3장. 선택 분류 : 철판 불량 상태 분류 신경망

1. 개요

선택 분류를 이용한 철판 불량 상태 분류 신경망 실험에서는 소프트맥스 함수, 크로스엔트로피의 결과와 시그모이드 함수,평균 제곱 오차의 결과를 비교하는 작업을 진행하였다. 이 실험에서 사용되는 불량 철판 데이터셋은 육안으로 확인 가능한 철판들의 각종 특징값 27필드와 원-핫 벡터 형태의 불량 유형 정보인 7필드로 구성되어 있다. 선택 분류 문제에서 소프트맥스 함수, 크로스엔트로피와 시그모이드 함수, 평균 제곱 오차의 차이를 알아보고 소프트맥스 함수를 많이 사용하는 이유에 대해 분석해보고자 한다.

1-1. 개념

시그모이드 함수와 소프트맥스 함수의 관계

: 시그모이드 함수는 입출력 벡터 크기를 2에서 1로 줄인 변형된 소프트맥스 함수에 해당한다. 거짓인 경우에 대한 로짓값을 0으로 전제하고 계산하게 된다. (로짓값은 상대적인 값이므로 하나를 고정시켜도 상관이 없다.) 소프트맥스 함수의 경우 출력의 합이 1이지만, 시그모이드 함수의 경우 출력 값이 0 - 1 사이에 나타날 뿐, 1이 아닐 수도 있다. 즉, 후보 항목들의 로짓값에 소프트맥스 함수 대신 시그모이드 함수를 적용하면 각 항목이 답으로 선택될 확률이 구해지지만 그 합이 1로 맞취지진 않는다. 개별 확률로 구해지게 되고, 그 중 가장 큰 확률을 선택하면 되는 것이다.

cf) 시그모이드 함수 : $\sigma(x) = \frac{1}{1+e^{-x}}$, 소프트맥스 함수 : $y_i = \frac{e^{xi}}{e^{x1}+\ldots+e^{xn}}$

크로스 엔트로피 : 두 개의 확률 분포 유사도의 정량화 방법으로, 한 확률 분포로 다른 확률 분포에서의 정보량을 가중한다.

$$H_{p,q}(X) = -\sum_{i=1}^{N} p(x_i)\log q(x_i)$$

p는 true probability로써 true label에 대한 분포(정답 정보)를 나타내고, q는 현재 예측 모델의 추정값에 대한 분포(신경망 추정 결과)를 나타낸다.

2. 학습결과

2.1 소프트맥스 함수와 크로스 엔트로피를 사용한 경우

(1) 모델 1

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - 학습률(learning rate) : 0.001
데이터 정확도	45.5%
결과 화면	steel_exec() Epoch 1: loss=15.984, accuracy=0.306/0.320 Epoch 2: loss=15.509, accuracy=0.326/0.197 Epoch 3: loss=15.984, accuracy=0.306/0.348 Epoch 4: loss=15.004, accuracy=0.348/0.197 Epoch 5: loss=15.286, accuracy=0.336/0.202 Epoch 6: loss=15.390, accuracy=0.332/0.440 Epoch 7: loss=15.509, accuracy=0.326/0.442 Epoch 8: loss=15.628, accuracy=0.321/0.455 Epoch 9: loss=15.360, accuracy=0.333/0.322 Epoch 10: loss=15.316, accuracy=0.335/0.455 Final Test: final accuracy = 0.455

(2) 모델 2

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - 학습률(learning rate) : 0.0001
데이터 정확도	18.9%
결과 화면	LEARNING_RATE = 0.0001 steel_exec() Epoch 1: loss=16.471, accuracy=0.284/0.110 Epoch 2: loss=15.479, accuracy=0.328/0.414 Epoch 3: loss=15.509, accuracy=0.326/0.402 Epoch 4: loss=15.316, accuracy=0.335/0.432 Epoch 5: loss=15.479, accuracy=0.328/0.338 Epoch 6: loss=14.796, accuracy=0.357/0.332 Epoch 7: loss=15.346, accuracy=0.334/0.215 Epoch 8: loss=15.524, accuracy=0.326/0.164 Epoch 9: loss=15.375, accuracy=0.332/0.281 Epoch 10: loss=15.331, accuracy=0.334/0.189 Final Test: final accuracy = 0.189

2.2 시그모이드 함수와 평균 제곱 오차를 사용한 경우

다음과 같이 시그모이드 함수와 평균 제곱 오차를 포함하도록 코드를 일부 수정하여 실험을 진행하였다.

```python
def forward_neuralnet(x):
    global weight, bias
    output = sigmoid(np.matmul(x, weight) + bias)
    return output, x
```

```python
def relu(x):
    return np.maximum(x, 0)

def sigmoid(x):
    return np.exp(-relu(-x)) / (1.0 + np.exp(-np.abs(x)))
```

```python
def forward_postproc(output, y):
    diff = output - y
    square = np.square(diff)
    loss = np.mean(square)
    return loss, diff

def backprop_postproc(G_loss, diff):
    shape = diff.shape

    g_loss_square = np.ones(shape) / np.prod(shape)
    g_square_diff = 2 * diff
    g_diff_output = 1

    G_square = g_loss_square * G_loss
    G_diff = g_square_diff * G_square
    G_output = g_diff_output * G_diff

    return G_output
```

