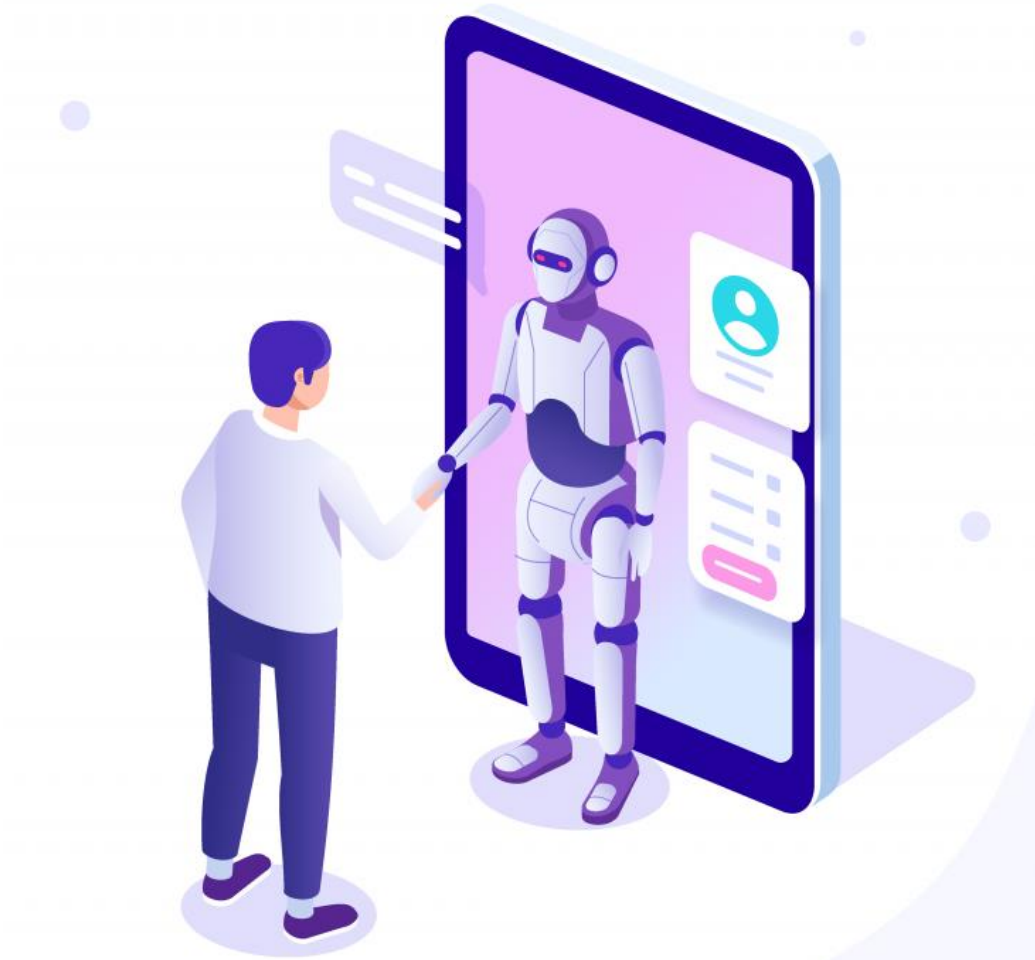
AI Programming

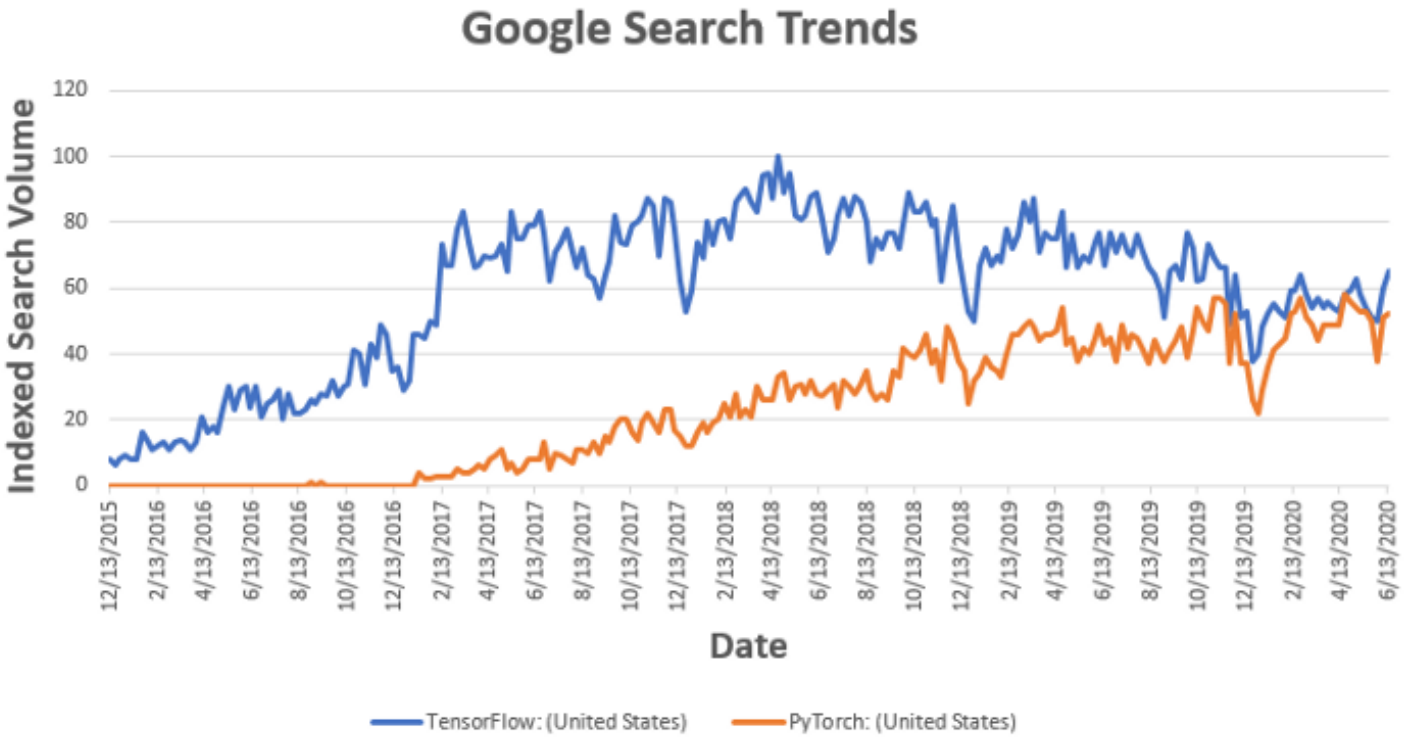
Lecture4 – Pytorch



Prof. Yong-Goo Shin

Pytorch

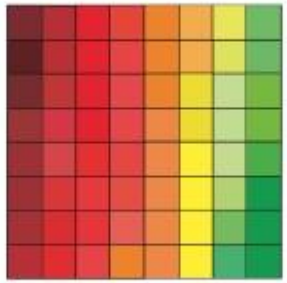
- Deep learning programming language
  - Tensorflow (Google)
  - Pytorch (FaceBook)
  - Keras



