

13장. 오토인코더 과제

부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공
201724579 정현모

1. 개요

오토인코더는 레이블 정보 없이 입력을 재현하는 방식으로 스스로 학습하는 비지도학습 방식의 신경망이다. 입력 재현 과정을 통해 데이터의 분포 특성 및 핵심 개념이나 패턴을 스스로 파악하며, 인코더와 디코더 두 부분으로 구성되어 있다. 인코더는 입력 데이터를 정보가 압축된 중간 표현에 해당하는 코드로 변환하는 것이고, 디코더는 코더에 담긴 중간 표현으로부터 입력과 비슷한 출력을 생성하는 것이다.

이번 과제에선 지도 학습과 비지도 학습의 차이를 알아보고, 오토인코더에 대해 자세하게 알아본다. 또한 오토인코더가 어디에 사용되는지, 왜 사용되는지에 대해서 알아보겠다.

2. 학습 결과

2.1. MNIST 손글씨 오토인코더 신경망 학습 비교

2.1.1. Epoch 에 따른 순환 신경망 모델 비교

Model mnist_mlp_all

| 구분 | 내용 | 값 |
|----------|---|----------|
| 조건 | Epoch | 10 |
| | hidden layer | MLP [10] |
| | 지도학습에 쓰인 데이터 사용량 | 100% |
| 최종 결과 | Accuaracy | 90.5% |
| | Train time | 30 secs |
| 결과 화면 | Model mnist_mlp_all train started: Epoch 2: cost=0.514, accuracy=0.864/0.880 (5/5 secs) Epoch 4: cost=0.450, accuracy=0.884/0.930 (6/11 secs) Epoch 6: cost=0.424, accuracy=0.896/0.820 (6/17 secs) Epoch 8: cost=0.405, accuracy=0.901/0.840 (7/24 secs) Epoch 10: cost=0.399, accuracy=0.902/0.960 (6/30 secs) Model mnist_mlp_all train ended in 30 secs: Model mnist_mlp_all test report: accuracy = 0.905, (0 secs) | |

13장. 오토인코더 과제

Model mnist_mlp_1p

| 구분 | 내용 | 값 |
|----------|---|-----------------|
| 조건 | Epoch | 10 |
| | hidden layer | MLP [10] |
| | 지도학습에 쓰인 데이터 사용량 | 1% |
| 최종 결과 | Accuarcy | 68.4% |
| | Train time | 0 secs |
| 결과 화면 | Model mnist_mlp_1p train started: Epoch 2: cost=1.402, accuracy=0.533/0.540 (0/0 secs) Epoch 4: cost=0.870, accuracy=0.681/0.550 (0/0 secs) Epoch 6: cost=0.579, accuracy=0.781/0.670 (0/0 secs) Epoch 8: cost=0.419, accuracy=0.817/0.790 (0/0 secs) Epoch 10: cost=0.337, accuracy=0.871/0.750 (0/0 secs) Model mnist_mlp_1p train ended in 0 secs: Model mnist_mlp_1p test report: accuracy = 0.684, (0 secs) | |

Model mnist_auto_1

| 구분 | 내용 | 값 |
|----------|---|-------------------------|
| 조건 | Epoch | 10 |
| | hidden layer | MLP [10] + 확장인코더 |
| | 지도학습에 쓰인 데이터 사용량 | 1% |
| 최종 결과 | Accuarcy | 74.6% |
| | Train time | 1 secs |
| 결과 화면 | Model mnist_auto_1 train started: Epoch 2: cost=1.375, accuracy=0.552/0.540 (0/0 secs) Epoch 4: cost=0.955, accuracy=0.677/0.590 (1/1 secs) Epoch 6: cost=0.821, accuracy=0.706/0.640 (0/1 secs) Epoch 8: cost=0.718, accuracy=0.758/0.660 (0/1 secs) Epoch 10: cost=0.638, accuracy=0.787/0.760 (0/1 secs) Model mnist_auto_1 train ended in 1 secs: Model mnist_auto_1 test report: accuracy = 0.746, (0 secs) | |

