NeuralNet 101

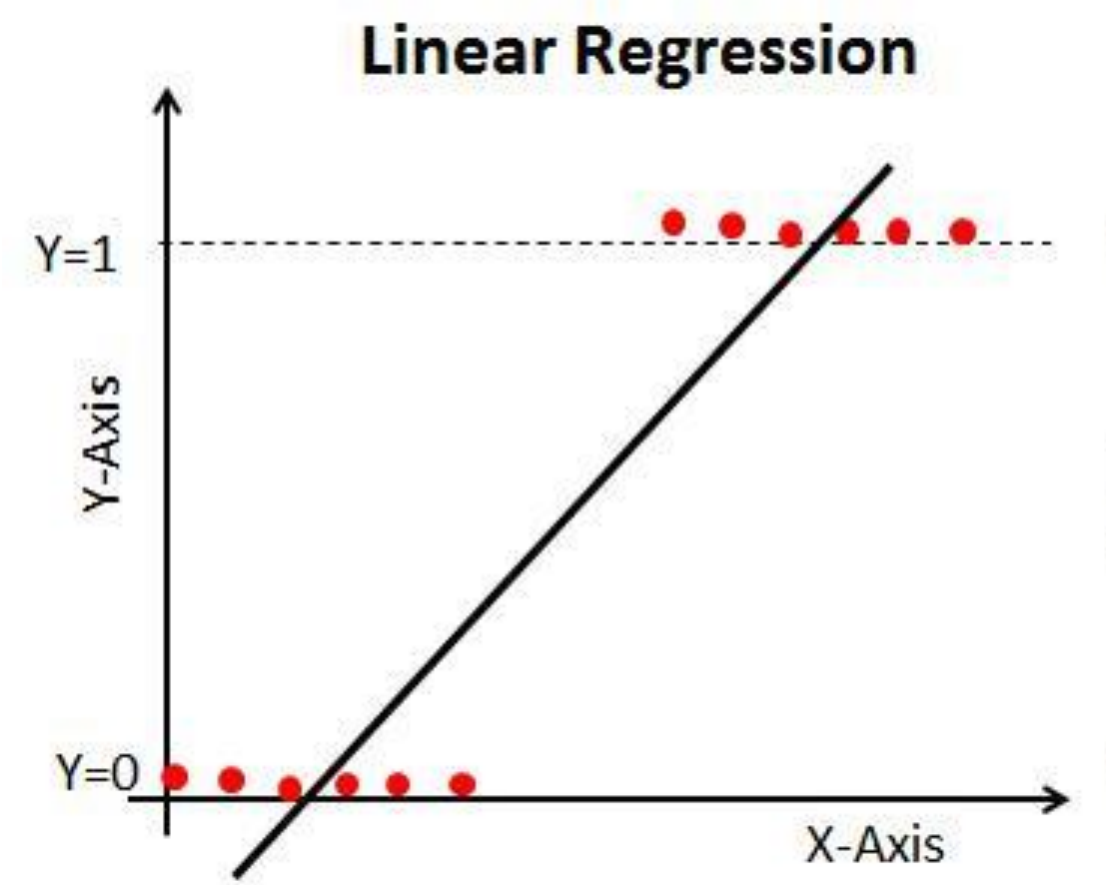
4. Logistic Regression

What we have learned in linear regression...

- Can calibrate parameters of linear based mean value function by L2 distance and gradient descent.
- The input and output are all continuous

But when the output data is only 0 or 1, can linear regression represent the relation of dataset?

Let's look this picture



Referenced from: https://satisfactoryplace.tistory.com/322

It clearly shows that it is inappropriate to find an answer to describe these kinds of problems

we need other point of view, at least

Let's think based on the point of probability, then

- Why? - outputs are only 0 and 1, so it can be interpreted as True or False
- How? - based on likelihood

- Why likelihood? - because we do not know entirely about all cases.