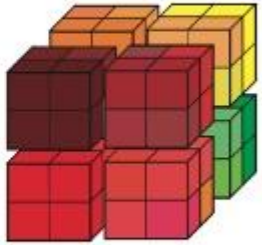
Pytorch tensor

- What is tensor?
  - Tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space.
  - 0-D tensor : scalar value
  - 1-D tensor : vector
  - 2-D tensor : matrix
  - 3-D tensor : cube

1-D tensor



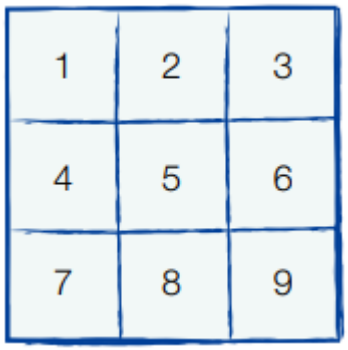
2-D tensor



3-D tensor

Pytorch tensor

- Tensor programming example
  - 2D tensor




```
x = torch.tensor([[1,2,3], [4,5,6], [7,8,9]])
print(x)
```

```
tensor([[1,  2,  3], [ 4,  5,  6], [7, 8, 9]])
```

```
print("Size:", x.size())
print("Shape:", x.shape)
print("랭크(차원):", x.ndimension())
```

```
Size: torch.Size([3, 3])
Shape: torch.Size([3, 3])
랭크(차원): 2
```

Pytorch tensor

- Tensor programming example
  - Check the tensor shape

```
import torch

x = torch.tensor([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print("Size :", x.size())
print("Shape :", x.shape)
print("Dimension (Rank)", x.ndimension())
```

```
Size : torch.Size([3, 3])
Shape : torch.Size([3, 3])
Dimension (Rank) 2

Process finished with exit code 0
```

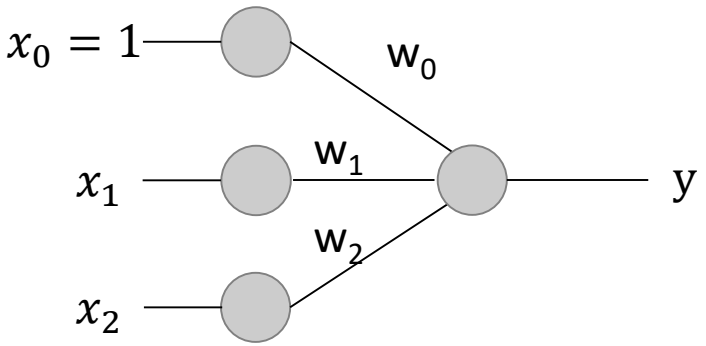
Pytorch tensor

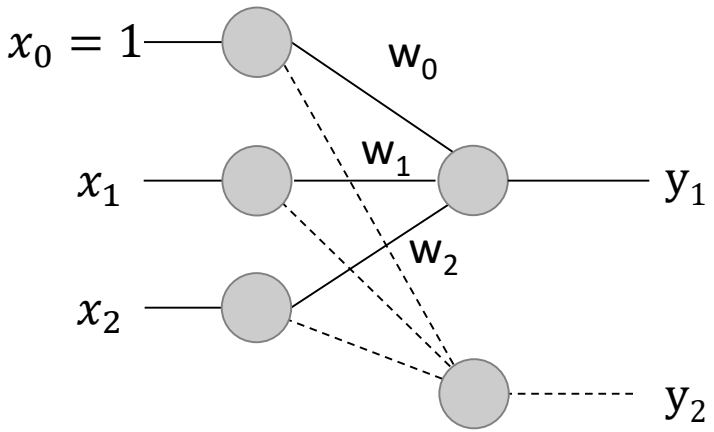
- Matrix multiplication

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} a* & b* \\ c* & d* \\ e* & f* \end{bmatrix} = \begin{bmatrix} aa*+bc*+ce* & ab*+bd*+cf* \\ aa*+ec*+fe* & db*+ed*+ff* \end{bmatrix}$$

2x3 3x2 = 2x2

- Neural Network





Pytorch tensor

- Tensor programming example
  - matrix w – 5 x 3 random matrix
  - matrix x – 3 x 2 constant matrix

```
import torch

w = torch.randn(5,3, dtype=torch.float)
x = torch.tensor([[1.0,2.0], [3.0,4.0], [5.0,6.0]])
wx = torch.mm(w,x) # w의 행은 5, x의 열은 2, 즉 shape는 [5, 2]입니다.
print("wx size:", wx.size())
print("wx:", wx)
```

```
wx size: torch.Size([5, 2])
wx: tensor([[-3.8183, -4.2929],
        [-4.7751, -5.3661],
        [ 2.1691,  2.2244],
        [-0.0916, -0.8514],
        [-3.8995, -4.3956]])
```