13장. 오토인코더 과제

Model mnist_auto_2

| 구분 | 내용 | 값 |
|----------|--|--------------------------|
| 조건 | Epoch | 10 |
| | hidden layer | MLP [64, 10, 64] + 확장인코더 |
| | 지도학습에 쓰인 데이터 사용량 | 1% |
| 최종 결과 | Accuarcy | 79.6% |
| | Train time | 306 + 1 secs |
| 결과 화면 | Model mnist_auto_2 autoencode started: Epoch 5: cost=2692.180, accuracy=-0.564/-0.564 (135/135 secs) Epoch 10: cost=2646.832, accuracy=-0.551/-0.551 (171/306 secs) Model mnist_auto_2 autoencode ended in 306 secs: Model mnist_auto_2 train started: Epoch 5: cost=0.764, accuracy=0.735/0.660 (1/1 secs) Epoch 10: cost=0.611, accuracy=0.798/0.880 (0/1 secs) Model mnist_auto_2 train ended in 1 secs: Model mnist_auto_2 test report: accuracy = 0.796, (0 secs) | |

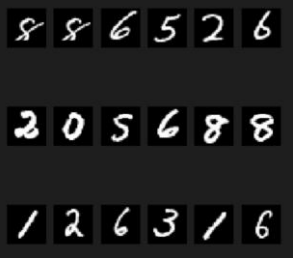
Model mnist_auto_3

| 구분 | 내용 | 값 |
|----------|--|-------------------------|
| 조건 | Epoch | 10 |
| | hidden layer | MLP [64, 1, 64] + 확장인코더 |
| | 지도학습에 쓰인 데이터 사용량 | 1% |
| 최종 결과 | Accuarcy | 18.7% |
| | Train time | 3 secs |
| 결과 화면 | Model mnist_auto_3 autoencode started: Epoch 5: cost=5323.163, accuracy=-1.201/-1.201 (115/115 secs) Epoch 10: cost=5090.021, accuracy=-1.152/-1.152 (179/294 secs) Model mnist_auto_3 autoencode ended in 294 secs: Model mnist_auto_3 train started: Epoch 5: cost=2.223, accuracy=0.154/0.180 (2/2 secs) Epoch 10: cost=2.192, accuracy=0.177/0.130 (1/3 secs) Model mnist_auto_3 train ended in 3 secs: Model mnist_auto_3 test report: accuracy = 0.187, (0 secs) | |

13장. 오토인코더 과제

2.2.1. 시맨틱 해싱 모델 비교

Model mnist_hash_1

| 구분 | 내용 | 값 |
|-------|---|--------|
| 조건 | Epoch | 10 |
| | hidden layer | 시맨틱 해시 |
| 결과 화면 |  | |

Model mnist_hash_2

| 구분 | 내용 | 값 |
|-------|--|-----------------------------|
| 조건 | Epoch | 10 |
| | hidden layer | 시맨틱 해시 + 10 hidden layer 추가 |
| 결과 화면 |  | |

Model mnist_hash_3

| 구분 | 내용 | 값 |
|-------|--|-----------------------------|
| 조건 | Epoch | 40 |
| | hidden layer | 시맨틱 해시 + 10 hidden layer 추가 |
| 결과 화면 |  | |

13장. 오토인코더 과제

3. 결론

3.1. 지도학습과 비지도학습의 차이

지도학습은 기본적으로 학습 데이터에 부착된 레이블 정보, 즉 정답 정보를 활용하여 학습하는 방식이다. 활용할 구체적인 용도에 맞춰 학습시키는 신경망 개발의 필수적인 과정으로 레이블링 작업에는 보통 학습용 데이터 수집보다 더 많은 인력과 비용이 필요하기 때문에 비지도 학습이 등장했다.

비지도 학습은 이름에서 유추할 수 있듯이 레이블 정보 없이 원래의 수집된 데이터만으로 스스로 학습하는 방식이다. 입력과 비슷한 출력 재현을 목표로 삼는 오토인코더가 대표적이고, 데이터만 수집하면 레이블링 작업 없이 바로 학습이 가능하기 때문에 저렴하고 신속한 처리가 가능하다.

