<삼성 AI전문가 교육과정> 실습

Lab 2: Logic Gate

서울대학교 BI LAB (장병탁 교수님)
김정현, 신수연
2023.06.05

Biointelligence Laboratory
Dept. of Computer Science and Engineering
Seoul National University

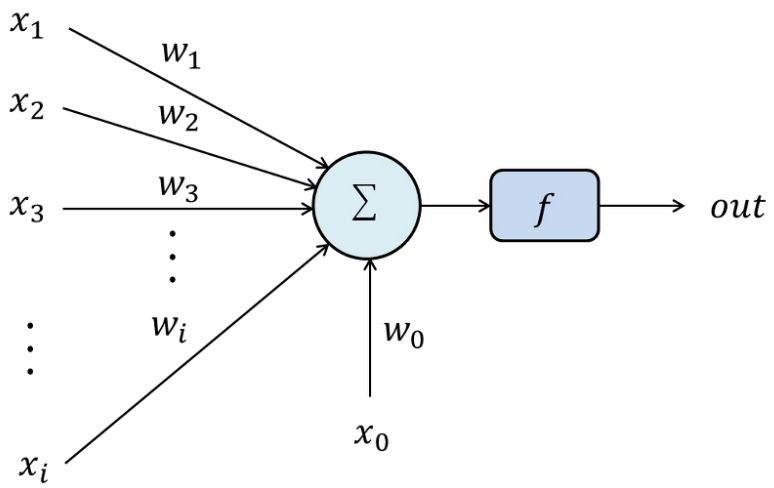
BI
BIOINTELLIGENCE

Logic Gate Example

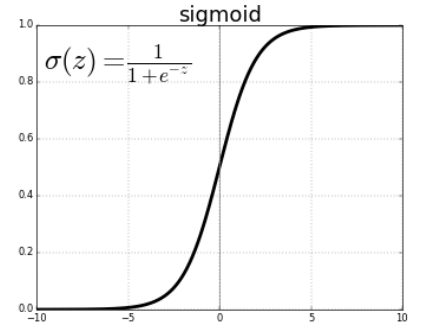
Logic Gate Example

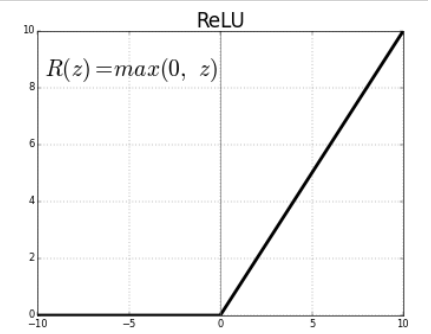
- **Single Perceptron Separator**

$$net = \sum_{i}^{N} w_i x_i + w_0 x_0$$



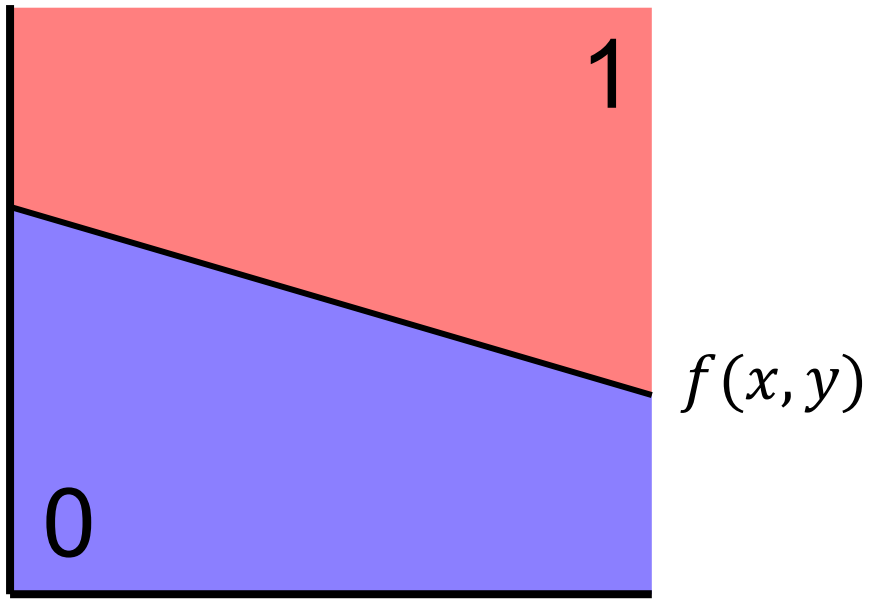
- Activation function $g(x)$: sigmoid, relu, etc.





