Softmax Classification

상태	Basic ML
담당자	

Softmax Classifcation

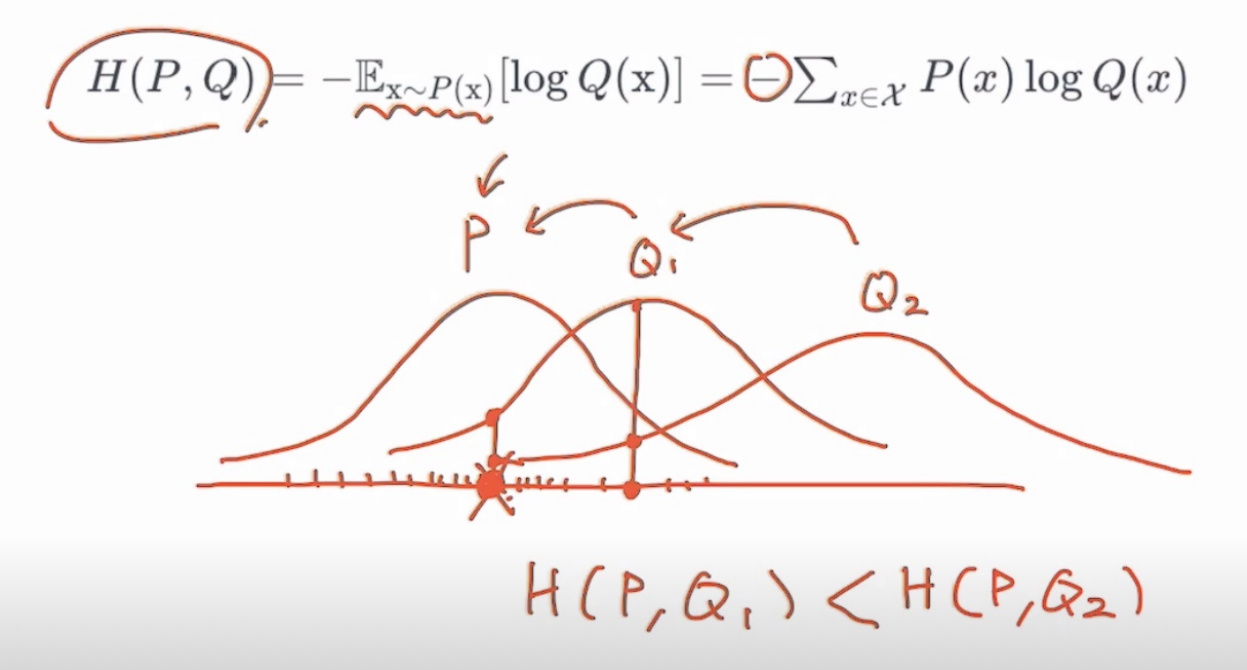
> Multinomial Classification을 위한 방법

> Logistic regression의 연장선, 각각의 값이 max가 될 수 있는 확률을 soft하게 리턴하라.

discrete probability distribution을 따르기 때문에 특정 point에서의 확률을 계산할 수 있다.
$P(class = i) = \frac{e^i}{\sum e^i}$ 이며 모든 확률의 합은 1이다.

Cross-Entropy

> 두 개의 확률분포가 주어졌을 때, 이 둘이 얼마나 비슷한지를 나타내는 것



P라는 확률분포를 따르는 x를 뽑으면, 그 x가 확률분포 Q_1, Q_2 상에서 어떤 확률로 등장할 것인지 알 수 있다.

이것을 주어진 $H(P,Q)$에 넣으면 위 그림의 경우에는 Q에서의 확률에 로그를 취한 뒤 다시 음수를 곱해주게 된다. 따라서 $Q_1(x) > Q_2(x)$이지만 $H(P,Q_1) < H(P,Q_2)$가 된다.

따라서 Cross-Entropy를 줄인다는 것은 곧 모델의 확률분포함수 Q가 비교대상인 함수 P에 가까워진다는 것을 의미하며, 이는 우리가 지향하는 모델 업데이트와 일치한다.

PyTorch

```
x_train = [[1, 2, 1, 1],
           [2, 1, 3, 2],
           [3, 1, 3, 4],
           [4, 1, 5, 5],
           [1, 7, 5, 5],
           [1, 2, 5, 6],
           [1, 6, 6, 6],
           [1, 7, 7, 7]]

y_train = [2, 2, 2, 1, 1, 1, 0, 0]

x_train = torch.FloatTensor(x_train)
y_train = torch.LongTensor(y_train)


class SoftmaxClassification(nn.Module):
```

```
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(4, 3) # 8 x 4 가 4 x 3을 지나 8 x 3으로

    def forward(self, x):
        return self.linear(x)

model = SoftmaxClassification() # model에 x_train을 대입하면 8 x 3


optimizer = optim.SGD(model.parameters(), lr=0.1)

nb_epochs = 1000
for epoch in range(nb_epochs + 1):

    # H(x) 계산
    prediction = model(x_train)

    # Loss(Cost) 계산
    cost = F.cross_entropy(prediction, y_train)

    # Loss로 H(X) 개선
    optimizer.zero_grad()
    cost.backward()
    optimizer.step()

    if epoch % 100 == 0:
        print('Epoch : {:4d}/{} Cost : {:.6f}'.format(
            epoch, nb_epochs, cost.item()
        ))

>>
Epoch :    0/1000 Cost : 3.246066
Epoch :  100/1000 Cost : 0.703490
Epoch :  200/1000 Cost : 0.620850
Epoch :  300/1000 Cost : 0.563644
Epoch :  400/1000 Cost : 0.513388
Epoch :  500/1000 Cost : 0.465922
Epoch :  600/1000 Cost : 0.419682
Epoch :  700/1000 Cost : 0.373925
Epoch :  800/1000 Cost : 0.328385
Epoch :  900/1000 Cost : 0.283776
Epoch : 1000/1000 Cost : 0.247032
```

▼ 왜 `SoftMaxClassification` 클래스를 구현할 때 hypothesis 함수로 linear만 사용하는가?

Logistic regression을 사용한 binary classification을 할 때에는 다음과 같은 흐름이었다.

1. `sigmoid(linear())` 형태로 hypothesis 함수 작성
2. `F.binary_cross_entropy`로 loss를 계산

3. Loss를 gradient 하여 hypothesis 함수를 업데이트

Softmax classification을 할 때에는 다음과 같은 흐름이었다.

1. `linear()` 형태로 hypothesis 함수 작성
2. `F.cross_entropy`로 loss 계산
3. Loss를 gradient 하여 hypothesis 함수를 업데이트

왜 `softmax(linear())` 형태가 아닌 것일까? 이유는 loss function의 차이!

- `F.binary_cross_entropy` : Function that measures the Binary Cross Entropy between the target and input probabilities.
- `F.cross_entropy` : This criterion combines log_softmax and nll_loss in a single function.

Loss를 계산하기에 앞서 log softmax를 적용해 주기 때문이다.