Pytorch tensor

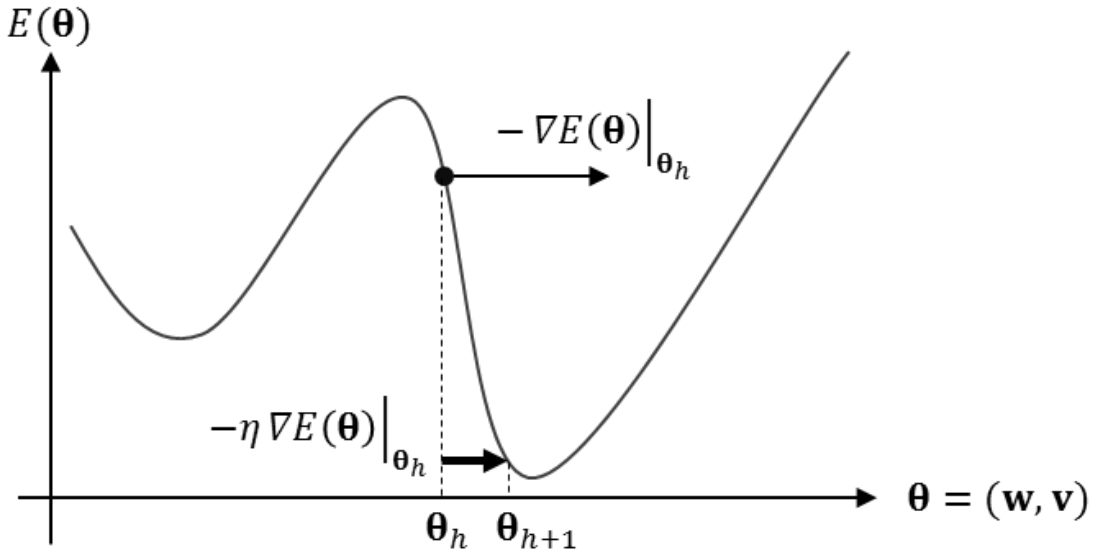
- Tensor programming example
  - matrix wx – 5 x 3 matrix
  - matrix b –5 x 2 random matrix

```
b = torch.randn(5,2, dtype=torch.float)

result = wx + b
print("result size:", result.size())
print("result:", result)
```

```
result size: torch.Size([5, 2])
result: tensor([[-3.7656, -4.9200],
        [-2.7057, -5.4891],
        [ 2.1144,  0.2841],
        [ 1.4777,  0.0384],
        [-3.6158, -3.2231]])
```

Gradient decent

- Auto gradient (Autograd)
  - Minimizing loss using gradient decent
  - Autograd in Pytorch computes the gradient automatically



Gradient decent

- Autograd example

$$l = a^2 = (3w)^2 = 9w^2$$

```
w = torch.tensor(1.0, requires_grad=True)
a = w*3
l = a**2
```

"w.grad" contains the gradient of w

```
l.backward()
print('l을 w로 미분한 값은 {}'.format(w.grad))
```

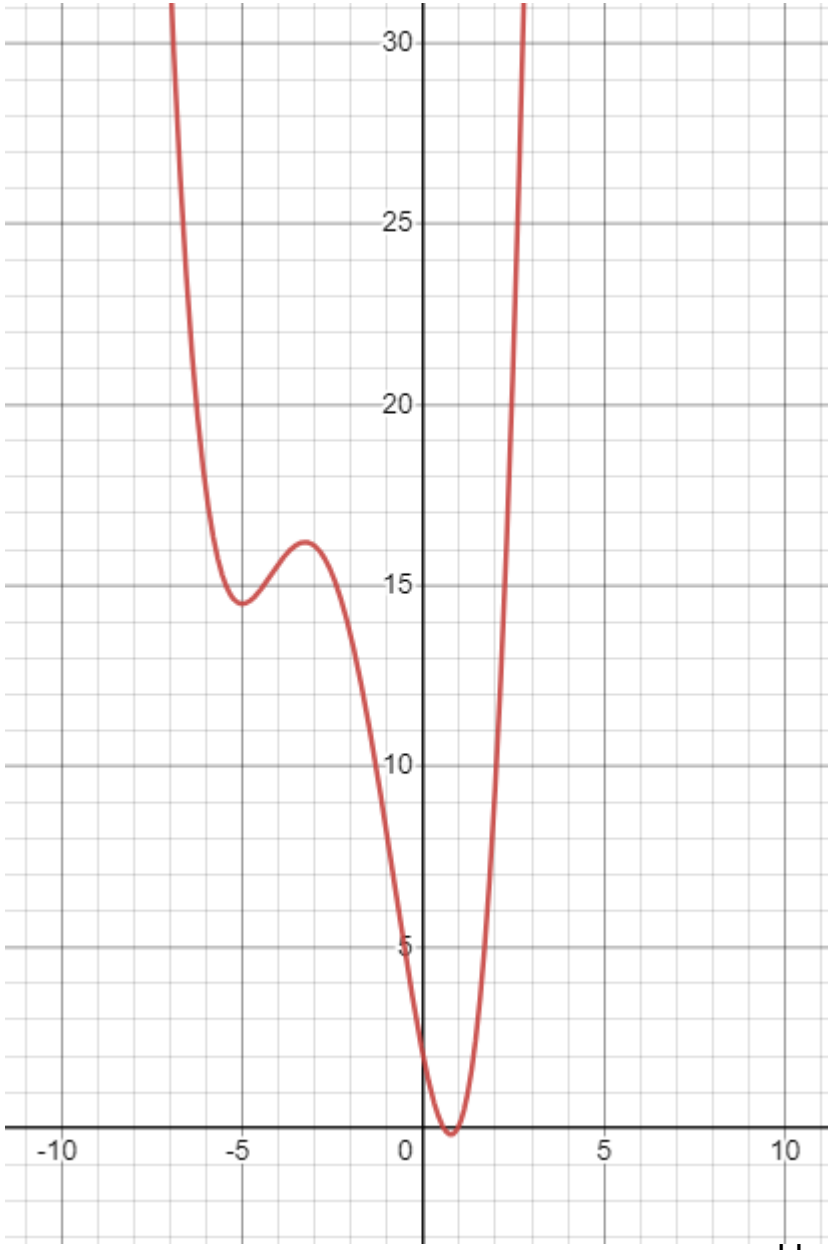
```
l을 w로 미분한 값은 tensor(18.0)
```

Gradient decent

- Practice1 – minimization using Autograd
  - Find minimum y value

$$y = 0.1x^4 + x^3 + 2x^2 - 5x + 2$$

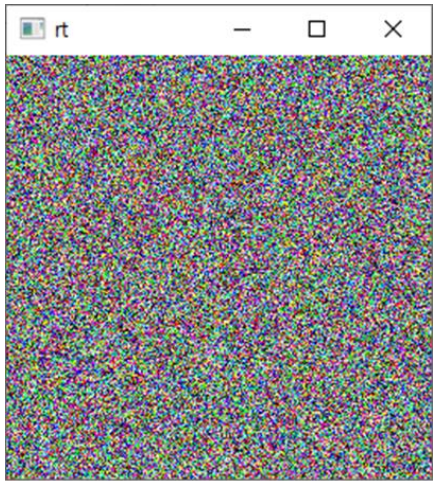
```python
import torch

x_t = -10.0
lr = 0.1
for it in range(100):
    x = torch.tensor(x_t, requires_grad=True)
    y = 0.1 * x ** 4 + x ** 3 + 2 * x ** 2 - 5 * x + 2

    print("x : %f  y : %f " % (x_t, y))

    y.backward()
    # print(format(x.grad))

    x_t = x_t - lr * x.grad
```