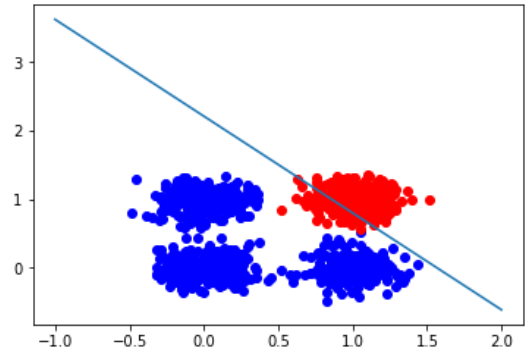
Logic Gate Example

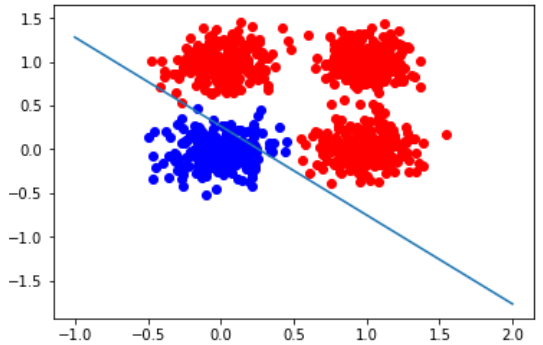
- **Single Perceptron Separator**
  - 단일 퍼셉트론 : $\text{out} = f(x,y) = g(ax + by - c \times 1) = g(ax + by - c)$
  - 공간 분할 측면에서의 퍼셉트론 (Sigmoid)

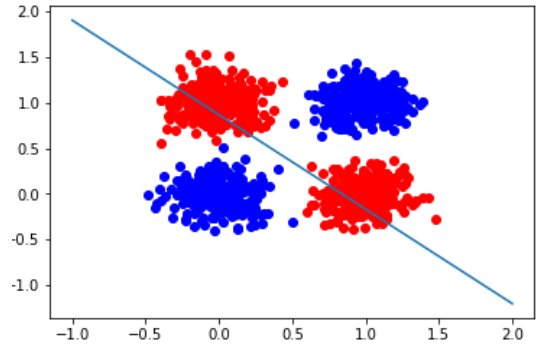


Logic Gate Example

- **Single Perceptron Separator**
  - 문제 : 단순한 Boolean 함수 근사(AND, OR, XOR)
  - 입력
    - 노이즈를 포함하는 0 혹은 1의 값을 가지는 x, y
  - 출력
    - 예측된 함수의 결과 : val
  - Loss function
    - 정답과 예측된 결과의 차이를 최소화
    - $\mathcal{L} = \sum_{n=1}^{N}(val - val^{*})^{2}$





