NeuralNet 101

5. Softmax and Cross-entropy Loss

Announcement

- I encouraged you to solve the lab problems and assignments but almost everyone does not make github issues

- So, I made the policy which determines the portion of fee of Lab dinner, based on the progression of lab problems & Assignment

- It can be hard, so I make a group by random to solve the problem together, and the progression will be counted by the median value of a group

Group allocation

- Group A: 기민준, 이진욱, 전채훈
- Group B: 이현서, 조건우, 김지나
- Group C: 이지혁, 송재훈, 강 건
- Group D: 박성빈, 박준하, 김민규
- Group E: 최유민, 임현진, 유원호

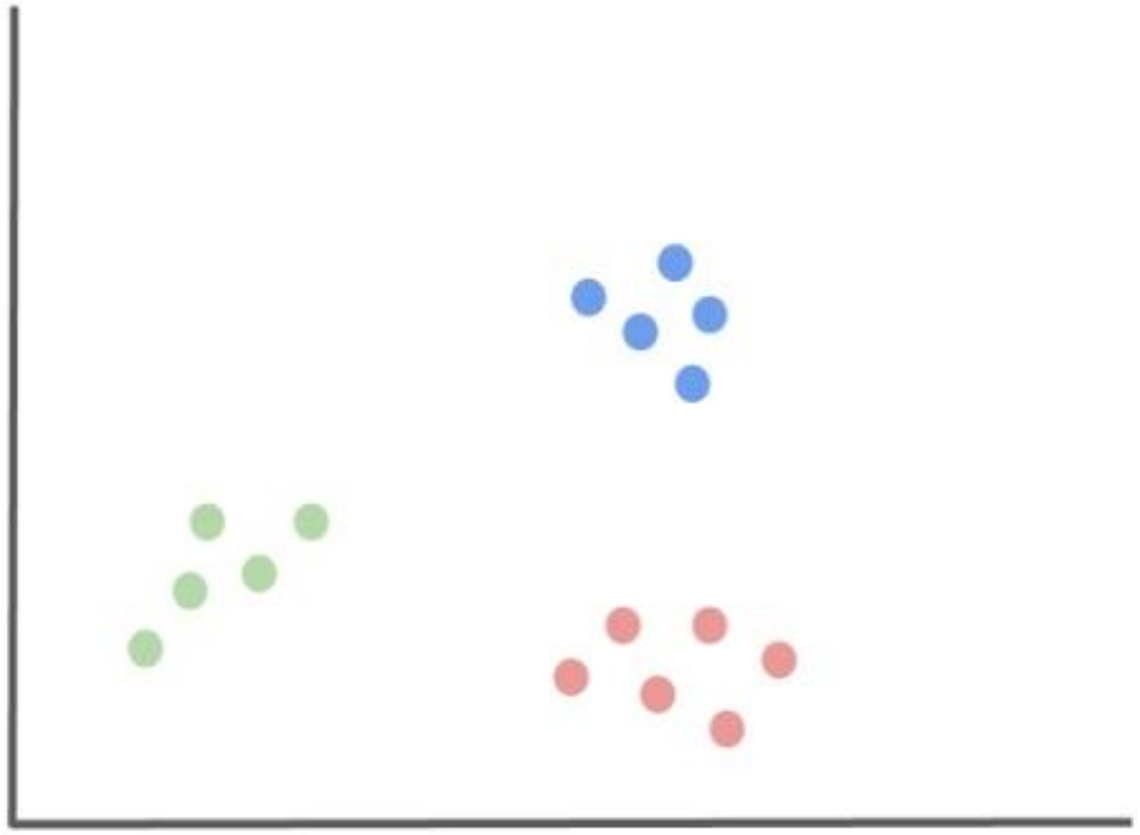
The formula which determines the portion of buying Lab dinner

