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

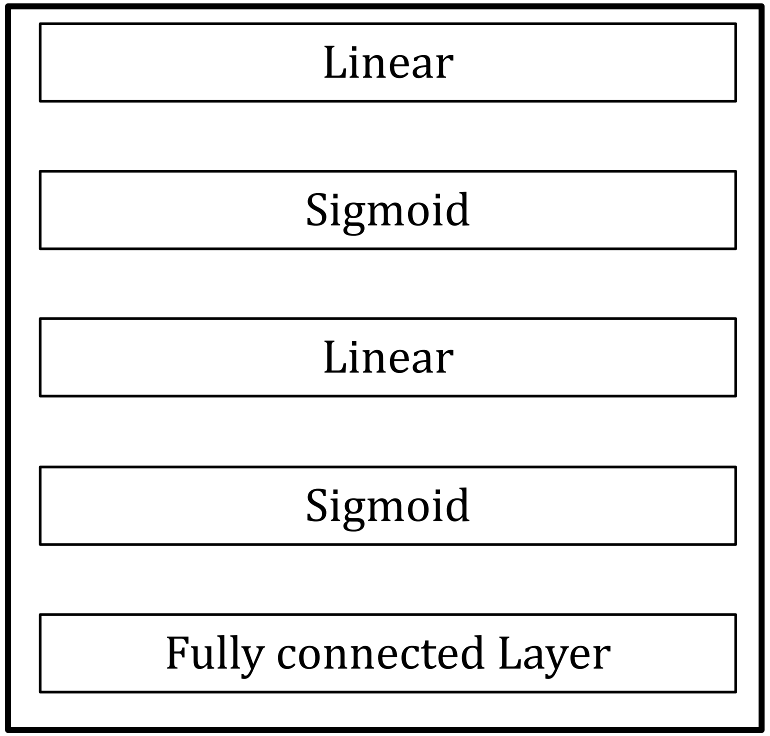
- Model

```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
```



Recap - Forward

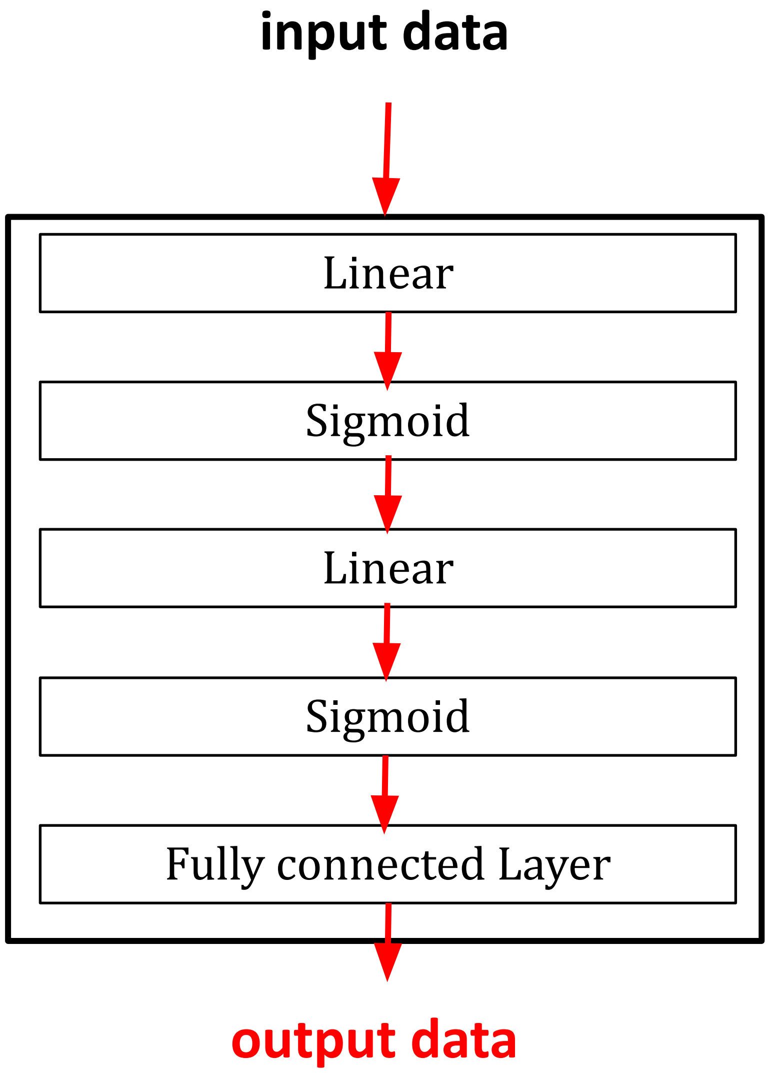
- Model

```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
```



Recap - Backpropagation

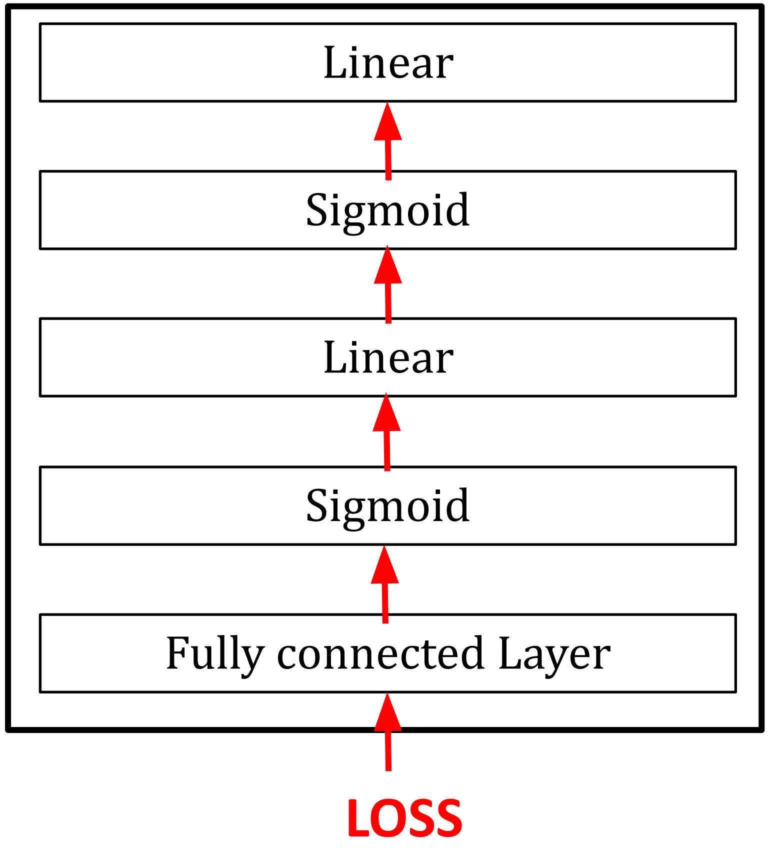
- Model

```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
```