Then, we need to change the linear function to be fit in 0~1

Hence, we define a probability function as...

$$\log \frac{p}{1-p} = WX + b$$

And this function can be changed into under below process

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

$$\log\left(\frac{p}{1-p}\right) = WX + b$$

$$\frac{p}{1-p} = c^{WX+b}$$

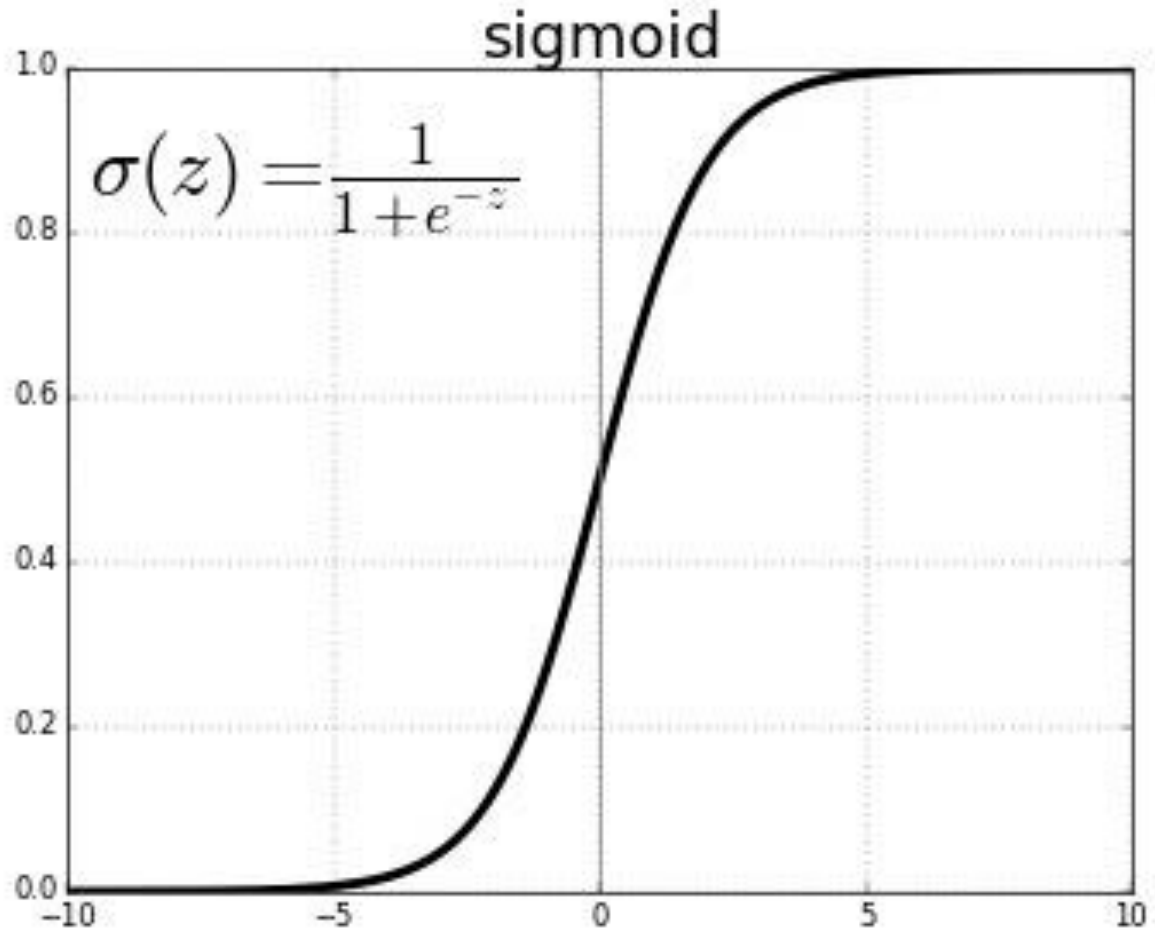
$$\left(\frac{p}{1-p}\right)^{-1} = \frac{1-p}{p} = \left(c^{WX+b}\right)^{-1} = \frac{1}{c^{WX+b}}$$

$$\frac{1-p}{p} = \frac{1}{p} - 1 = \frac{1}{c^{WX+b}}$$

$$\frac{1}{p} = \frac{1}{c^{WX+b}} + 1 = \frac{1 + c^{WX+b}}{c^{WX+b}}$$

$$\therefore p = \frac{c^{WX+b}}{1 + c^{WX+b}}$$

Therefore, we can get a generalized form of probability function **σ(z), z = WX + b** (when W = 0, b=0)

