

딥러닝 및 응용

dropout과 adam 적용

2015004475 김태훈

1. 구동 환경

OS: Windows 10 64비트

Language: Python 3.7

환경설정은 conda를 이용하였으며 필요 패키지는 requirements.txt에 기재됨.

2. 코드 설명

mnist 데이터 one-hot 인코딩 로드 (N개)

X: mnist 이미지가 784픽셀이므로 [None, 784] 꼴의 인풋벡터를 placeholder로 구성. 추후 [N, 784] 형태가 될 것.

Y: 각 샘플 당 0~9까지 라벨이 있으므로 길이 10짜리 라벨. 따라서 [N, 10]

rate: 최신 tensorflow는 $1 - \text{keep_prob}$ 의 값을 rate 라는 파라미터로 받아 사용.

레이어 1, 2

- sigmoid 함수를 activation 함수로 사용.
- rate 만큼 뉴런을 disable.

레이어 3

- 출력 레이어. 10개의 뉴런으로 각각 라벨을 담당.
- softmax 함수를 activation으로 사용.

cost: loss함수로 크로스 엔트로피를 사용.

opt: Adam 옵티마이저 사용. 각 파라미터는 흔히 사용하는 수치. cost를 minimize 하는 방향으로 작동함.

batch_size: 미니배치 사이즈. 100으로 설정 됨.

세션을 가동한 후...

1. 텐서 variable을 랜덤 초기화.
2. 15회의 epoch을 반복하며
 - A. 해당 epoch의 평균 코스트를 초기화.
 - B. 샘플 총량을 batch 사이즈로 나눠 미니배치 개수 계산.
 - C. 미니배치 개수만큼 반복하며
 - i. 배치 사이즈만큼의 데이터를 뽑아냄
 - ii. 미니배치만큼 뽑은 인풋데이터 batch_xs, 정답 데이터 batch_ys 그리고 dropout을 위한 rate를 지정하여 feed_dict로, fetch에 코스트와 옵티마이저를 파라미터로 세션을 작동.
 - iii. 평균 코스트 계산.
 - D. epoch 번호와 평균 코스트를 출력
3. is_correct에 hypothesis(소프트맥스 적용 후 출력값 = y_hat)와 Y에 있는 정답이 서로 같은지 확인하여 T/F 할당.
4. is_correct를 이용해 정확도 계산식 정의.
5. 테스트 데이터를 모델에 적용하여 정확도 출력

3. 실험 결과

A. dropout과 adam 옵티마이저를 적용한 결과.

```
dropout_and_adam x
2021-04-10 17:14:57.805939: I tensorflow
Epoch: 1, cost = 2.410091444
Epoch: 2, cost = 0.894938071
Epoch: 3, cost = 0.621789177
Epoch: 4, cost = 0.483720618
Epoch: 5, cost = 0.403777923
Epoch: 6, cost = 0.344851438
Epoch: 7, cost = 0.292958858
Epoch: 8, cost = 0.259689502
Epoch: 9, cost = 0.231359763
Epoch: 10, cost = 0.208487631
Epoch: 11, cost = 0.194343513
Epoch: 12, cost = 0.175761574
Epoch: 13, cost = 0.158976399
Epoch: 14, cost = 0.150167090
Epoch: 15, cost = 0.138609221
Accuracy: 0.9691

Process finished with exit code 0
```

B. adam 옵티마이저만 적용한 결과.

```
2021-04-10 20:58:23.476001: I tensorflow
Epoch: 1, cost = 0.684463256
Epoch: 2, cost = 0.242721558
Epoch: 3, cost = 0.168937829
Epoch: 4, cost = 0.125917790
Epoch: 5, cost = 0.095231864
Epoch: 6, cost = 0.072908049
Epoch: 7, cost = 0.056563722
Epoch: 8, cost = 0.043210895
Epoch: 9, cost = 0.032159434
Epoch: 10, cost = 0.024817717
Epoch: 11, cost = 0.018738138
Epoch: 12, cost = 0.013881819
Epoch: 13, cost = 0.010889992
Epoch: 14, cost = 0.007805049
Epoch: 15, cost = 0.005861137
Accuracy: 0.9689

Process finished with exit code 0
|
```

C. dropout과 gradient descent 옵티마이저를 사용한 결과.

```
2021-04-10 21:03:48.805238: I tensorflow/
Epoch: 1, cost = 2.566466910
Epoch: 2, cost = 1.071221633
Epoch: 3, cost = 0.793128642
Epoch: 4, cost = 0.676251264
Epoch: 5, cost = 0.605656575
Epoch: 6, cost = 0.560334897
Epoch: 7, cost = 0.529205528
Epoch: 8, cost = 0.497438130
Epoch: 9, cost = 0.477231758
Epoch: 10, cost = 0.453020129
Epoch: 11, cost = 0.438989304
Epoch: 12, cost = 0.423054952
Epoch: 13, cost = 0.413041485
Epoch: 14, cost = 0.404667236
Epoch: 15, cost = 0.390790855
Accuracy: 0.93

Process finished with exit code 0
```

3종류의 코드를 실험한 결과 dropout과 adam 옵티마이저를 사용했을 때 정확도가 가장 높게 나타났다.