[HW#3 | NLP preprocessing & MLP 구현 실습과제]

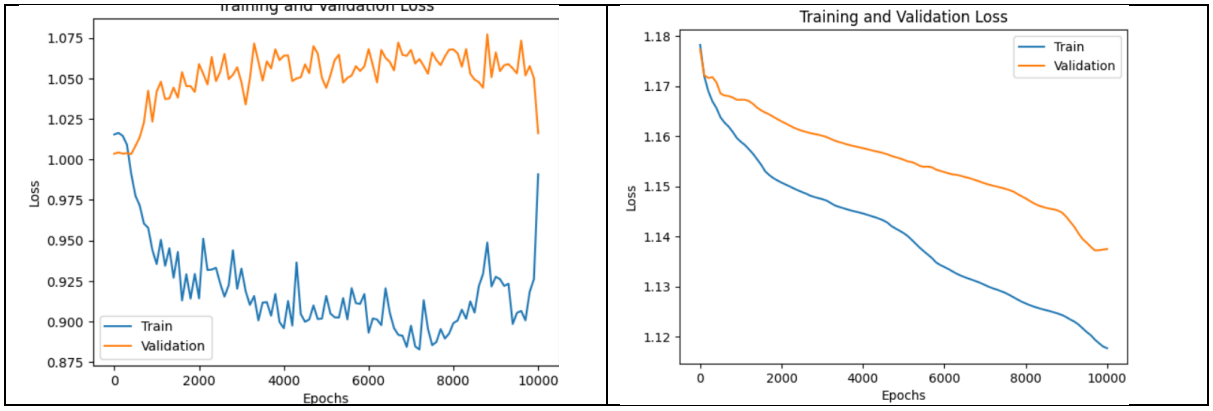
컴퓨터공학부
201901208
김현동

모델 설명

5개의 선형 레이어와 활성화 함수(4개 : ReLU 1개: Softmax)로 이루어져 있습니다. 첫 번째 layer의 입력 크기 42와 마지막 layer 출력 크기 3 데이터의 특징에 있습니다. Train data의 shape(4640,42)의 feature 수 42와 다중 분류 3가지(positive, negative, neutral) 3개입니다.

은닉층의 크기는 (10,10)으로 구성을 했습니다. 실험적으로 정한 것인데요. 한 번에 feature를 줄이는 것보다 점차 줄이는게 나을 수 있을 거라 생각했습니다. 그에 따라 은닉층(30,20)(20,10)(10,3)을 실행하고 learning rate를 확인했을때, 매우 불안정한 모습을 볼 수 있었습니다. 따라서 안정적인 형태의 (10,10) layer를 여러개 사용하게 되었습니다.

- 실험을 위해 동일한 조건에서 수행한 학습입니다. (layer 입력, 출력 크기를 제외한)



다중 분류를 위해서 마지막 활성화 함수 Softmax와 loss function CrossEntropy를 사용했습니다.

Softmax의 각 클래스 예측을 확률 값으로 한다는 특징. Loss function은 예측값과 실제 y_label의 차이를 계산할 때 사용합니다. Loss를 줄이는 방향으로 학습이 진행됩니다. Y_label이 one-hot encoding 되어 있는 categorical 한 값에 적합하다는 점과 log를 사용해서 잘못된 예측에 대한 손실을 크게 한다는 특징이 있습니다. 이러한 결론으로 softmax와 CrossEntropy Loss가 서로 잘 맞는다고 생각했습니다.

Hyper parameter 설명입니다.