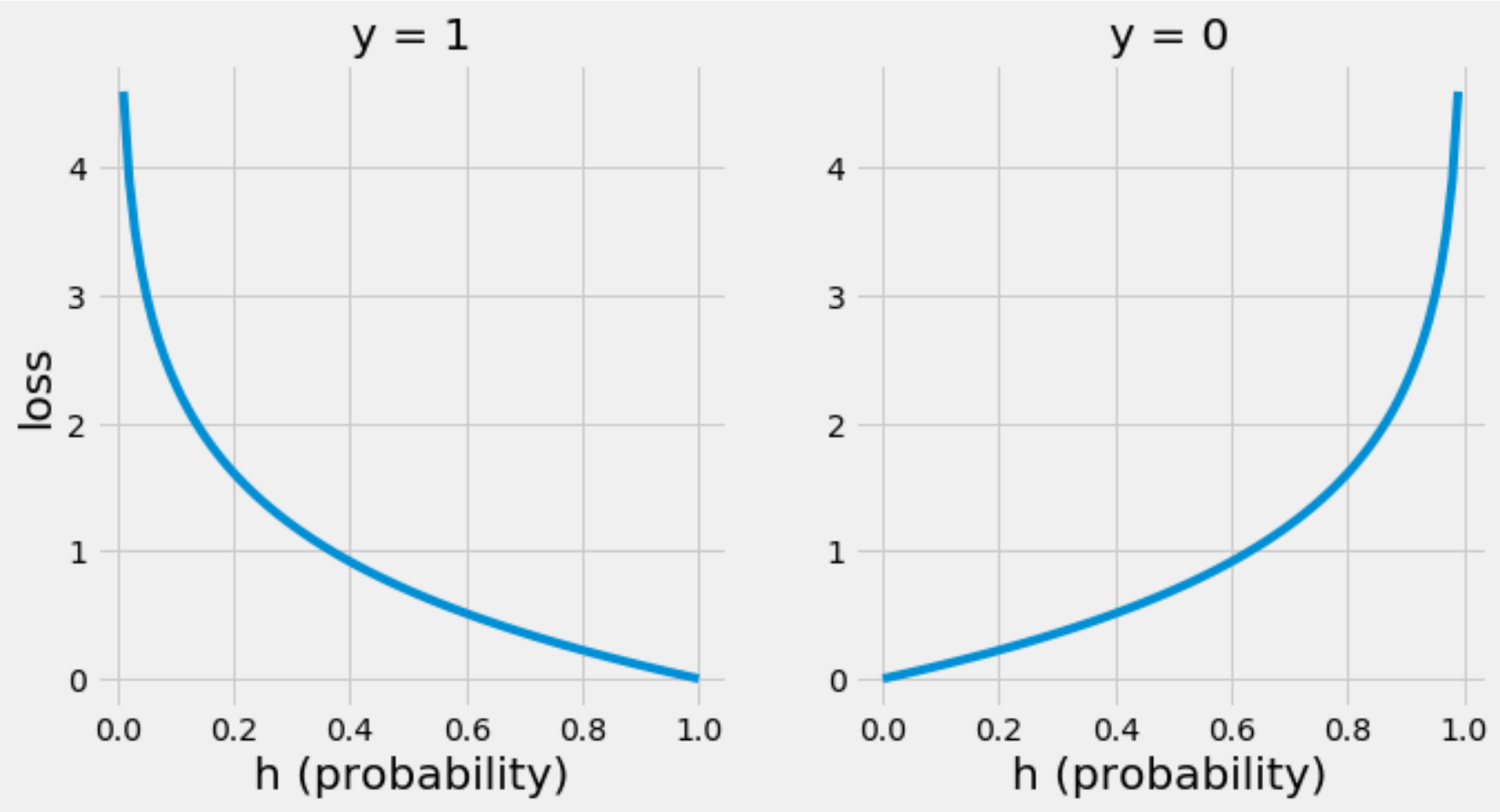


Referenced from: https://sanghyu.tistory.com/102

Well done… but how to calibrate w and b?

If we just use Mean Squared Error, then it is hard to represent the loss because real loss is 0 or infinity but the loss function is quadratic.

Therefore, we need to define new loss function L like next slide



Referenced from: https://hleecaster.com/ml-logistic-regression-concept/

And mathematically it can be written as…

$$L(b, W) = \prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$$

$$l(b, W) = \sum_{i=1}^{n} y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))$$

$$= \sum_{i=1}^{n} \log(1-p(x_i)) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1-p(x_i)}$$

$$= \sum_{i=1}^{n} \log(1-p(x_i)) + \sum_{i=1}^{n} y_i (b_i + x_i \bullet W_i)$$

$$= \sum_{i=1}^{n} -\log(1+c^{b_i + x_i \cdot W_i} + \sum_{i=1}^{n} y_i (b_i + x_i \bullet W_i)$$

And therefore, the gradient of this loss is equal as under below formula

$$\frac{\partial l}{\partial W_j} = -\sum_{i=1}^{n} \frac{1}{1 + c^{b_i + x_i \cdot W_i}} c^{b_i + x_i \cdot W_i} x_{ij} + \sum_{i=1}^{n} y_i x_{ij}$$

$$= \sum_{i=1}^{n} (y_i - p(x_i; b_i, W_i)) x_{ij}$$

Therefore, we can optimize logistic regression based on gradient descent method also!