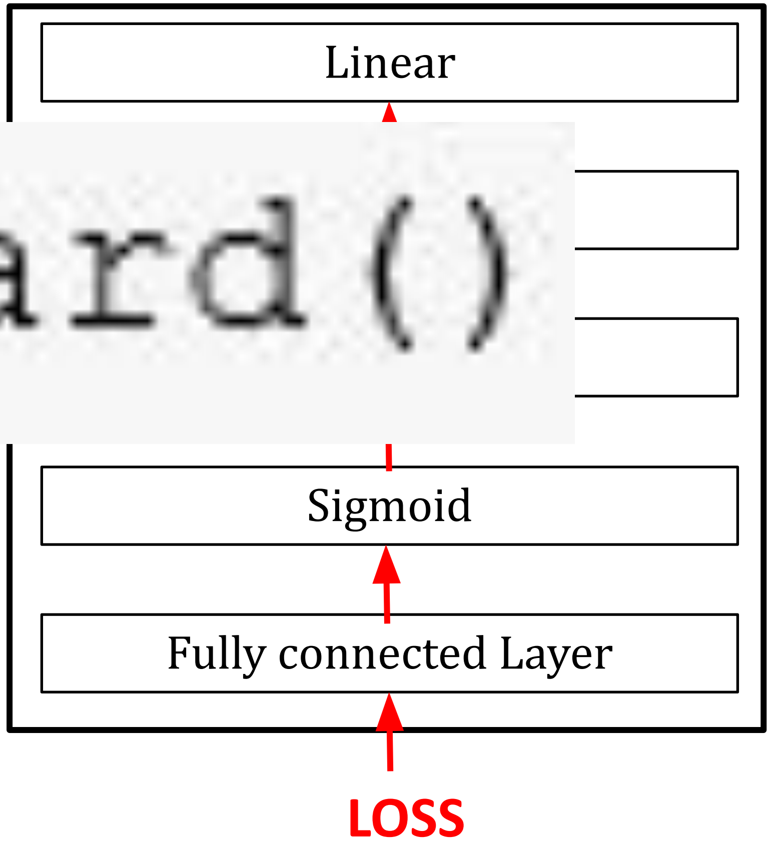
Recap - Backpropagation

- Model

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

```
loss.backward()
```

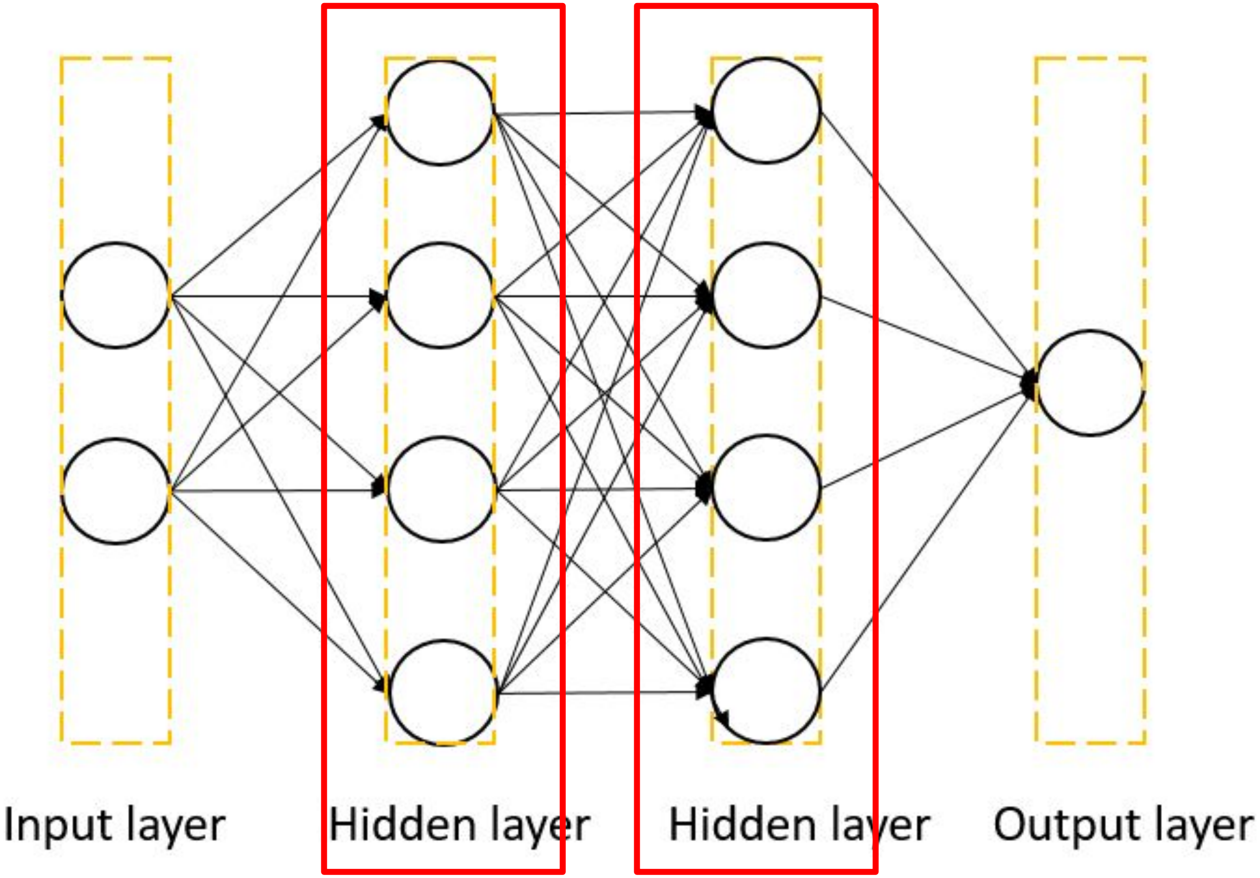
```python
    def forward(self, x):
        h1 = self.layer1(x)
        h2 = self.layer2(h1)
        out = self.flatten(h2)
        return out
```