- Non-determined, but will be

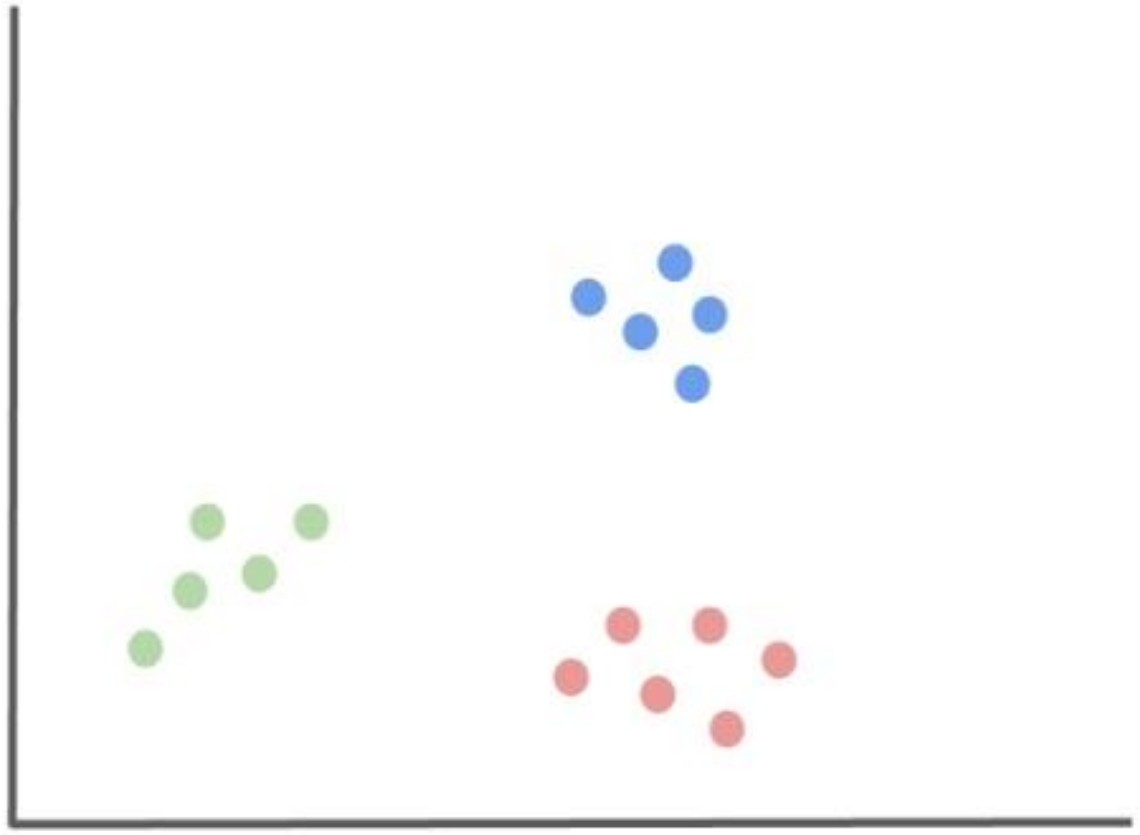
So, let us see what have learned in lecture 4?

- Logistic regression: which can separate the two different kinds of data
- Sigmoid function: the function that approximates the two different kinds of data