References

- Cosma Shalizi, Undergraduate Advanced Data Analysis Chapter 12 LectureNote,2012, https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf

- 김성훈, 모두를 위한 딥러닝 Logistic (Regression) Classification, https://hunkim.github.io/ml/lec5.pdf

Lab session

In Lab session…

- Logistic regression with PyTorch
- Implementation with class

Logistic regression with mathematical operation

$$H(x) = \frac{1}{1 + e^{-(x \cdot W + b)}}$$

$$L(x) = -\frac{1}{m}\sum \left(y \times \log\left(H(x)\right) + (1 - y) \times \log\left(1 - H(x)\right)\right)$$

Logistic regression with mathematical operation

$$H(x) = \frac{1}{1 + e^{-(x \cdot W + b)}}$$

```
hypothesis = 1 / (1 + torch.exp(-(x_train.matmul(W) + b)))
```

```
hypothesis = torch.sigmoid(x_train.matmul(W) + b)
```

Logistic regression with mathematical operation

$$L(x) = -\frac{1}{m}\sum\left(y \times \log\left(H(x)\right) + (1-y) \times \log\left(1 - H(x)\right)\right)$$

```
losses = -(y_train * torch.log(hypothesis) + (1 - y_train) * torch.log(1 - hypothesis))
cost = losses.mean()
```

```
import torch.nn.functional as F
cost = F.binary_cross_entropy(hypothesis, y_train)
```

Logistic regression with mathematical operation

Lecture	Lab	S/U (S=1)
10	10	1
5	7	1
10	0	0
1	6	0

Lab03-ex-data.csv

Logistic regression with mathematical operation

```
filename = "Lab03-ex-data.csv"

df = pd.read_csv(filename)
x_train = torch.tensor(df.loc[:, ['lecture','lab']].values.tolist(), dtype=torch.float32)
y_train = torch.tensor(df['su'].values.tolist(), dtype=torch.float32).reshape(-1, 1)

W = torch.zeros((2,1), requires_grad=True, dtype=torch.float32)
b = torch.zeros(1, requires_grad=True, dtype=torch.float32)

optimizer = torch.optim.SGD([W, b], lr=0.1)

nb_epochs = 5000
for epoch in range(nb_epochs):
    hypothesis = torch.sigmoid(x_train.matmul(W) + b)
    cost = F.binary_cross_entropy(hypothesis, y_train)

    optimizer.zero_grad()
    cost.backward()
    optimizer.step()

    if epoch%100==0:
        print(cost.item())
```

Adjust learning rate for appropriate gradient descent

Logistic regression with mathematical operation

```
31  x_predict = torch.tensor([[3,6]], dtype=torch.float32)
32  y_predict = torch.sigmoid(x_predict.matmul(W) + b)
33  print(y_predict)#0.7283
34  prediction = (y_predict >= torch.Tensor([0.5])).int()
35  print(prediction)#1
```

Implement with class

```
6   class BinaryClassifier(nn.module):
7       def __init__(self):
8           super.__init__()
9           self.linear = nn.Linear(2,1)
10          self.sigmoid = nn.Sigmoid()
11
12      def forward(self, x):
13          return self.sigmoid(self.linear(x))
14  model = BinaryClassifier()
```

Implement with class

```
22  optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
23
24  nb_epochs = 5000
25  for epoch in range(nb_epochs):
26      hypothesis = model(x_train)
27      cost = F.binary_cross_entropy(hypothesis, y_train)
28
29      optimizer.zero_grad()
30      cost.backward()
31      optimizer.step()
32
33      if epoch%100==0:
34          print(cost.item())
```

Lab02 Problems

- Check github vlab-kaist
- https://github.com/vlab-kaist/NeuralNet101
- Problems>Lab03

- If you solved all of problems, please make issue.
- Title : ‘Section_Name_lab_week3’
  (ex. ‘B_장유진_lab_week3’)

References.

- 모두를 위한 딥러닝 시즌2 – PyTorch https://www.youtube.com/playlist?list=PLQ28Nx3M4JrhkqBVIXg-i5_CVVoS1UzAv, Lab05