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



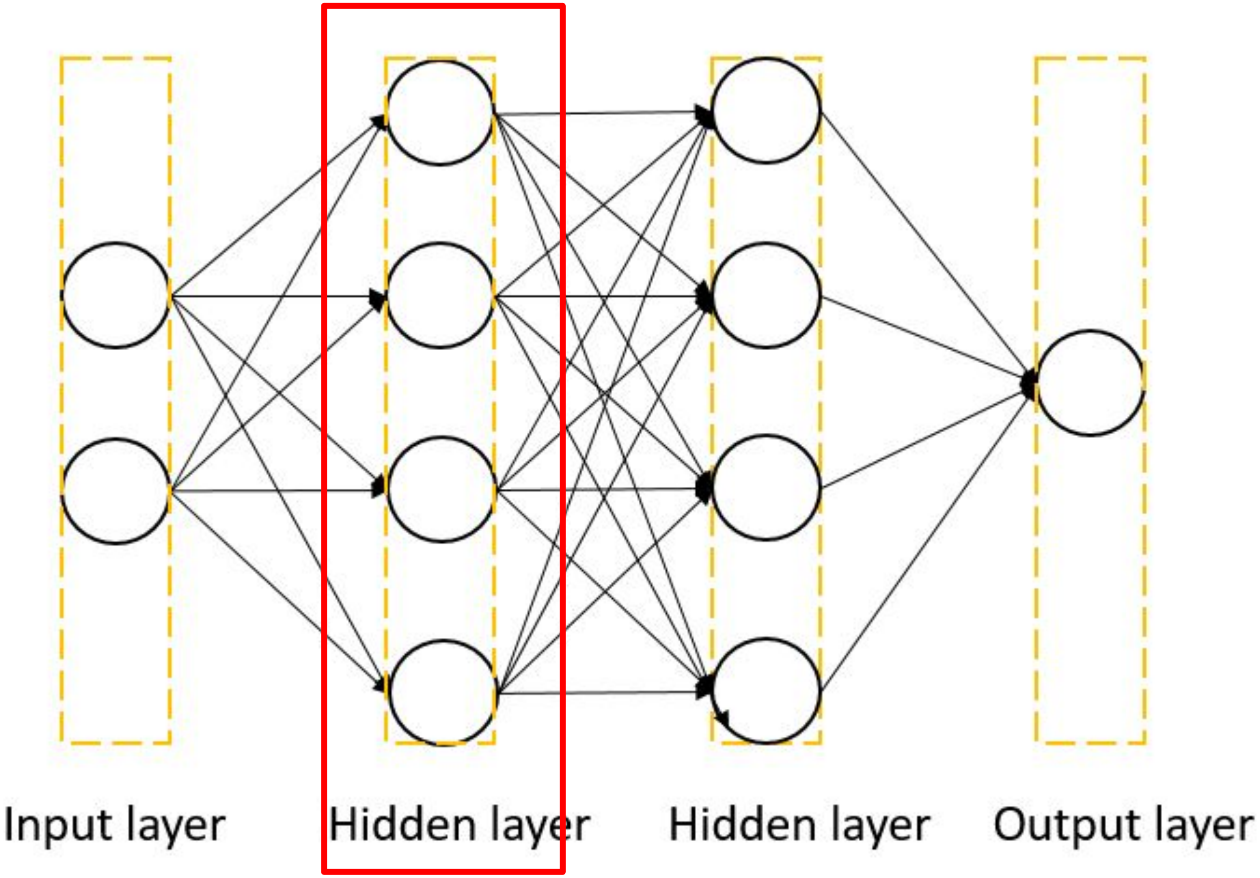
Lab Description

Lab Description

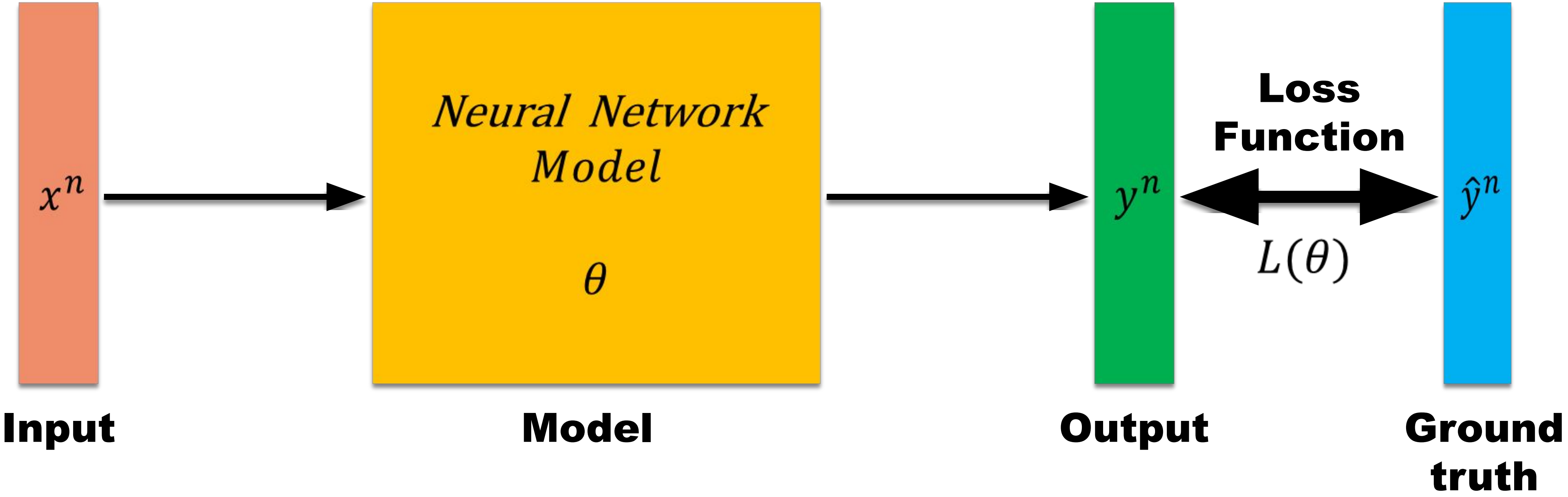
- Implement a simple neural network with two hidden layers
- You can only use **Numpy** and other **python standard libraries**.
  - **Pytorch, TensorFlow,….. are not allowed**
- Plot your comparison figure showing the predictions and ground truth.