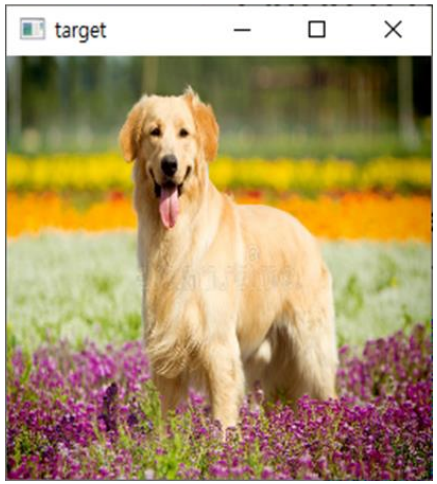


Gradient decent

- Practice2 –

  random noise to image

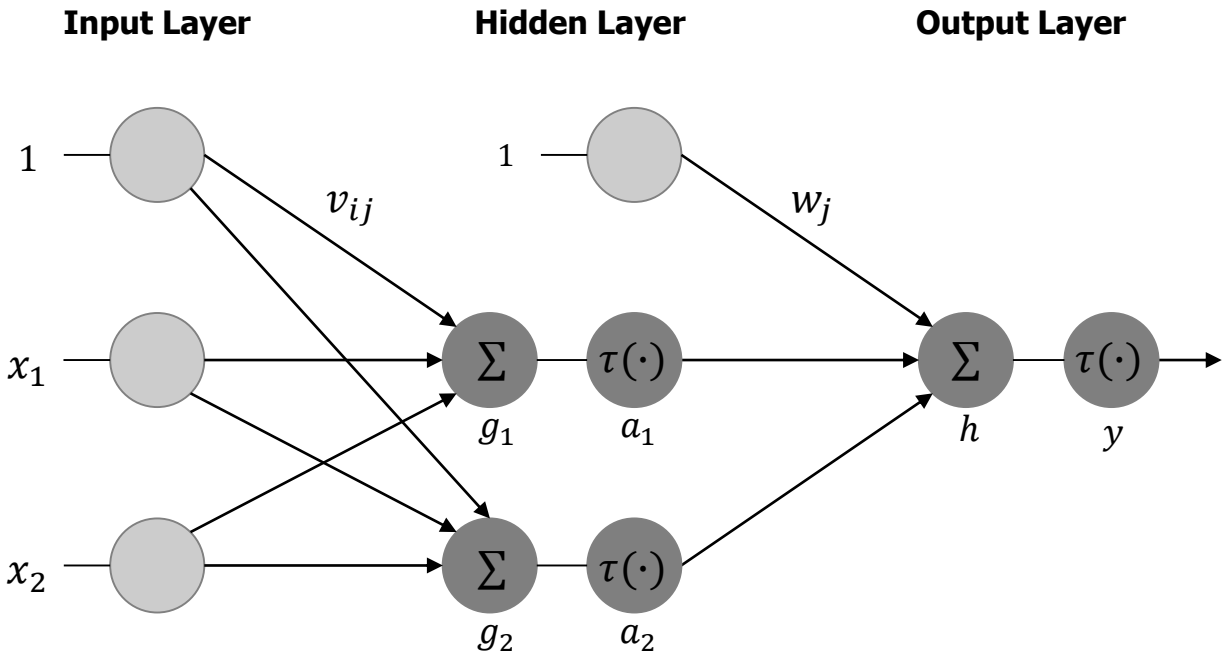





```python
import torch
import cv2
import numpy as np

size = 256
num_training = 10000
lr = 100

random_tensor = torch.randn((size, size, 3), dtype=torch.float)
random_tensor = torch.clip(random_tensor, min=-1.0, max=1.0)

# ---- random tensor visualization
vis_rt = (random_tensor + 1) / 2 * 255.0
vis_rt = vis_rt.numpy()
# cv2.imshow('rt', vis_rt.astype(np.uint8))
# cv2.waitKey(-1)

# ---- training
target = cv2.imread('./images/dog1.jpg')
target = cv2.resize(target, (size, size))
cv2.imshow('target', target)
# cv2.waitKey(-1)

target = (target / 255.0) * 2 - 1
target = torch.from_numpy(target)

for it in range(num_training):
    random_tensor.requires_grad_(True)
    loss = torch.mean((target - random_tensor)**2)
    loss.backward()

    with torch.no_grad():
        random_tensor = random_tensor - lr*random_tensor.grad

    print('it : %d  loss val : %.5f ' %(it, loss))

    vis_rt = (random_tensor + 1) / 2 * 255.0
    vis_rt = vis_rt.numpy()
    cv2.imshow('rt', vis_rt.astype(np.uint8))
    cv2.waitKey(1)
```

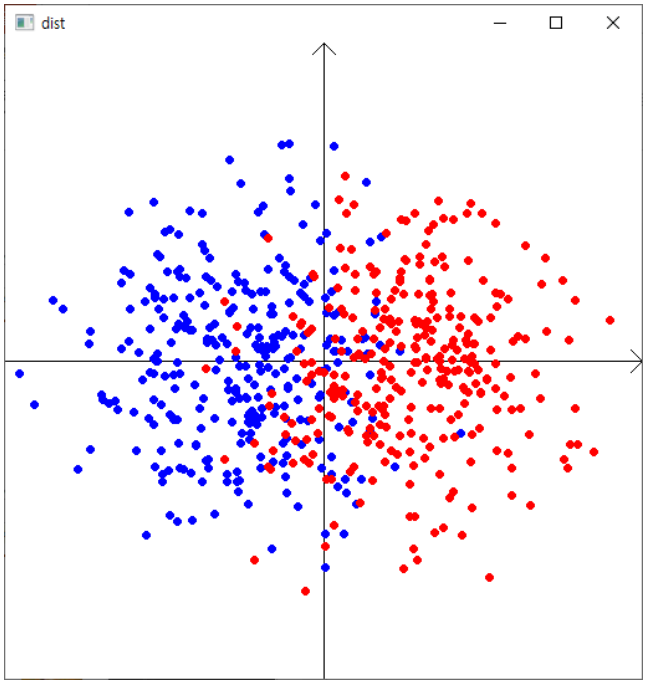
Neural Network

- Example of neural network
  - Input, hidden, and out layers
  - x_1 and x_2 : input values
  - y : target value
  - **v, w : training values**