(1) 모델 3

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - 학습률(learning rate) : 0.001
데이터 정확도	32.8%
결과 화면	steel_exec() Epoch 1: loss=0.200, accuracy=0.408/0.228 Epoch 2: loss=0.192, accuracy=0.476/0.637 Epoch 3: loss=0.194, accuracy=0.499/0.649 Epoch 4: loss=0.193, accuracy=0.476/0.606 Epoch 5: loss=0.192, accuracy=0.499/0.290 Epoch 6: loss=0.197, accuracy=0.504/0.274 Epoch 7: loss=0.192, accuracy=0.506/0.552 Epoch 8: loss=0.189, accuracy=0.494/0.274 Epoch 9: loss=0.190, accuracy=0.473/0.514 Epoch 10: loss=0.191 accuracy=0.510/0.328 Final Test: final accuracy = 0.328

(2) 모델 4

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - 학습률(learning rate) : 0.0001
데이터 정확도	21.7%
결과 화면	LEARNING_RATE = 0.0001 steel_exec() Epoch 1: loss=0.203, accuracy=0.270/0.151 Epoch 2: loss=0.196, accuracy=0.315/0.422 Epoch 3: loss=0.188, accuracy=0.330/0.430 Epoch 4: loss=0.191, accuracy=0.315/0.402 Epoch 5: loss=0.187, accuracy=0.330/0.192 Epoch 6: loss=0.190, accuracy=0.333/0.182 Epoch 7: loss=0.185, accuracy=0.335/0.366 Epoch 8: loss=0.192, accuracy=0.327/0.182 Epoch 9: loss=0.191, accuracy=0.314/0.340 Epoch 10: loss=0.191, accuracy=0.337/0.217 Final Test: final accuracy = 0.217

3.결론

<데이터 정확도>

종류 // 학습률	0.001	0.001
소프트맥스함수+크로스엔트로피	45.5%	18.9%
시그모이드함수+평균제곱오차	32.8%	21.7%

소프트맥스 함수와 크로스 엔트로피를 사용한 경우, 45.5%와 18.9%의 결과가 나타났고, 시그모이드 함수와 평균 제곱 오차(MSE)를 사용한 경우, 32.8%와 21.7%의 결과가 나타났다.

이 결과에서 소프트맥스 함수와 크로스 엔트로피 사용의 결과와 시그모이드 함수와 평균 제곱 오차 사용의 결과가 모두 약 20%~ 40% 정도로 나타났음을 알 수 있다. 또한, 학습률이 0.001인 경우에 비해 0.0001인 경우에 정확도가 하락하는 경향을 똑같이 보였다. 이를 통해 선택 분류 문제에서 소프트맥스 함수와 시그모이드 함수를 모두 사용 가능함을 유추해볼 수 있다.

하지만, 평균 제곱 오차를 사용하면 지역 최소해(local minima)에 빠질 확률이 높기 때문에 엔트로피를 사용하는 것이 나은 선택이라 할 수 있다. [1]

---

[1] 모델의 수치(미니 배치, 에포크 횟수, 학습률 등)를 약간씩 다르게 조절해 보았더니 17.2% 의 정

크로스 엔트로피 손실함수는 머신 러닝 분류 모델의 발견된 확률 분포와 예측 분포 사이의 차이를 측정한다. 예측이 가능한 모든 값이 저장되므로, 세 가지 이상의 분류 가능성이 있는 모델에서 주로 사용된다. 따라서 철판의 불량 여부 판정 문제는 이진 판단으로 시그모이드 함수와 평균 제곱 오차를 사용하고, 불량 유형을 가려내는 문제는 선택 분류로 소프트맥스 함수와 크로스 엔트로피를 사용하는 것이 적당할 것이라 판단된다.

cf) 엔트로피 : 정보 이론에서, 정보를 정량화하는 단위이다. 정보가 클수록 엔트로피가 크고, 정보가 작으면 엔트로피가 작다. 확률 분포가 균일할수록 엔트로피가 높고, 치우쳐져 있을수록 엔트로피가 낮다.

$$H(P) = H(x) = E_{X\sim P}[I(x)] = E_{X\sim P}[-log_2 P(x)]$$
$$= -\sum_i p(x) * log_2 p(x)$$
$$= \sum_i p(x) * log_2 \frac{1}{p(x)}$$

크로스 엔트로피 : 두 확률 분포 사이에 존재하는 정보량을 구하는 함수이다. 즉, 모델링을 통해 구한 분포값인 q를 모델링을 이용해 실제 확률 분포 p를 예측하게 된다. 크로스 엔트로피를 손실함수로 사용하게 되면, 0에 가깝도록 값을 줄이는 것이 목적이 되며, 정답에 해당하는 예측값이 1에 가깝게 나올때 0에 가까워진다.

$$H(P,Q) = \sum_{i=1}^{k} P_i * log_2 \frac{1}{Q_i}$$

쿨백-라이블러 발산(KLD) : 두 확률분포의 차이를 계산하는 데에 사용하는 함수이다. 어떤 이상적인 분포에 대해, 그 분포를 근사하는 다른 분포를 사용해 샘플링을 한다면 발생할 수 있는 정보 엔트로피 차이를 계산한다.[2] 아래의 식을 통해, KL Divergence를 변형하면, 크로스 엔트로피에 대한 식으로 정리됨을 알 수 있다.

$$D_{KL}(p\|q) = \sum_{i=1}^{N} p(x_i)(\log \frac{p(x_i)}{q(x_i)})$$
$$= \sum_{i=1}^{N} p(x_i) \log p(x_i) - \sum_{i=1}^{N} p(x_i) \log q(x_i)$$
$$= -H_p(X) + H_{p,q}(X)$$

확도가 나오는 등 수치가 낮게 나오는 경우가 있음을 알 수 있었다. 이는 local minima에 빠져 나타난 결과임을 유추해 볼 수 있었다.

[2] https://curt-park.github.io/2018-09-19/loss-cross-entropy/