- Plot your learning curve (loss, epoch).
- Print the 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....)



Lab Description



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

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

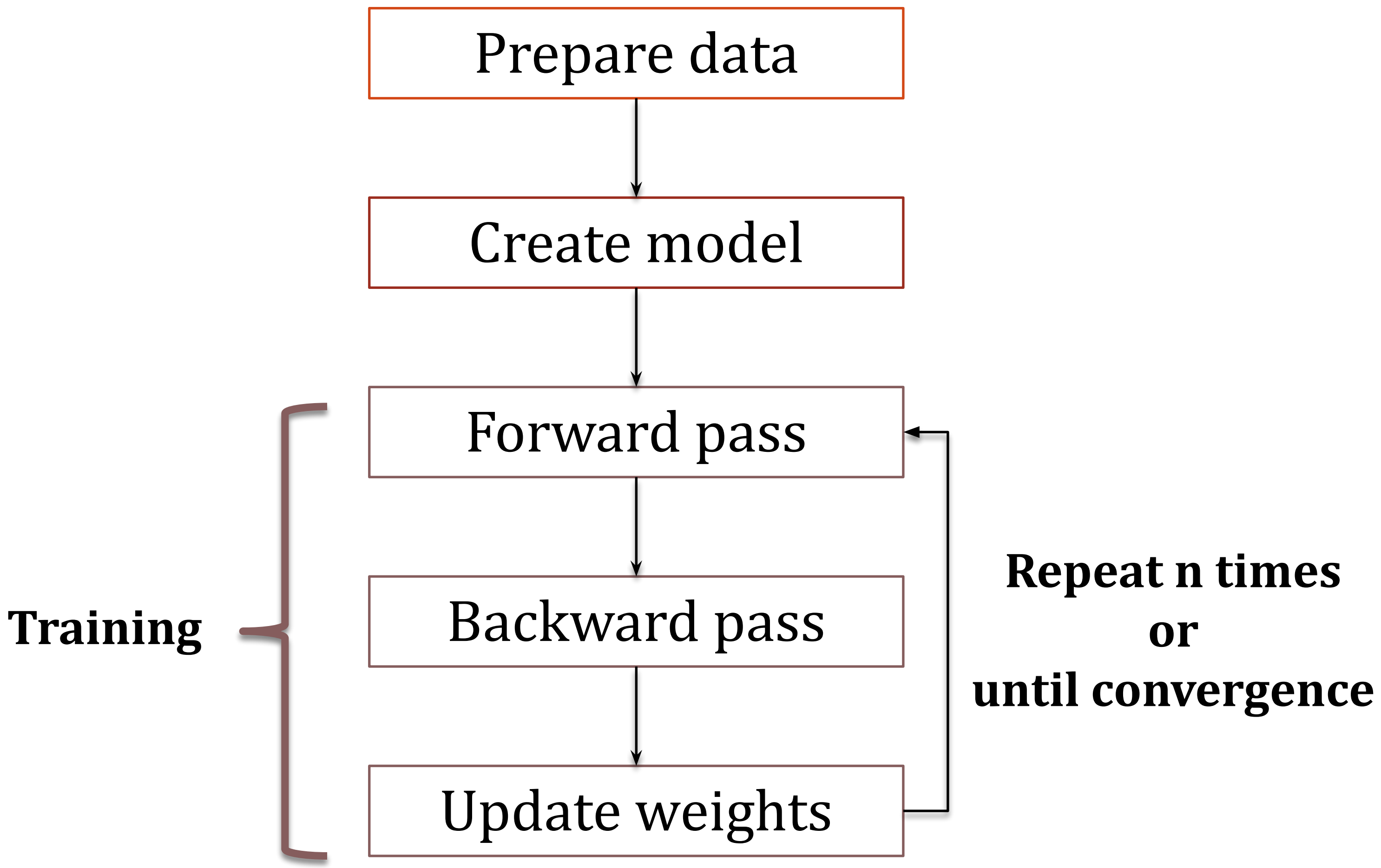
Compute $\nabla L(\theta^0)$ $\quad \theta^1 = \theta^0 - \rho \nabla L(\theta^0)$