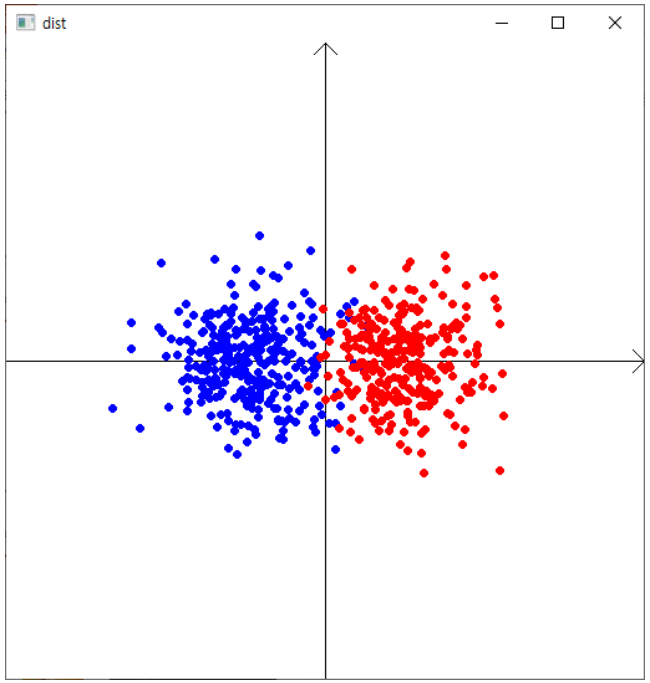
Neural Network

- Example of neural network (Single layer)
  - Step1. Build dataset and visualization
    - Two different Gaussian distributions



mean / std : (-1.0, 0.0) / (1.0, 1.0)

mean / std : (1.0, 0.0) / (1.0, 1.0)



mean / std : (-1.0, 0.0) / (0.5, 0.5)

mean / std : (1.0, 0.0) / (0.5, 0.5)

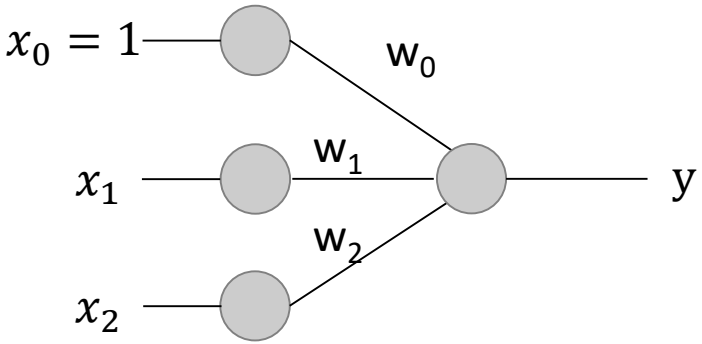
Neural Network

- Example of neural net
  - Step1. Building da
  - Two different Gau

```python
import numpy as np
import cv2

def gauss_2d(mu1, mu2, sigma=0.5, num=300):
    x = np.random.normal(mu1, sigma, num)
    y = np.random.normal(mu2, sigma, num)

    return (x, y)

def visualization(paper, dist, color):
    x = dist[0]
    y = dist[1]

    for it in range(len(x)):
        x_p = int(64 * x[it] + 256)
        y_p = int(-64 * y[it] + 256)
        cv2.circle(paper, (x_p, y_p), radius=2, color=color, thickness=2)


dist1 = gauss_2d(-1.0, 0.0)
dist2 = gauss_2d(1.0, 0.0)

paper = np.ones((512, 512, 3), dtype=np.uint8) * 255
paper = cv2.arrowedLine(paper, (0, 256), (512, 256), (0, 0, 0), tipLength=0.025)
paper = cv2.arrowedLine(paper, (256, 512), (256, 0), (0, 0, 0), tipLength=0.025)

visualization(paper, dist1, color=(255, 0, 0))
visualization(paper, dist2, color=(0, 0, 255))
cv2.imshow('dist', paper)
cv2.waitKey(-1)
```

Neural Network

- Example of neural network (Single layer)
  - Step2. Design network




```python
# ---- network
class NeuralNet(torch.nn.Module):
    def __init__(self, input_size):
        super(NeuralNet, self).__init__()
        self.inpput_size = input_size
        self.fc1 = torch.nn.Linear(self.input_size, 1)
        self.act1 = torch.nn.Sigmoid()

    def forward(self, input_tensor):
        fc1 = self.fc1(input_tensor)
        output = self.act1(fc1)
        return output
```

Neural Network

- Example of neural network (Single layer)
  - Step3. Training and test set

```python
def train_test_set(mu_x1, mu_y1, mu_x2, mu_y2, sigma=0.5):
    # ---- train / test
    train = np.zeros((400, 2), dtype=np.float32)
    train_gt = np.zeros((400, 1), dtype=np.float32)

    test = np.zeros((200, 2), dtype=np.float32)
    test_gt = np.zeros((200, 1), dtype=np.float32)

    # ---- training set
    for it in range(400):
        if it < 200:
            train[it, 0] = np.random.normal(mu_x1, sigma)
            train[it, 1] = np.random.normal(mu_y1, sigma)
            train_gt[it, 0] = 0
        else:
            train[it, 0] = np.random.normal(mu_x2, sigma)
            train[it, 1] = np.random.normal(mu_y2, sigma)
            train_gt[it, 0] = 1

    # ---- test set
    for it in range(200):
        if it < 100:
            test[it, 0] = np.random.normal(mu_x1, sigma)
            test[it, 1] = np.random.normal(mu_y1, sigma)
            test_gt[it, 0] = 0
        else:
            test[it, 0] = np.random.normal(mu_x2, sigma)
            test[it, 1] = np.random.normal(mu_y2, sigma)
            test_gt[it, 0] = 1

    return train, train_gt, test, test_gt
```