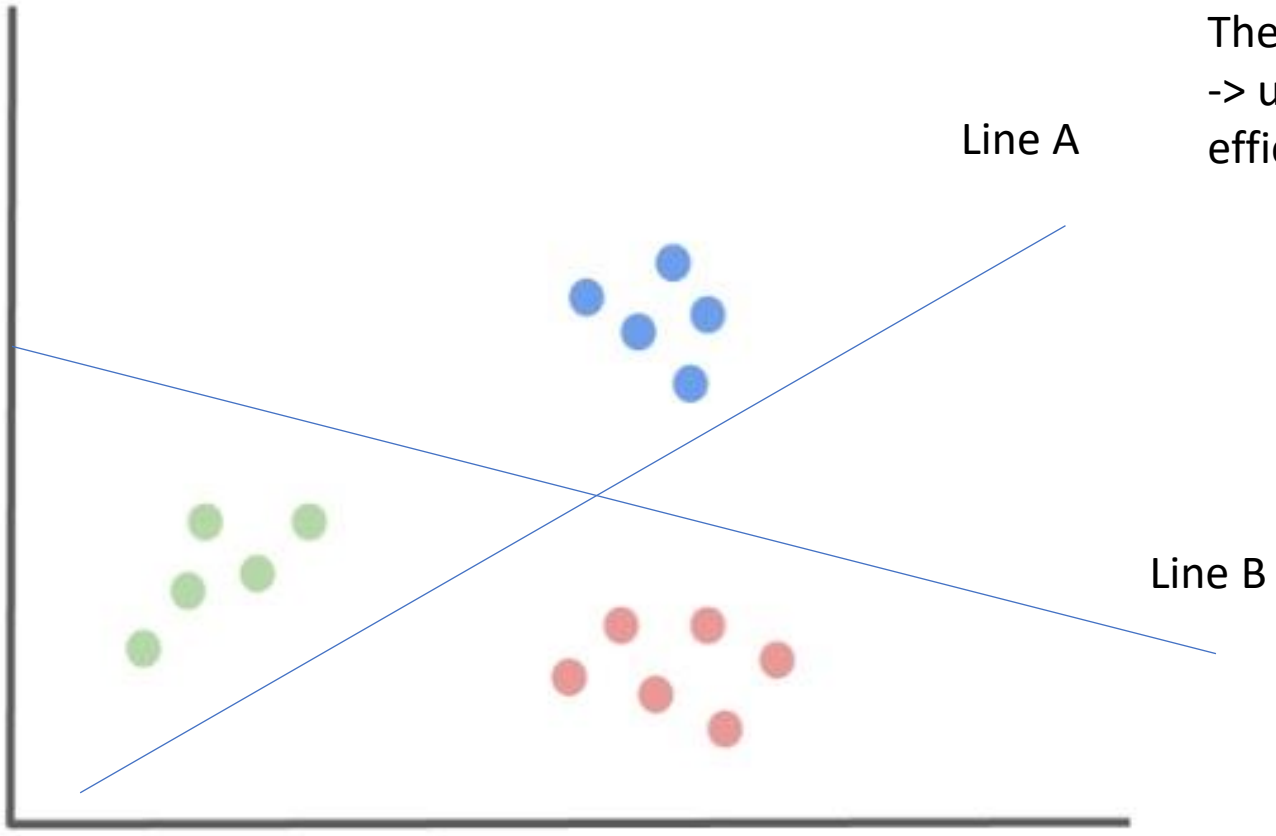
But how about in these kinds of multiple-class?



Can we separate the data by multiple class by sigmoid?