Loss function

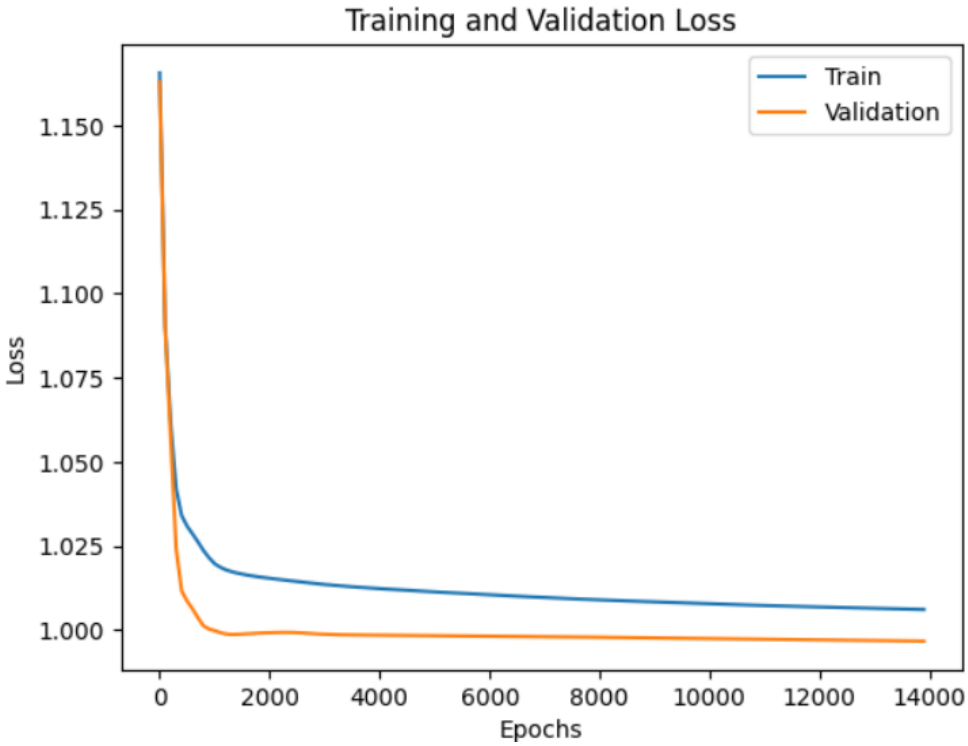
1. Weight=None : 클래스간의 불균형이 존재할때 사용합니다. (1:3115 2:1838 0:847) 확인을 통해 불균형이 존재한다고 판단하여 아래 코드를 통해 가중치를 계산하고 적용해 보았습니다. 가중치를 적용한 것과 적용하지 않은 모델이 약 0.05의 정확도 차이를 보여줬고, None 설정이 더 높았습니다.

```
class_counts = train['Sentiment'].value_counts().tolist()
class_weights = torch.tensor([max(class_counts) / count for count in
class_counts], dtype=torch.float32)
```

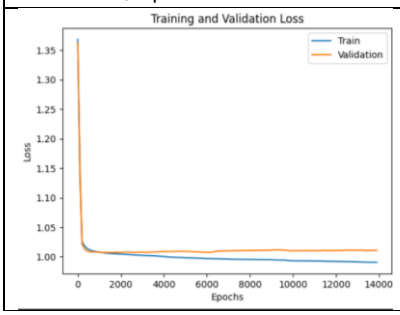
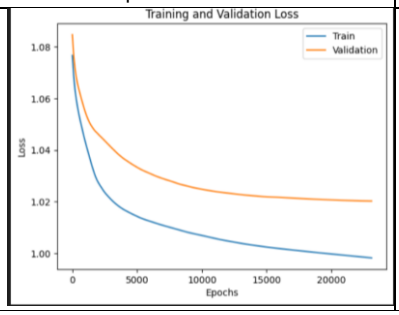
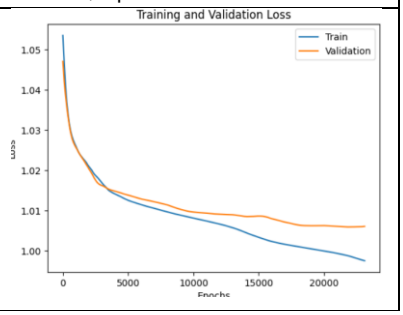
Optimizer

1. Lr=0.0001은 실험적으로 구했습니다. 결정 기준은 다음과 같습니다. Learning curve를 확인했을때 fit 한가? Overfiting or underfiting의 위험이 있는가? 정확도는 준수한가? 위아래로 튀지 않고 안정적으로 학습 하는가?
   그리고 해당 모델의 learning curve 입니다.
2. Epoch = 3 또한 1번과 같은 기준으로 결정했습니다.

validation에 대한 정확도 0.701



비교 샘플

Lr = 0.01 , epoch=3	Lr=0.001 epoch = 5	Lr=0.1 , epoch=5
		

Open AI는 사용해봤지만 모델을 직접 작성하여 사용하는 것은 처음입니다. 잘 한건지, 성능이 어느정도 나온건지 모르겠지만 정확도를 기준으로 판단하는 것이 최선이었습니다. 모델을 구성하는 각 요소에 대한 목적을 복기하고 각 요소들은 실험적으로 구하는 것을 느낄 수 있는 시간이었습니다.