Neural Network

- Example of neural network (Single layer)
  - Step4. Training and test network

```python
from Lecture4_function import *
import torch

# ---- network
class NeuralNet(torch.nn.Module):
    def __init__(self, input_size):
        super(NeuralNet, self).__init__()
        self.input_size = input_size
        self.fc1 = torch.nn.Linear(self.input_size, 1)
        self.act1 = torch.nn.Sigmoid()

    def forward(self, input_tensor):
        fc1 = self.fc1(input_tensor)
        output = self.act1(fc1)
        return output

dist1 = gauss_2d(-1.0, 0.0)
dist2 = gauss_2d(1.0, 0.0)

paper = np.ones((512, 512, 3), dtype=np.uint8) * 255
paper = cv2.arrowedLine(paper, (0, 256), (512, 256), (0, 0, 0), tipLength=0.025)
paper = cv2.arrowedLine(paper, (256, 512), (256, 0), (0, 0, 0), tipLength=0.025)

visualization(paper, dist1, color=(255, 0, 0))
visualization(paper, dist2, color=(0, 0, 255))
cv2.imshow('dist', paper)
cv2.waitKey(1)
```

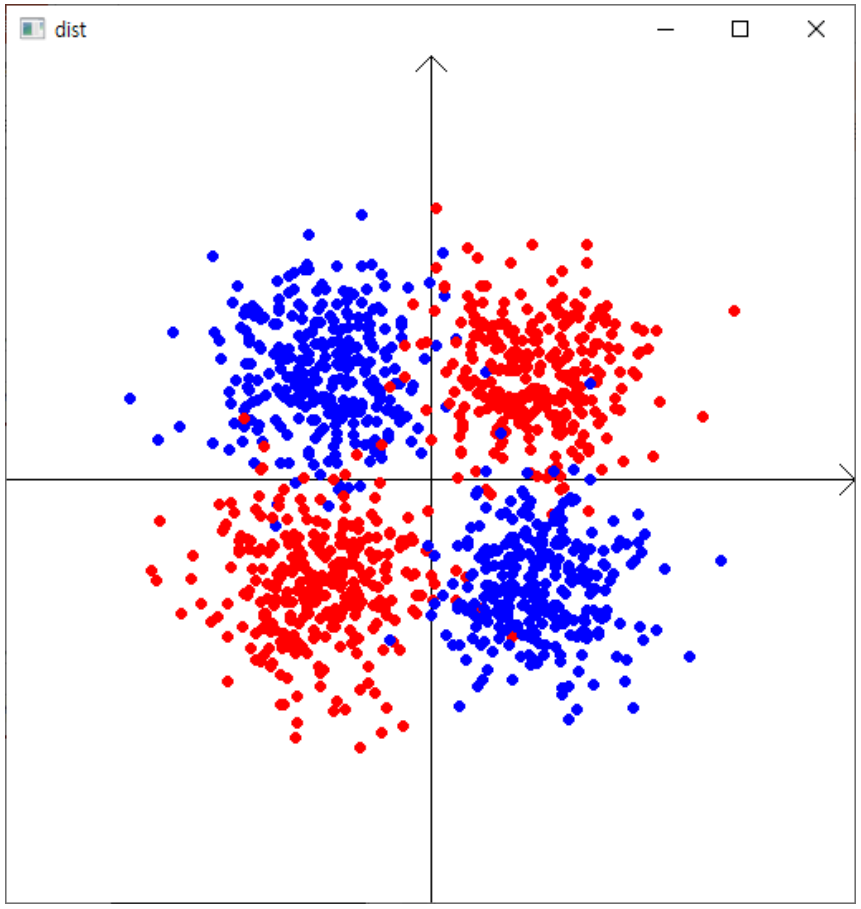
Neural Network

- Example of neural network
  - Step4. Training and test

```
# ------ network
model = NeuralNet(2)
learning_rate = 0.01
loss = torch.nn.BCELoss()
num_iter = 10000
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

train_set, train_gt, test_set, test_gt = train_test_set(-1.0, 0.0, 1.0, 0.0)

for it in range(num_iter):
    model.train()
    optimizer.zero_grad()
    train_output = model(torch.from_numpy(train_set))

    train_loss = loss(train_output, torch.from_numpy(train_gt))
    train_loss.backward()
    optimizer.step()

    model.eval()
    test_loss = loss(model(torch.from_numpy(test_set)), torch.from_numpy(test_gt))

    if it % 100 == 0:
        print("train : %f  test : %f " %(train_loss.item(), test_loss.item()))


# ------- user input test
user_input = np.zeros((1, 2), dtype=np.float32)
user_input[0, 0] = 0.0
user_input[0, 1] = 0.0

user_output = model(torch.from_numpy(user_input))
print("user : %f " %user_output.item())
```

Neural Network

- Example of neural network (Multiple layers)
  - XOR Problem



mean1 / mean2: (-1.0, 1.0) / (1.0, -1.0)

mean1 / mean2: (1.0, 1.0) / (-1.0, -1.0)