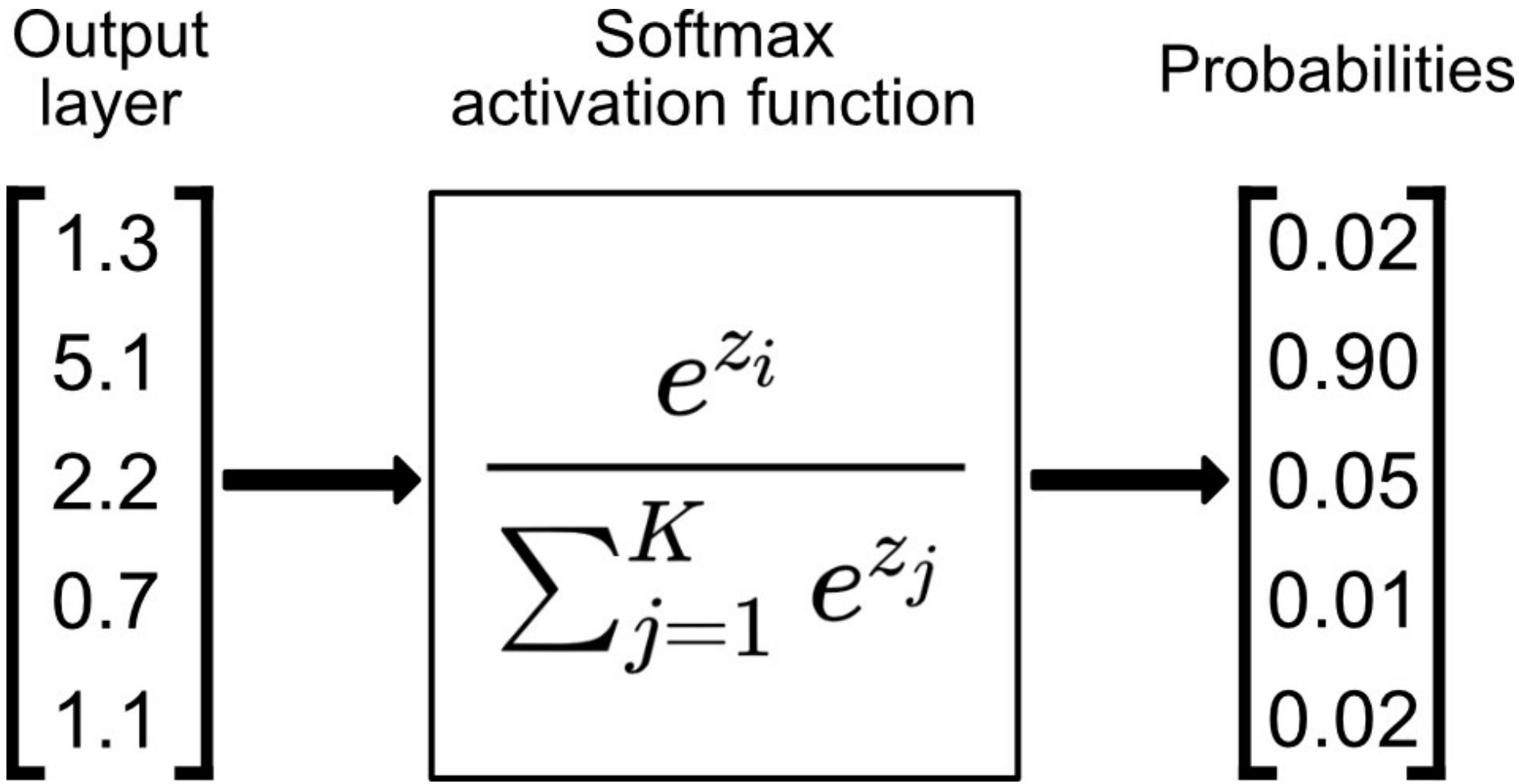


If you draw lines on these data and set the condition like..



The class C is over the line A and B
-> using multiple sigmoid function but it is not efficient

So we set the new function, softmax



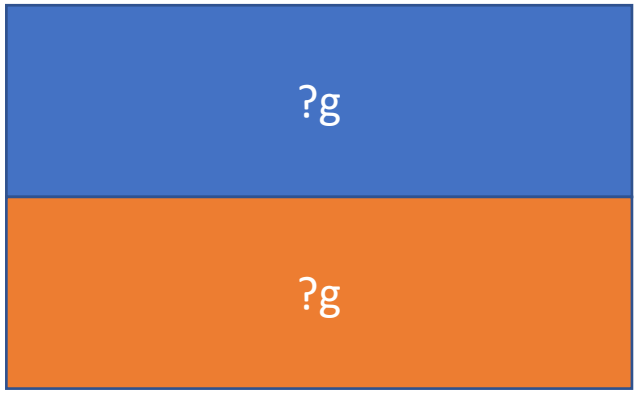
And it is multiple-class, so we need new loss