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

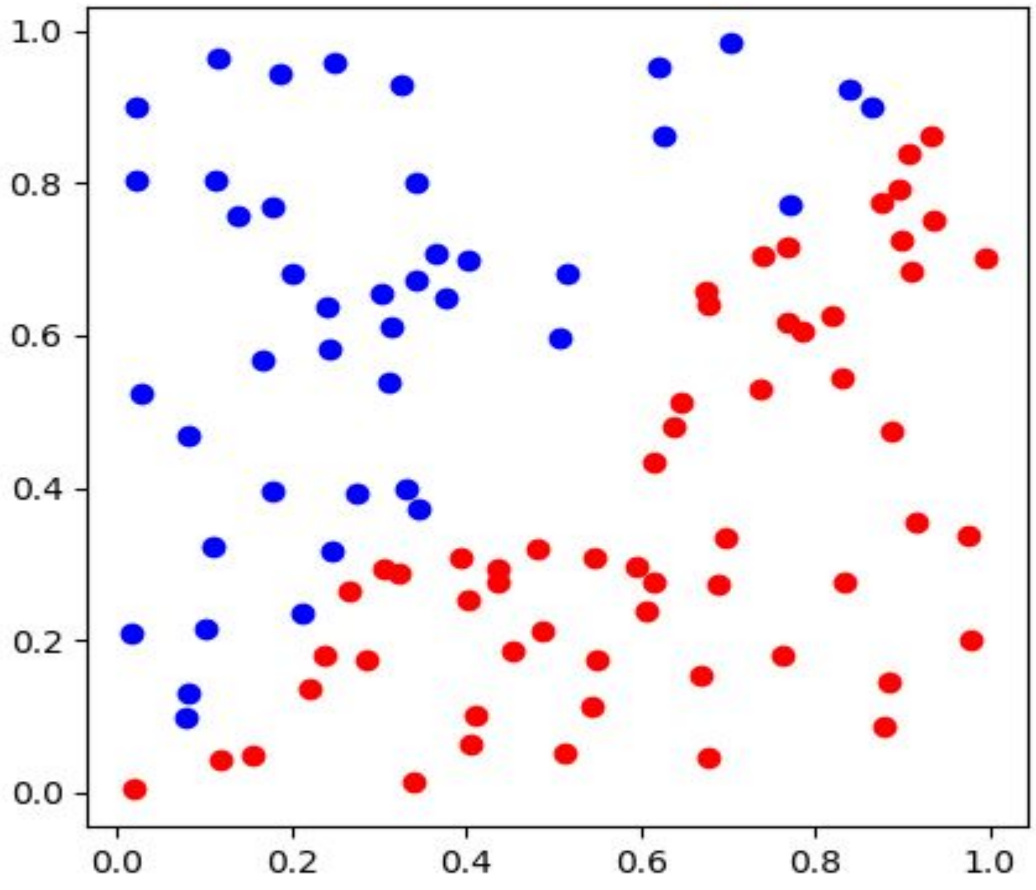
Compute $\nabla L(\theta^2)$ $\quad \theta^3 = \theta^2 - \rho \nabla L(\theta^2)$