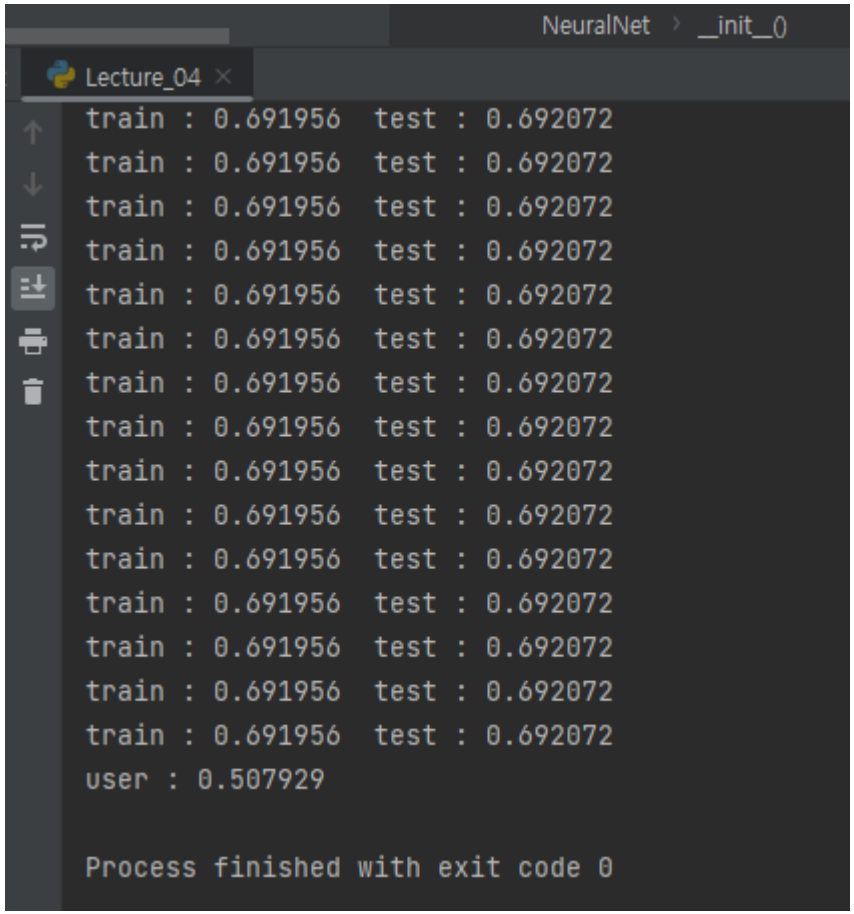
Neural Network

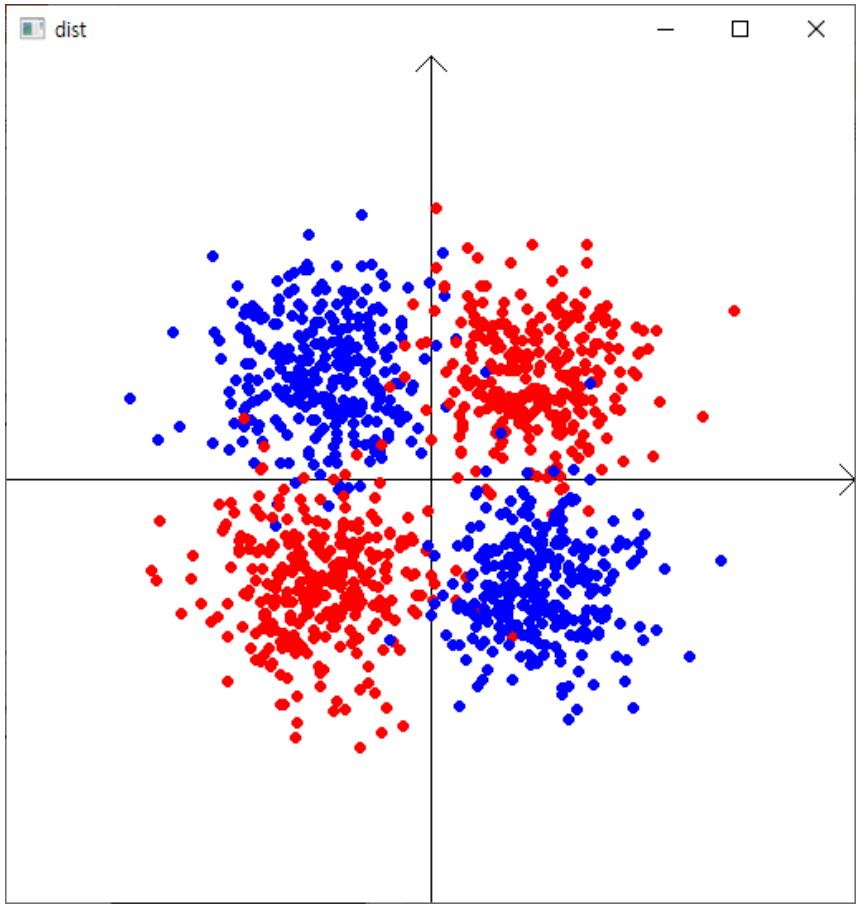
- Example of neural network (Multiple layers)
  - Training and test set

```python
def train_test_set(mu_x1, mu_y1, mu_x2, mu_y2, mu_x3, mu_y3, mu_x4, mu_y4, sigma=0.5):
    # ---- train / test
    train = np.zeros((400, 2), dtype=np.float32)
    train_gt = np.zeros((400, 1), dtype=np.float32)

    test = np.zeros((200, 2), dtype=np.float32)
    test_gt = np.zeros((200, 1), dtype=np.float32)

    # ---- training set
    for it in range(400):
        if it < 100:
            train[it, 0] = np.random.normal(mu_x1, sigma)
            train[it, 1] = np.random.normal(mu_y1, sigma)
            train_gt[it, 0] = 0
        elif it >= 100 and it < 200:
            train[it, 0] = np.random.normal(mu_x2, sigma)
            train[it, 1] = np.random.normal(mu_y2, sigma)
            train_gt[it, 0] = 1
        elif it >= 200 and it < 300:
            train[it, 0] = np.random.normal(mu_x3, sigma)
            train[it, 1] = np.random.normal(mu_y3, sigma)
            train_gt[it, 0] = 1
        else:
            train[it, 0] = np.random.normal(mu_x4, sigma)
            train[it, 1] = np.random.normal(mu_y4, sigma)
            train_gt[it, 0] = 0

    # ---- test set
    for it in range(200):
        if it < 50:
            test[it, 0] = np.random.normal(mu_x1, sigma)
            test[it, 1] = np.random.normal(mu_y1, sigma)
            test_gt[it, 0] = 0
        elif it >= 50 and it < 100:
            test[it, 0] = np.random.normal(mu_x2, sigma)
            test[it, 1] = np.random.normal(mu_y2, sigma)
            test_gt[it, 0] = 1
        elif it >= 100 and it < 150:
            test[it, 0] = np.random.normal(mu_x3, sigma)
            test[it, 1] = np.random.normal(mu_y3, sigma)
            test_gt[it, 0] = 1
        else:
            test[it, 0] = np.random.normal(mu_x4, sigma)
            test[it, 1] = np.random.normal(mu_y4, sigma)
            test_gt[it, 0] = 0

    return train, train_gt, test, test_gt
```