It is cross-entropy loss

Let's think a coin-toss

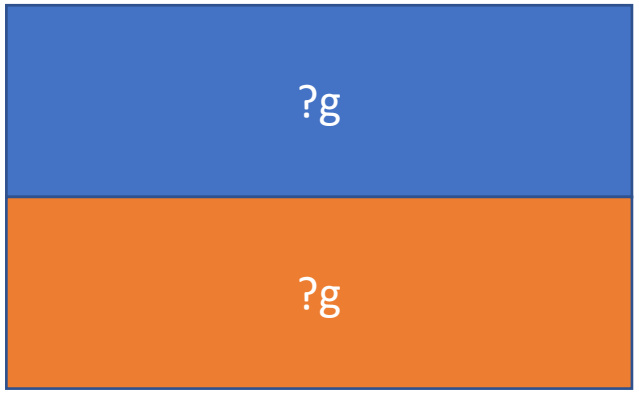
With a constraint…

- We do not know the difference of mass between front and rear



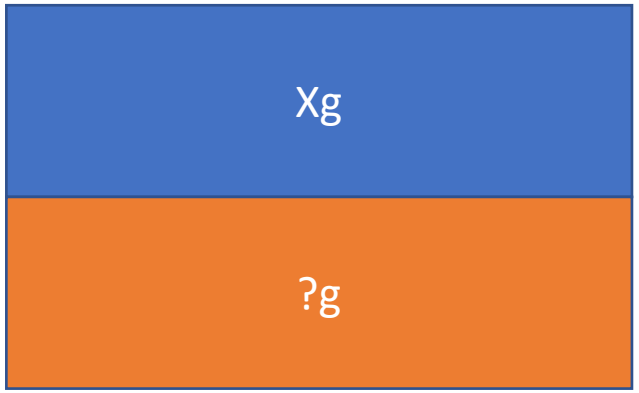
The process to get cross-entropy loss

- Then first, we can set ?:? As 0.5 each.



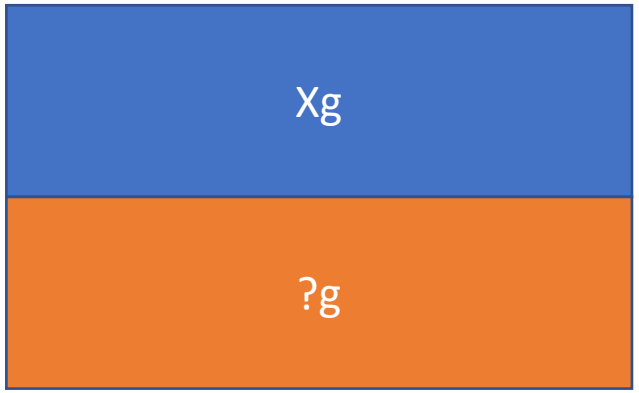
The process to get cross-entropy loss

- Then we toss the coin and check which side is upside, and set error only -ln(X)



Why?

- Because it only happens when blue side is upside.



And we can expand this things like dice.

Therefore, we can get the cross-entropy loss on particular input as:

$$-\sum_{c=1}^{C} q(y_c) log(p(y_c))$$

And the mean value of these losses are equal as:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} L_{ic} log(P_{ic})$$

References

- Nonzerok@gmail.com, 크로스 엔트로피 손실함수 기초,

딥러닝을 위한 신경망 기초,

http://kocw.xcache.kinxcdn.com/KOCW/document/2017/kumoh/kojaepil0302/8.pdf

- 김성훈, Softmax Regression (Multinomial Logistic Regression), 모두를 위한 딥러닝, https://hunkim.github.io/ml/lec6.pdf
- Rcchun, [머신러닝] 크로스 엔트로피(cross entropy), https://velog.io/@rcchun/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-%ED%81%AC%EB%A1%9C%EC%8A%A4-%EC%97%94%ED%8A%B8%EB%A1%9C%ED%94%BCcross-entropy

Lab session

Softmax

```
model = nn.Linear(4, 3, bias=True)
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for epoch in range(num_epochs):
    optimizer.zero_grad()
    hypothesis = model(x_train.view(-1, 4))
    cost = nn.CrossEntropyLoss()(hypothesis, y_train)
    cost.backward()
    optimizer.step()
    print('Epoch : %d, cost = %.9f' % (epoch, cost))
```

How to use MNIST data

```
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.utils.data import DataLoader

mnist_train = dsets.MNIST(root="MNIST_data/", train=True, transform=transforms.ToTensor(), download=True)
mnist_test = dsets.MNIST(root="MNIST_data/", train=False, transform=transforms.ToTensor(), download=True)

data_loader = DataLoader(dataset=mnist_train, batch_size=batch_size, shuffle=True, drop_last=True)
```

Install python package – “torchvision”

Lab Problems

1. Train MNIST softmax model with MNIST train data.

2. Train softmax model with “Iris.csv” train data.