab Description - Prediction

- In the training, you need to print loss
- In the testing, you need to show your predictions, also the accuracy

```
epoch 10000 loss : 0.16234523253277644
epoch 15000 loss : 0.2524336634177614
epoch 20000 loss : 0.1590783047540092
epoch 25000 loss : 0.22099447030234853
epoch 30000 loss : 0.3292173477217561
epoch 35000 loss : 0.40406233282426085
epoch 40000 loss : 0.43052897480298924
epoch 45000 loss : 0.4207525735586605
epoch 50000 loss : 0.39347595093424279
epoch 55000 loss : 0.3615008372106921
epoch 60000 loss : 0.33077879872648525
epoch 65000 loss : 0.30333537090819584
epoch 70000 loss : 0.2794858089741792
epoch 75000 loss : 0.25892812312991587
epoch 80000 loss : 0.24119780823897027
epoch 85000 loss : 0.22583656353511342
epoch 90000 loss : 0.21244497028971704
epoch 95000 loss : 0.2006912468389013
```

```
[[0.01025062]
 [0.99730607]
 [0.02141321]
 [0.99722154]
 [0.03578171]
 [0.99701922]
 [0.04397049]
 [0.99574117]
 [0.04162245]
 [0.92902792]
 [0.03348791]
 [0.02511045]
 [0.94093942]
 [0.01870069]
 [0.99622948]
 [0.01431959]
 [0.99434455]
 [0.01143039]
 [0.98992477]
 [0.00952752]
 [0.98385905]]
```

Input data generate

```python
def generate_linear(n=100):
    import numpy as np
    pts = np.random.uniform(0, 1, (n, 2))
    inputs = []
    labels = []
    for pt in pts:
        inputs.append([pt[0], pt[1]])
        distance = (pt[0]-pt[1])/1.414
        if pt[0] > pt[1]:
            labels.append(0)
        else:
            labels.append(1)
    return np.array(inputs), np.array(labels).reshape(n, 1)
```

```python
def generate_XOR_easy():
    import numpy as np
    inputs = []
    labels = []

    for i in range(11):
        inputs.append([0.1*i, 0.1*i])
        labels.append(0)

        if 0.1*i == 0.5:
            continue

        inputs.append([0.1*i, 1-0.1*i])
        labels.append(1)

    return np.array(inputs), np.array(labels).reshape(21, 1)
```