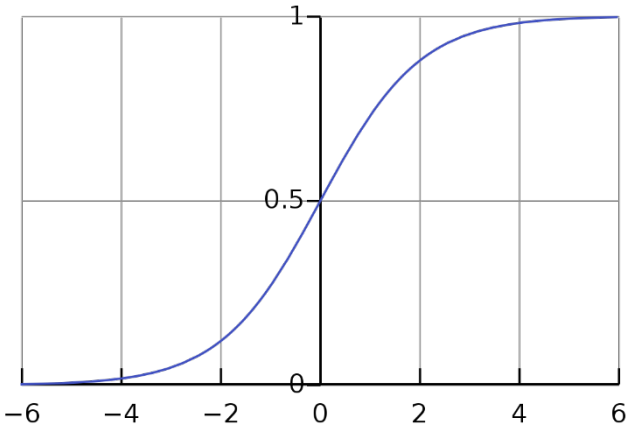


Logic Gate Example

- **Imports**
  - torch : 가장 기본적인 pytorch 모듈
  - torch.nn : 인공신경망 관련 함수/클래스 등이 선언된 모듈
  - torch.optim : optimizer가 정의된 모듈
    - Adam : Adam Optimizer
  - torch.utils.data.* : 학습 세팅을 편하게 해주기 위한 dataset의 class가 선언된 모듈들
  - matplotlib : 결과를 시각화 하는 모듈

```
import torch
import torch.nn as nn
from torch.optim import Adam
from torch.utils.data.dataset import Dataset
from torch.utils.data import DataLoader
import numpy as np
import matplotlib.pyplot as plt
```

Logic Gate Example



- **Class Separator**
  - torch.nn.Parameter(*TENSOR*)
    - 학습의 대상이 되는 변수(w)를 설정
  - torch.Tensor(*INITIAL_VALUE*)
    - Tensor의 형태로 initial value를 변환
    - Pytorch에서는 Tensor로 연산
    - Tensor.item() : 값을 가져옴
  - torch.sigmoid
    - 모든 원소에 sigmoid 함수 연산 수행
  - forward(*INPUT*)
    - 모델이 계산할 main function

```
class Separator(nn.Module):
  def __init__(self, func = None):
    super(Separator, self).__init__()
    self.a = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    self.b = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    self.c = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    if func is not None:
      self.func = func
    else:
      self.func = torch.sigmoid

  def forward(self,x,y):
    val_ = self.func(self.a * x + self.b * y - self.c)
    return val_, (self.a.item(), self.b.item(), self.c.item())
```

Logic Gate Example

- **Class DataGenerator**
  - val
  - (x,y)값이 (0,0), (0,1), (1,0), (1,1)일 때의 값을 저장함
  - x,y
  - {0,1}에서 값을 가져오고 N(0,0.16)의 노이즈를 더함

```
class DataGenerator(Dataset):

  def __init__(self,type_,length,custom=None):
    self.length=length
    if type_ == 'and':
      self.val_l = [0,0,0,1]
    elif type_ == 'or':
      self.val_l = [0,1,1,1]
    elif type_ == 'xor':
      self.val_l = [0,1,1,0]
    elif type_ == 'custom' and custom is not None:
      self.val_l = custom
    else:
      self.val_l = [0,0,0,0]

    self.dataset = []
    for i in range(length):
      x = np.random.normal(i%2,0.16)
      y = np.random.normal((i//2)%2,0.16)
      val = self.val_l[i%4]
      self.dataset.append((x,y,val))

  def get_dataset(self):
    return self.dataset

  def __len__(self):
    return self.length

  def __getitem__(self,idx):
    x,y,val = self.dataset[idx]
    return (torch.Tensor([x]),torch.Tensor([y]), torch.Tensor([val]))
```

Logic Gate Example

- **Training setup**
  - LEARNING_RATE
    - 한번의 학습을 통해 변화시키는 정도
    - 문제마다 적절한 값 설정 필요
  - BATCH_SIZE
    - 한번에 전체 데이터를 넣는 것은 크기가 너무 클 뿐 아니라, 좋지 않음
    - Batch의 크기를 설정하여 random sampling
  - NUM_EPOCHES
    - 전체 반복할 총 epoch의 수
    - 1 epoch == 데이터셋의 한 iter

```
DATASET = DataGenerator('and',1000)
LEARNING_RATE = 0.01
BATCH_SIZE = 20
NUM_EPOCHES = 20
NUM_WORKERS = 4
GRAPH_X = np.linspace(-1.0,2,2)

params = {
    'batch_size': BATCH_SIZE,
    'shuffle': True,
    'num_workers': 4,
}

dataloader = DataLoader(DATASET,**params)
model = Separator().cuda()
optimizer = Adam(model.parameters(), lr=LEARNING_RATE)
```