ρ : Learning rate

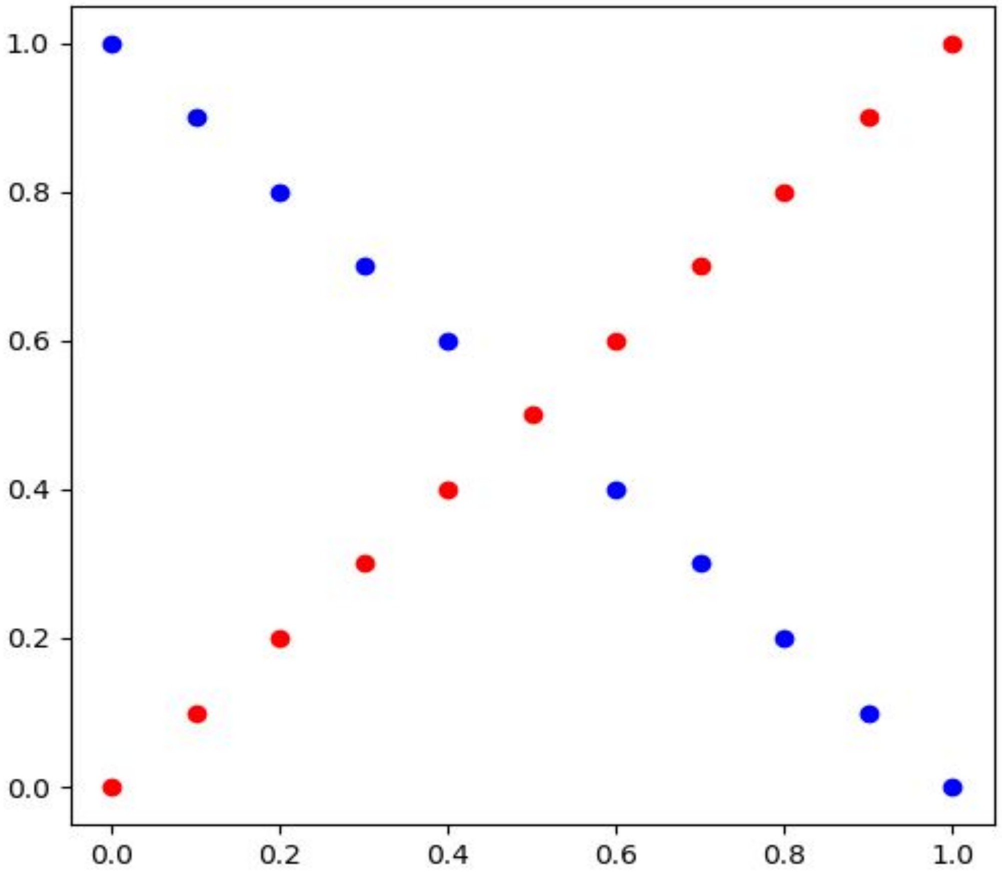
Lab Description – Flowchart



Lab Description - Data

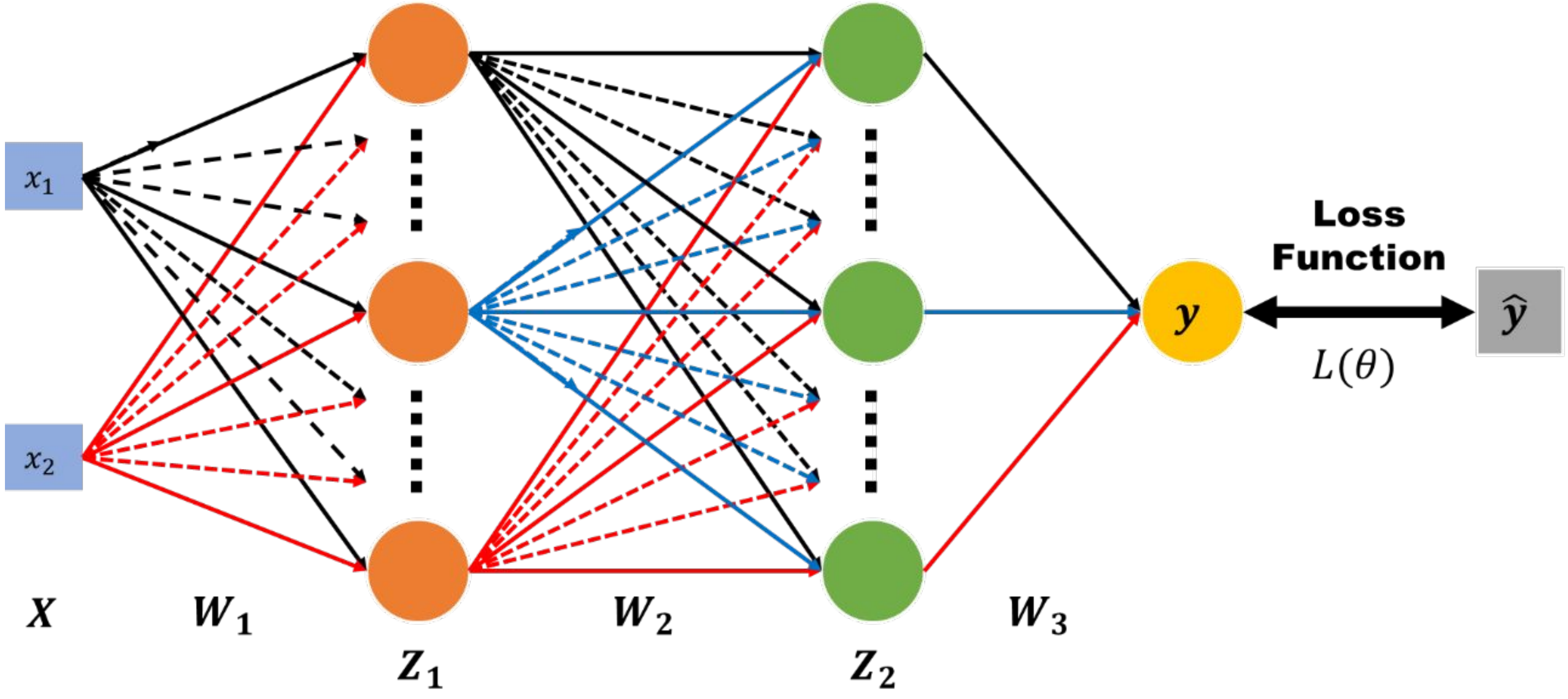


generate_linear()



generate_XOR_easy()