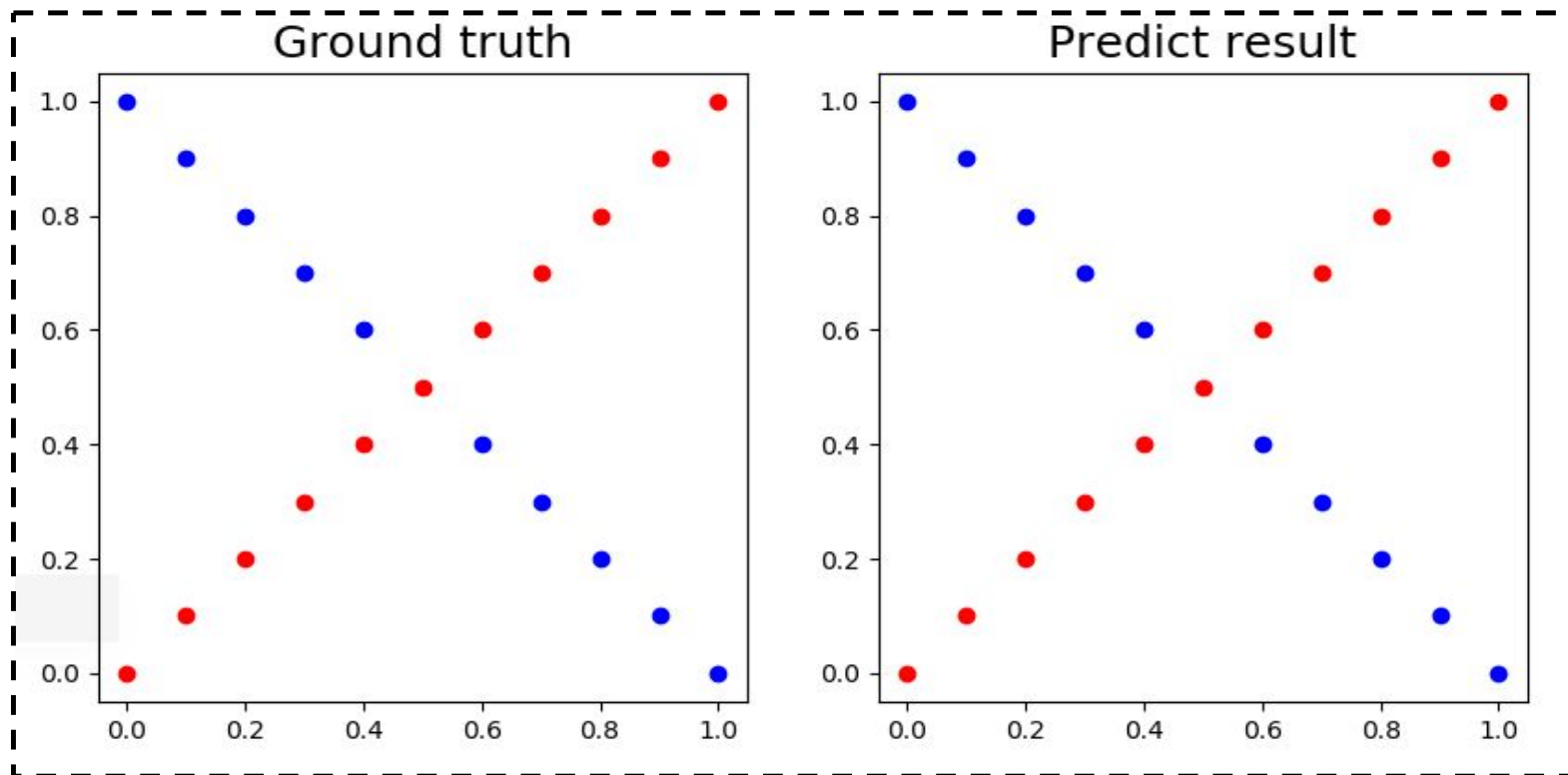**Don't overwrite these functions!!!**

Function usage

```python
x, y = generate_linear(n=100)
x, y = generate_XOR_easy()
```

Lab Description - Prediction

- Visualize the predictions and ground truth at the end of the training process

Scoring Criteria

- Report (40%)

- Demo(60%)

  - Experimental results (40%)

  - Questions (20%)

- Extra (10%)

  - Implement different optimizers. (2%)

  - Implement different activation functions. (3%)

  - Implement convolutional layers. (5%)

Scoring Criteria

- For experimental results, you have to achieve **at least 90% of accuracy to get the demo score**.

- If the zip file name or the report spec have format error, you will be **punished (-5)**

Report format

1. Introduction (20%)
2. Experiment setups (30%):

   A. Sigmoid functions

   B. Neural network

   C. Backpropagation
3. Results of your testing (20%)

   A. Screenshot and comparison figure
   B. Show the accuracy of your prediction
   C. Learning curve (loss, epoch curve)
   D. anything you want to present
4. Discussion (30%)

   A. Try different learning rates
   B. Try different numbers of hidden units
   C. Try without activation functions
   D. Anything you want to share
5. Extra (10%)

   A. Implement different optimizers. (2%)
   B. Implement different activation functions. (3%)
   C. Implement convolutional layers. (5%)

25

Important Date

- Assignment Deadline: 3/21 (Thu) 11:59 p.m.

- Demo date: 3/21 (Thu)

- Zip all files in one file
  - Report (.pdf)
  - Source code

- name it like「DL_LAB1_yourstudentID_name.zip」
  - ex:「DL_LAB1_311554005_高宗霖.zip」

Reference

1. http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html
2. http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML17_2.html

27

demo 時間

3/21 晚上實驗課 吳教授要上課

 Lab 1 demo 時間預定排在 3/21 下午 (每人約 5 分鐘)

 若時間上有困難, 請跟對應的助教另約 demo 時間

表單連結

U	V	W	X
	Google meet link		
	TA	高宗霖	OK/Please come in
15:50	高宗霖 (我是學生		
15:55			
16:00			
16:05			
16:10			
16:15			
16:20			
16:25			

	LAB1 Back-Propagation	LAB2 2048 TD	LAB3 CNN	LAB4 CNN	LAB5 RNN+VAE	LAB6 Deep RL	LAB7 GAN
Announce	3/14	3/21 (上課前30分鐘)	3/28 (上課前30分鐘)	4/11	4/25	5/2 (上課前30分鐘)	5/9
DEMO	*3/21	4/11	4/11	4/25	5/16		