Logic Gate Example

- **Training part**
  - tot_loss
    - 한 epoch에서의 모든 loss를 계산
    - 결과 출력을 위함
  - Tensor.cuda()
    - 기본적으로 Tensor를 선언하면 CPU에 할당
    - .cuda()를 통해 GPU로 계산하도록 함
  - optimizer.zero_grad()
    - 각 batch마다 gradient 값을 초기화

```
for epoch in range(NUM_EPOCHES):
  tot_loss = 0
  for x, y, val in dataloader:
    x = x.cuda()
    y = y.cuda()
    val = val.cuda()
    optimizer.zero_grad()
    val_, params = model(x,y)
    loss = torch.sum(torch.pow(val-val_,2))

    loss.backward()
    optimizer.step()
    tot_loss+=loss.item()
  print("Loss : {:.5f}".format(tot_loss/len(DATASET)))

  if epoch % 5 == 4:
    for item in DATASET.get_dataset():
      x,y,val = item
      if val ==1:
        plt.scatter(x,y,c='red')
      else:
        plt.scatter(x,y,c='blue')
    plt.plot(GRAPH_X,-(GRAPH_X*params[0]-params[2])/(params[1]+1e-10))
    plt.show()
```

Logic Gate Example

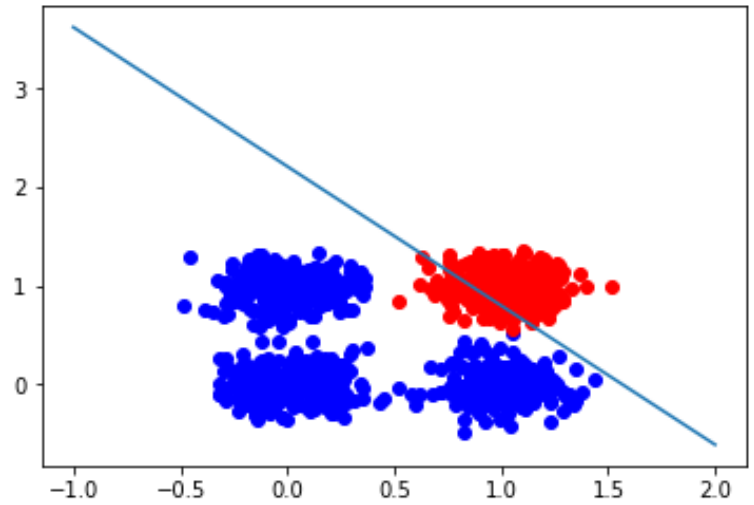
- **Training part**
  - loss.backward()
    - 계산된 tensor에 대해 역으로 계산하며 gradient를 구함
  - optimizer.step()
    - gradient를 활용하여 backpropagation을 수행, w를 update

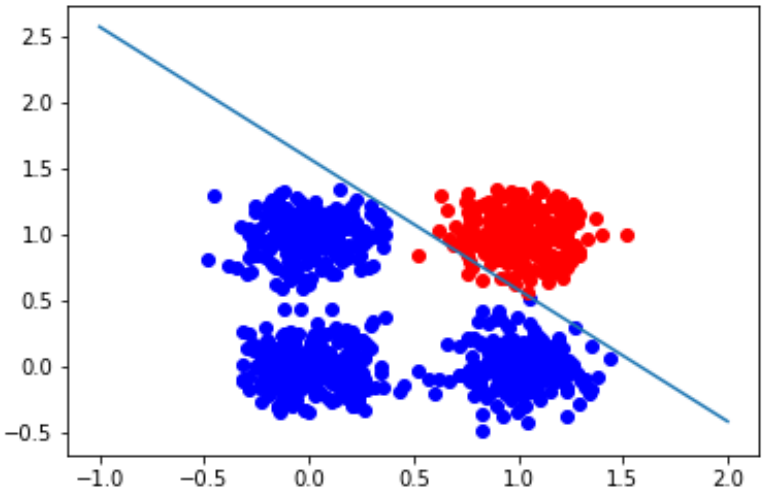
- **Test part (matplotlib)**
  - plt.scatter
    - 그래프에 산포도를 찍는 함수
  - plt.plot
    - 그래프에 선을 긋는 함수
  - plt.show
    - 그래프를 보여주는 함수

```
for epoch in range(NUM_EPOCHES):
  tot_loss = 0
  for x, y, val in dataloader:
    x = x.cuda()
    y = y.cuda()
    val = val.cuda()
    optimizer.zero_grad()
    val_, params = model(x,y)
    loss = torch.sum(torch.pow(val-val_,2))

    loss.backward()
    optimizer.step()
    tot_loss+=loss.item()
  print("Loss : {:.5f}".format(tot_loss/len(DATASET)))

  if epoch % 5 == 4:
    for item in DATASET.get_dataset():
      x,y,val = item
      if val ==1:
        plt.scatter(x,y,c='red')
      else:
        plt.scatter(x,y,c='blue')
    plt.plot(GRAPH_X,-(GRAPH_X*params[0]-params[2])/(params[1]+1e-10))
    plt.show()
```

