딥러닝 및 응용

image classification 개선

2015004475 김태훈

1. 구동 환경

OS: Windows 10 64비트

Language: Python 3.7

필요 패키지와 버전 정보는 requirements.txt에 기재됨.

1. 코드 설명

4가지 버전 중 v3 채택.

training\_epochs: 몇 에폭만큼 학습할지 결정. 20, 25, 30으로 테스트 해본 결과 25에서의 f1스코어가 가장 높았음.

batch\_size: 배치 사이즈 지정, 128.

x, y, rate: float32 데이터타입의 플레이스 홀더.

x, y\_train: 주어진 트레이닝 데이터 로드.

dev\_num: development set로 이용할 데이터 개수

x,, y\_dev, train: 각각 x, y 트레이닝셋, 데브셋만큼 원래 불러온 데이터 슬라이스.

y\_train, dev\_one\_hot: y 데이터를 원핫인코딩.

y\_pred, logits: build\_CNN\_classifier 함수의 리턴으로 각각 모델의 hypothesis와 logits를 받게 될 변수.

cost: 모델의 코스트.

train\_step: Adam 옵티마이저를 이용해 cost를 최소화 시키도록 지정.

Adam 옵티마이저의 하이퍼 파라미터:

learning\_rate = 0.001

beta1 = 0.0

beta2 = 0.999

epsilon = 10^-8

total\_batch: 미리 지정된 배치 사이즈로 트레이닝셋 개수를 나눠 모두 커버하도록 계산.

get\_batch\_data: 인덱스와 배치사이즈를 받아 x, y 데이터에서 데이터를 뽑아 반환.

build\_CNN\_classifier: 뉴럴넷 모델 구성. 3개의 CNN 레이어, 1개의 fc(fully connected) 레이어.

l1: 5 \* 5 사이즈 3채널 필터 32출력, 드롭아웃 적용, 3 \* 3 풀링 필터로 stride 2칸씩 적용. 전체 크기 유지를 위해 패딩 = SAME.

l2: 5 \* 5 32채널 64 출력, 이외 l1과 동일.

l3: 5 \* 5 64채널, 128 출력. 마찬가지로 나머지는 l1, l2와 동일.

l3\_plat: l3\_pool을 일자형 벡터로 펼쳤을 때. 초기 데이터가 32 \* 32 사이즈였고 풀링 stride를 2칸씩 적용했기 때문에 각 레이어를 지날 때 마다 출력 벡터의 shape가 16\*16\*32, 8\*8\*64, 마지막으로 l3을 지나면 4\*4\*128. 따라서 l3\_plat의 shape = 4 \* 4 \* 128.

fc1: Fully connected layer 1. CNN을 지난 feature를 classification에 이용하기 위해 10개의 뉴런으로 이어지는 FC 구조로 구성. softmax를 activate func로 이용한다.

세션 가동 후

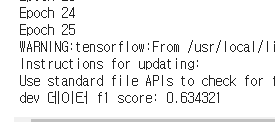
1. 변수 초기화
2. 정해진 epoch 수(25 epoch) 만큼 반복
   1. 데이터를 가져와
   2. batch 수 만큼
      1. batch 데이터를 가져와서
      2. dropout rate 0.25로 학습 진행
3. 학습된 모델 체크포인트 저장 후 불러와서
4. 떼어둔 dev셋으로 f1스코어를 측정
5. 결과 파일 저장.
6. 실험 결과

오리지널 코드



개선 버전 (버전3)

CNN 3중, epoch 25, rate 0.25 (0.2, 0.25, 0.3에서 테스트)



그 외 버전들

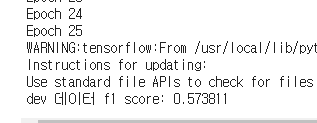
버전2

CNN 2층, epoch 10, rate 0.25



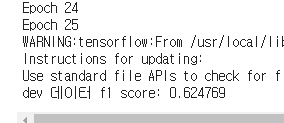
버전4

CNN 3중, FC 2중 레이어. rate 0.25

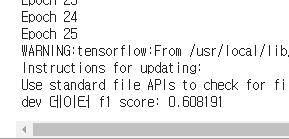


버전5

CNN 3중, FC 다시 1중, 풀링 3\*3에서 2\*2로 변경, dropout rate 0.25



dropout rate 0.3



Adam 옵티마이저 파라미터는 모두 같은 값을 사용했다.

1. 결론

CNN 레이어의 깊이를 늘리는 것이 f1스코어 개선 dropout rate 0.25에서 가장 좋은 성능을 보임을 확인했다.

에 효과적이었다.

Dropout을 적용하고 epoch 수를 늘리는 것 또한 효과적임을 확인했다.