

프로젝트2: MNIST NN모델의 prediction 실험

[1] 과제 개요

과제1 에서 구축한 NN 모델에 대하여 prediction 즉 예측 작업을 수행하여 결과가 가장 나쁜 예제들을 사진으로 출력하여 보는 실험을 수행한다.

[2] 과제 절차

(a) **모델구축:** 모델은 과제1의 두번째 모델 즉 model2 를 사용한다 (즉, 층의 총개수는 3 개로 한다. 층들의 크기: $m_1 = 256$, $m_2=128$, $m_3=10$). 모델의 이름은 "model" 로 한다고 하자. 모델 model 의 구축은 mnist_train.txt 내의 60,000 개를 이용한 훈련작업을 통하여 달성한다 (첫 80%는 훈련용, 나머지 20%는 검증용). batch size 및 epoch 는 과제1에서 알아 낸 값을 그대로 이용한다.

(b) **Prediction 작업:** model 의 훈련(검증)이 끝나면 모델이 구축된 것이다. 그 다음에는 예측(prediction) 작업을 수행한다. 대상은 mnist_test.txt 의 처음부분 1,600 개만을 이용한다. prediction 작업은 tensorflow의 model.predict 메소드를 호출하여 수행할 수 있다. 이용 예:

```
pred = model.predict(test_x, batch_size=16, verbose=2)
(여기서 test_x: 1,600개 테스트 예제들의 input 를 넣은 numpy 배열)
```

함수 predict 는 예측작업의 결과를 pred 에 반환한다. pred 는 $1,600 \times 10$ 의 2-dimension 데이터를 받는다. 각 행(row)은 예제 입력이 10 개의 분류범주(classification category) 즉 숫자에 속할 확률들을 가진다.

(c) **오류예제들 찾는 단계:** model 이 정답을 내주는 데 실패한 예제들 중에 최고로 나쁜 예측을 한 숫자 이미지 3 개를 출력해 보자. 일단 분류에 실패한 예제들이 누구인지를 알아낸다.

이것은 test_y 에 준비된 실제 정답 레이블(target label)을 이용한다. 1,600 개의 테스트 예제 중에 번호가 i 인 테스트 예제를 예로 들어 보자($i = 0, \dots, 1599$ 중 한 번호). 이 예제에 대한 모델의 출력은 $\text{pred}[i]$ 이다. $\text{pred}[i]$ 는 10개의 확률 $\text{pred}[i][j]$, $j=0, \dots, 9$, 을 가진다. 이 10 개의 값 중 최대가 되는 j 가 무엇인지를 구한다. 모델은 입력 $\text{test_x}[i]$ 에 주어진 숫자의 모양이 숫자 j 라고 예측한 것이다. 이 모델이 예측한 레이블 j 를 실제 정답 레이블($\text{test_y}[i]$)와 비교한다. 이 둘이 같으면 모델의 예측이 올바른 것이고, 아니면 예측을 잘못된 즉 틀린 것이다.

1,600 개의 예제에 대하여 이렇게 알아 보아서 예측이 틀린 예제들의 예제 번호를 리스트 wrong에 모은다. $\text{wrong} = [\dots]$ 은 정수 리스트로서 각 원소는 모델의 예측이 틀린 예제의 번호이다. wrong 에 모은 예제의 총 수를 wnum 이라 하자. 다음 wrong 에 모인 각 예제마다 실제정답 클래스에 대하여 모델이 내놓은 확률을 구한다. 이를 정답클래스예측확률, tg_prob,이라 부르자. 예를 들면 $\text{wrong}[k] = r$ 이라 하자. 즉 1,600 개의 테스트 예제중 위치 r 의 예제에 대해 모델의 예측이 틀렸다. 이 예제에 대하여 정답이 되는 클래스의 번호(class label)은 $\text{test_y}[r]$ 에 있다. $a = \text{test_y}[r]$ 이라 하자. 그러면 $\text{pred}[r][a]$ 는 모델이 예제 r 의 정답 클래스에 대하여 출력한 확률 즉 tg_prb 이다. 즉 model 은 예제 r 에 대하여 입력으로 들어온 숫자 이미지가 실제로 정답이 되는 범주 a 에 속할 확률을 $\text{pred}[r][a]$ 로 판단하였다. 그런데 이 판단은 옳바르지 않은 판단인 것으로 판명되었다.

wrong 에 모은 예제들에 대하여 정답클래스예측확률 즉 tg_prb 을 리스트 tg 에 모으자. $\text{tg} = [\dots]$. wrong과 tg 의 각 위치의 값을 한 쌍 (ex, pr) 로 하여(즉 두개의 원소를 가진 리스트로 하여) 리스트 wr_tg 에 넣자. 그 다음 numpy.asarray 를 이용하여 이 wr_tg 를 numpy array 로 변경한다. 그러면 다음 그림과 같은 numpy 배열이 된다. 차원1 의 위치 0 은 예제번호, 위치 1 은 이 예제의 정답클래스예측확률을 가진다.

배열 wr_tg:

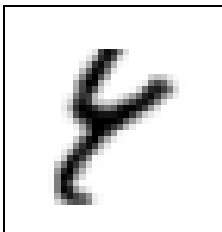
	0	1
0	48	0.23
1	98	0.14
2	354	0.31

wnum-1	1523	0.12

이 배열 wr_tg 의 각 행은 모델이 정답을 못맞춘 예제의 예제번호 와 tg_prob 을 가진다. 이 배열을 tg_prob 를 기준으로 하여 오름차순으로 정렬한다. 이 정렬작업을 하는 파이썬 코드를 직접 짜 넣어도 되고 아니면 numpy.sort 를 사용해도 된다. 그런데 후자의 경우 매우 조심하여야 한다. 이유는 각 행을 하나의 구조체 즉 structure 로 취급해야 한다는 점이다.

(d) 출력단계: 정렬을 마친 결과를 배열 res 에 넣는다고 하자. 이 배열의 구조는 wr_tg 와 같다. res 의 맨 처음부터 아래로 가면서 각 행은 가장 예측이 실패한 즉 예측을 잘못된 예제들부터 덜 나쁜 순서대로 나온다. 첫 행부터 아래로 3 개의 예제들에 대하여 순서대로 각각 다음과 같은 정보를 출력한다:

예제번호: ??
정답레이블: ??
예측레이블: ??
이 예제의 tg_prob: ??
이 예제의 이미지:



(3) 제출할 파일:

개발한 프로그램 파일: ipynb 타입

보고서: 위의 실험과정에서 다음 사항이 나오는 실행 결과창의 해당부분을 캡처하여 보고서에 넣는다.

- 훈련(검증)의 마지막 epoch 의 출력,
- pred 의 shape (프로그램에 이 정보를 출력하는 print 출력문을 넣을 것.)
- wnum 의 값 (프로그램에 값을 출력하는 문장을 넣을 것.)
- (2-d) 의 출력 (3개의 예제에 대하여 내용).

주: 파일명에 자신의 이름을 넣을 것.