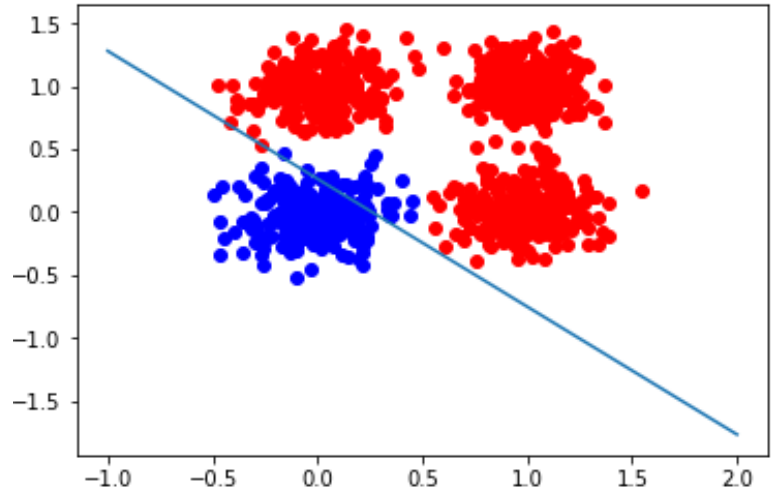
Logic Gate Example

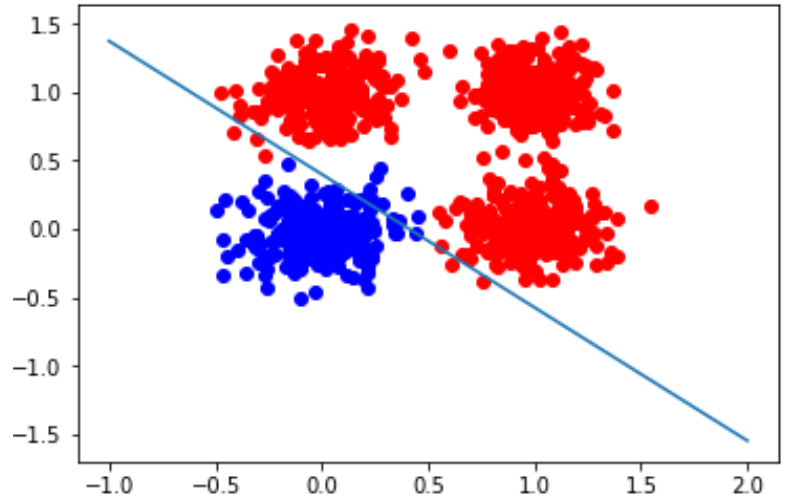
- 결과 1 (AND)