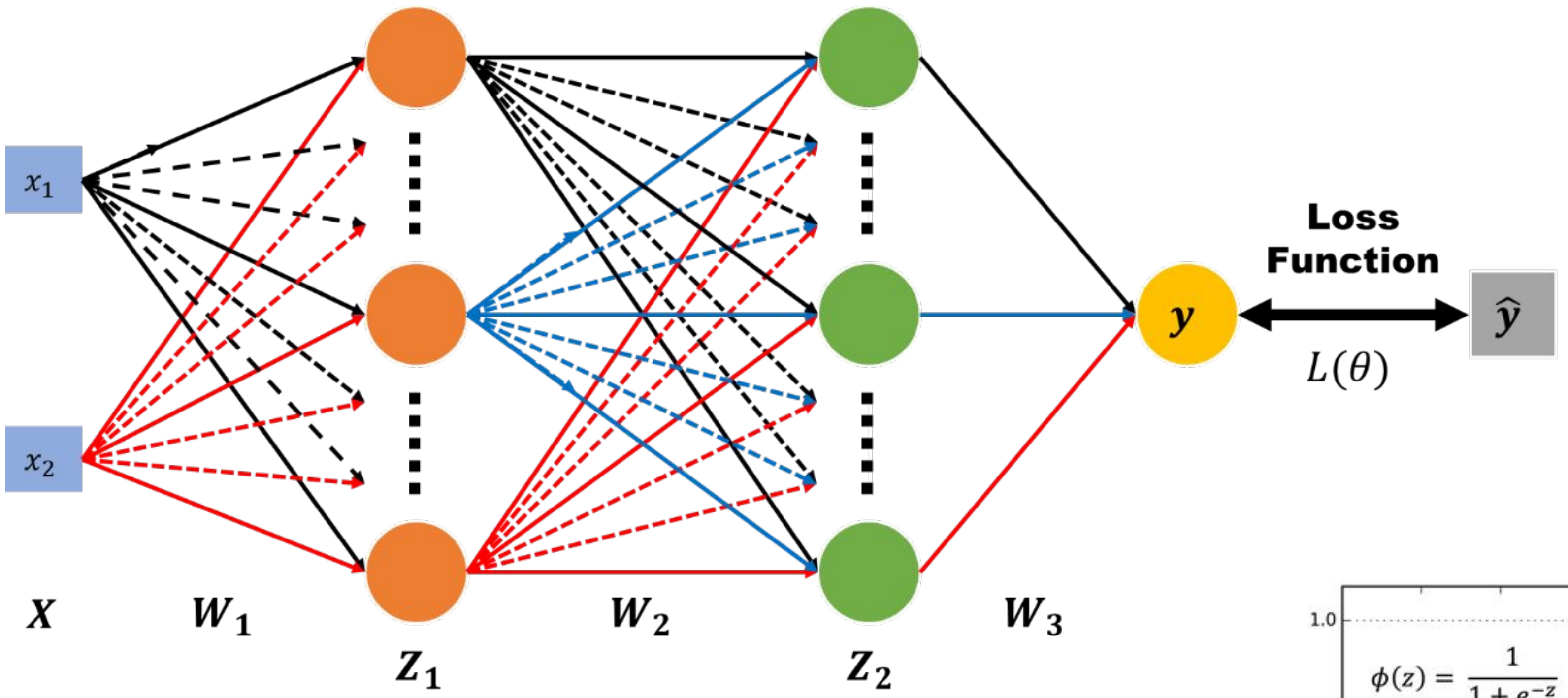
Lab Description – Architecture



X : $[x_1, x_2]$ y : *outputs* $\hat{y}$: *ground truth*

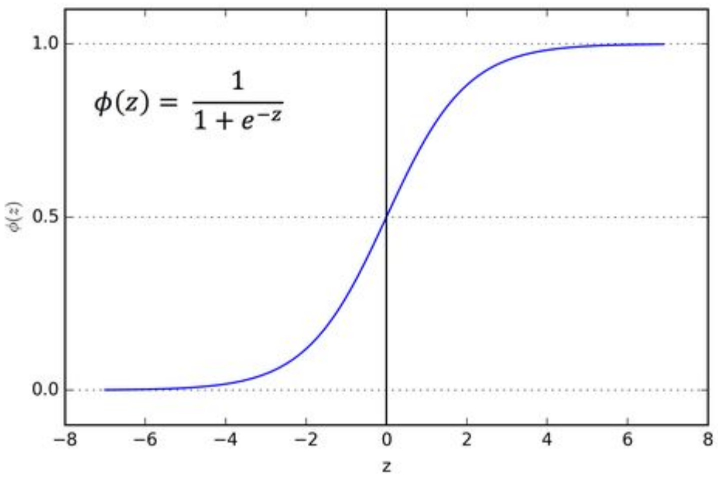
W_1, W_2, W_3 : *weight matrix of network layers*

Lab Description – Forward

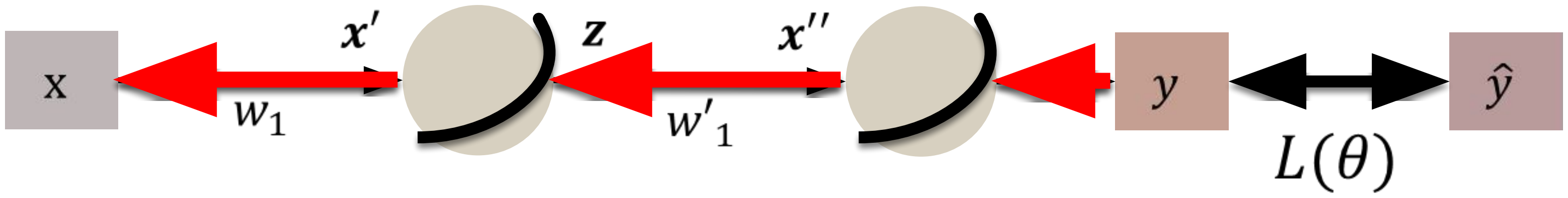


$$Z_1 = \sigma(XW_1) \qquad Z_2 = \sigma(Z_1 W_2) \qquad y = \sigma(Z_2 W_3)$$

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



Lab Description – Backward



$\boldsymbol{x' = xw_1}$ $\boldsymbol{z = \sigma(x')}$ $\boldsymbol{x'' = zw'_1}$ $\boldsymbol{y = \sigma(x'')}$

Chain rule

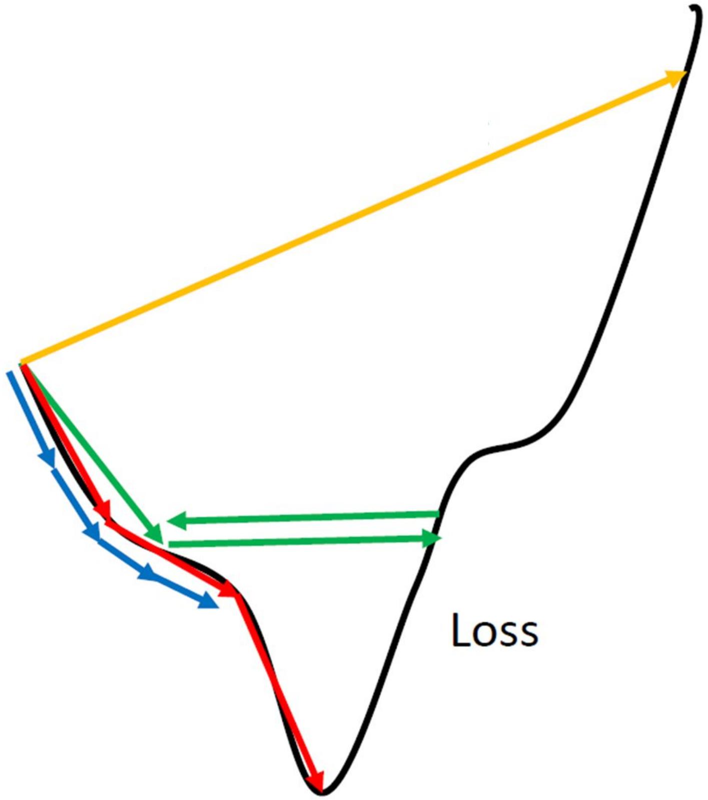
$$y = g(x) \qquad z = h(y)$$

$$\mathbf{x} \overset{\mathbf{g}()}{\longrightarrow} \boldsymbol{y} \overset{\boldsymbol{h}()}{\longrightarrow} \boldsymbol{z} \qquad \frac{d\boldsymbol{z}}{d\boldsymbol{x}} = \frac{d\boldsymbol{z}}{\cancel{d\boldsymbol{y}}}\frac{\cancel{d\boldsymbol{y}}}{d\boldsymbol{x}}$$