Neural Network

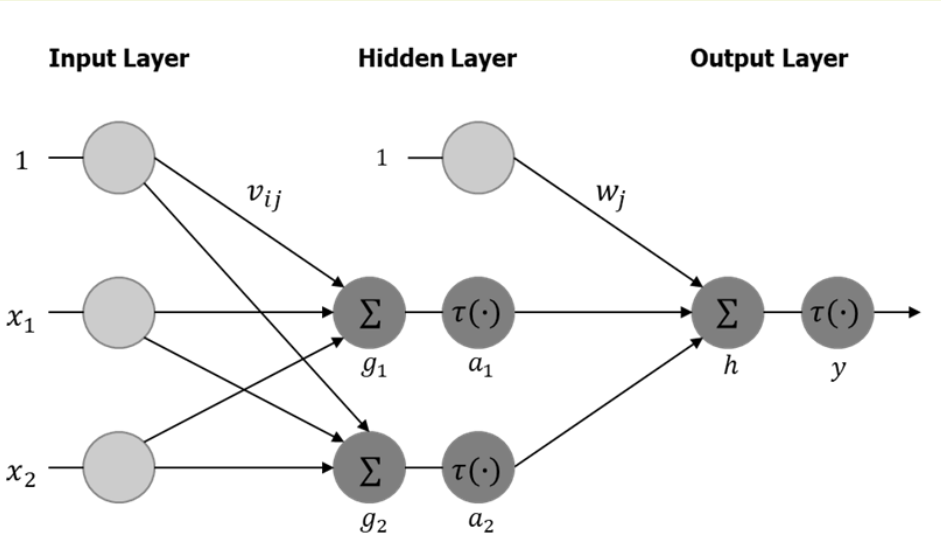
- Example of neural network (Multiple layers)
  - Training Network – **Fail**





Neural Network

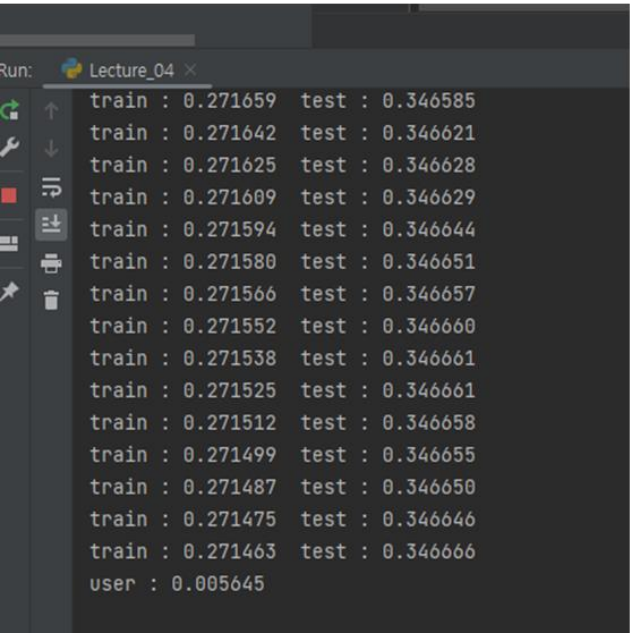
- Example of neural network (Multiple layers)
  - Redesign network

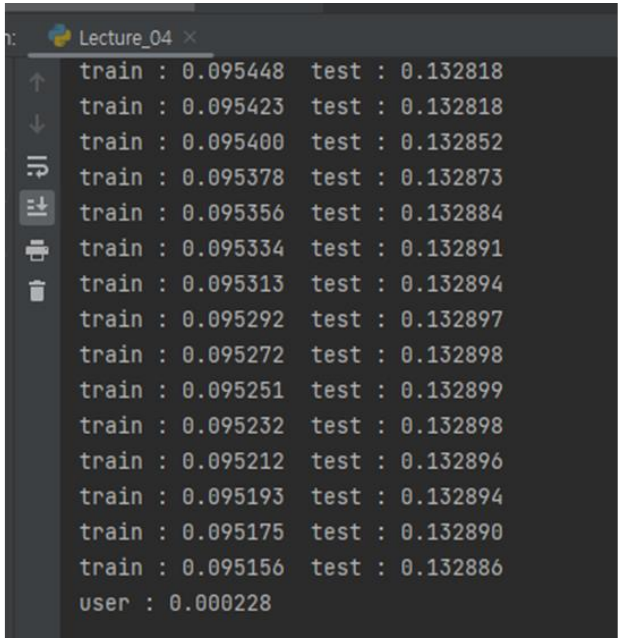


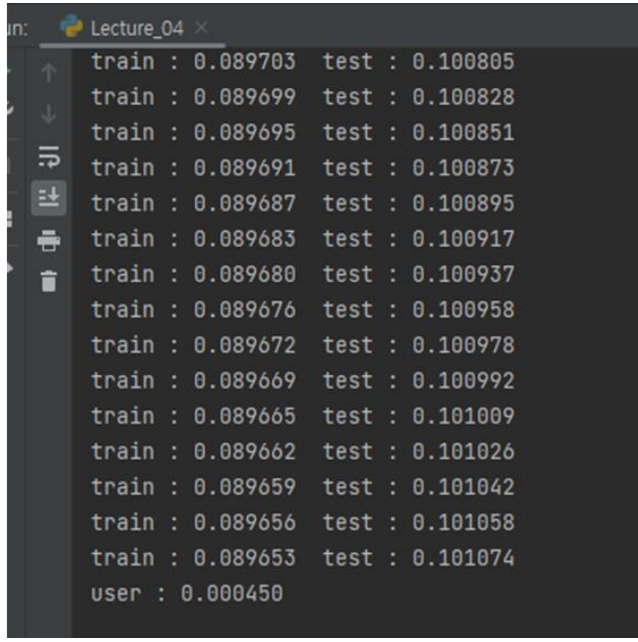

```python
# ---- network
class NeuralNet(torch.nn.Module):
    def __init__(self, input_size, hidden_size):
        super(NeuralNet, self).__init__()
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.fc1 = torch.nn.Linear(self.input_size, self.hidden_size)
        self.act1 = torch.nn.ReLU()
        self.fc2 = torch.nn.Linear(self.hidden_size, 1)
        self.act2 = torch.nn.Sigmoid()

    def forward(self, input_tensor):
        fc1 = self.fc1(input_tensor)
        act1 = self.act1(fc1)
        fc2 = self.fc2(act1)
        output = self.act2(fc2)
        return output
```

Neural Network

- Example of neural network (Multiple layers)
  - Redesign network
    - **Hidden layer : 2 / 5 / 10**







Neural Network

- Practice : Add one more layer