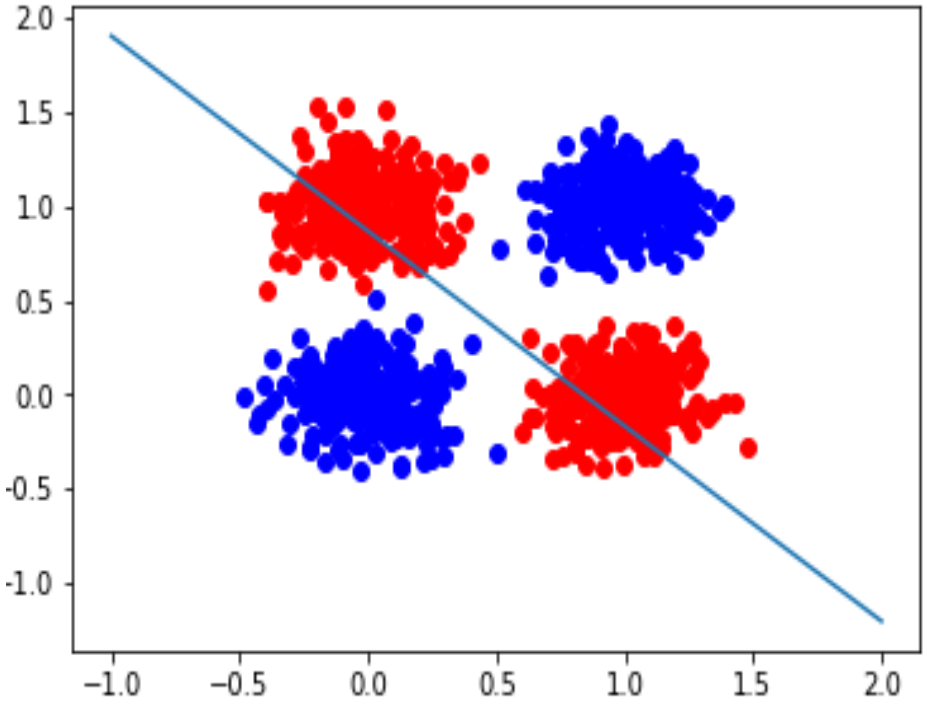
- 결과 2 (OR)

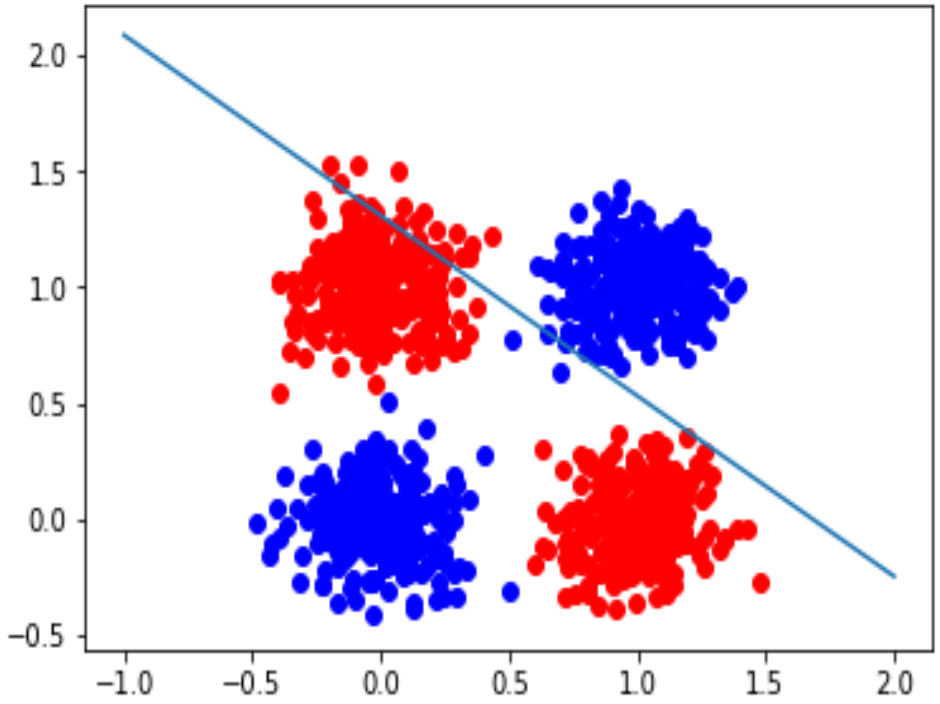




Logic Gate Example

- **결과 3 (XOR)**
  - 단일 퍼셉트론만으로는 XOR에 해당하는 함수를 근사할 수 없음
  - 해결방법?





Codes

- 실습 코드 Github 주소
  - https://github.com/JHKim-snu/AI_Expert/tree/main