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

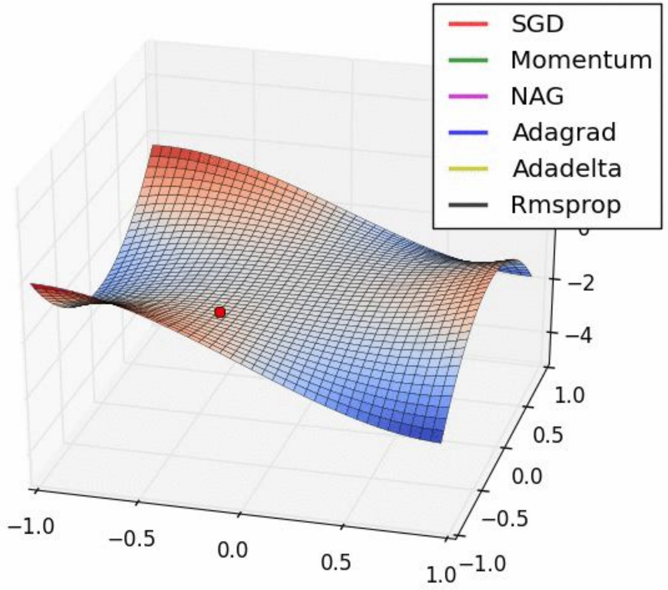
Lab Description – Gradient descent

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



http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Lab Description - Prediction

- In the training, you need to print loss

```
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 : 0.3934759509342479
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
```

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

```
[[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)
```

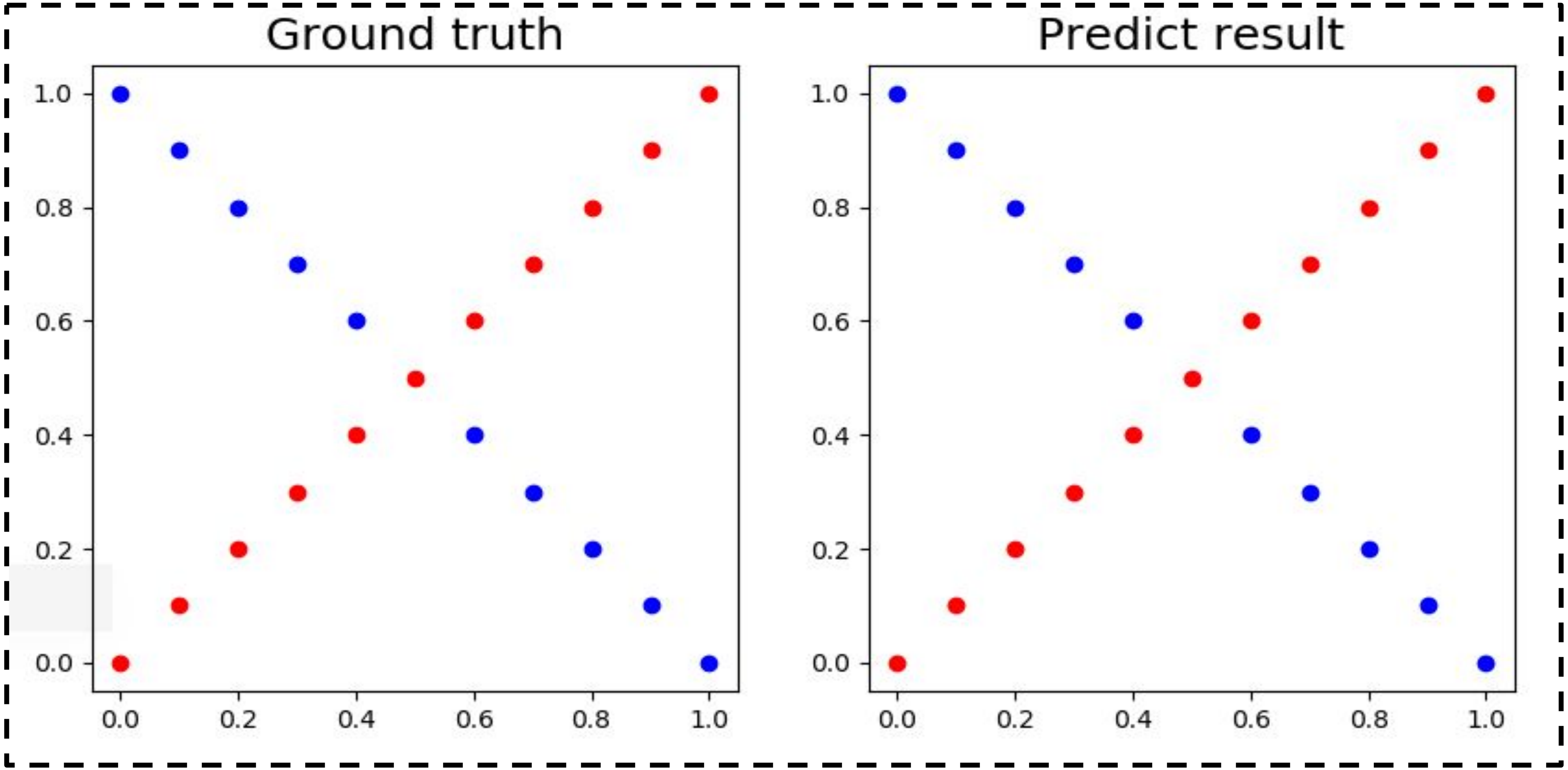
Don't overwrite these functions!!!

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%)

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*

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		