비지도학습 만으로 어떤 유용한 결과도 얻을 수 없다. 굳이 입력과 출력이 같은 오토 인코더를 사용할 바에 단순 복사를 하면 된다. 하지만 오토인코더를 사용하는 이유는, 잘 훈련 된 오토 인코더로 특정한 하나의 용도가 아닌 다른 다양한 용도에 활용될 뿐만 아니라, 인코더가 압축 생성하는 내부 코드에는 입력의 여러 유용한 패턴이 압축되어 표현될 가능성이 높기 때문에 활용가치가 충분하다.

이러한 비지도학습과 지도학습의 단점을 상호보완하기 위해, 비지도학습과 지도학습을 결합한 준지도학습이란 것이 등장했는데. 이는 전체 데이터를 이용한 오토인코더 학습 후 약간의 레이블링된 데이터를 이용해 지도학습을 수행하는 방식이다. 인코더가 만들어내는 내부 코드 속에 압축된 유용한 패턴들을 활용할 수 있으며, 동시에 적은 양의 레이블링된 데이터만으로도 높은 품질의 학습 결과를 기대할 수 있다. 많은 양의 데이터가 수집되었으나 일부 데이터에만 레이블링 작업이 수행된 경우에 특히 유용하다.

3.2. 오토인코더

각각 별도의 신경망인 인코더와 디코더 두 부분으로 구성된 신경망이다. 입력과 출력이 같다는 특징이 있으며 학습 시에도 입력과 출력사이의 오차를 손실 함수로 이용한다. 아래 [그림1]에서 가운데 회색부분을 '코드'라고 부르는데, 코드는 인코더가 입력을 처리하여 만들어진 내부 중간 표현으로, 디코더는 이 정보를 처리하여 출력을 생성한다. 보통 신경망 설계자가 형태를 마음대로 정할 수 있지만, 입력에 비해 작은 크기로 정하는 것이 보통이다.

13장. 오토인코더 과제

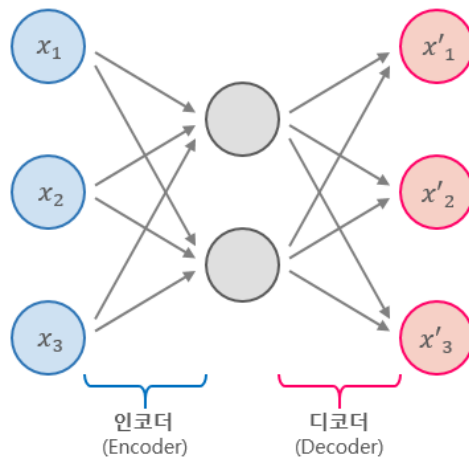


그림 1. 오토 인코더

오토 인코더가 쓰이는 한 예시로 잡음 제거용 오토 인코더가 있는데, 단어 그대로 입력에서 잡음을 제거해주는 신경망이다. 오토 인코더의 입력과 인코더 사이에 잡음을 추가하여 디코더 출력에선 잡음 주입 이전의 원본 입력과 비교하여 손실함수를 계산한다. 이렇게 되면 변형된 입력에서 잡음을 제거해 원본 데이터로 회복시키는 기능을 갖는 신경망이 만들어진다.

이런 특성을 가진 오토 인코더가 자주 쓰이는 이유엔 여러가지가 있는데, 우선 데이터 압축 측면에서 AutoEncoder를 이용해 이미지나 음성 파일에서 중요 Feature만 가지고 압축할 경우, 용량도 더 작고 품질도 더 좋다는 장점이 있다. 또한 비지도 학습을 하는 오토 인코더의 특성상, 자동으로 중요한 Feature를 찾아준다.

3.3. 실험 결과

실험은 다층 퍼셉트론 계층을 사용했다는 점에선 모든 실험군이 동일하지만, 첫번째, 두번째 모델의 경우 지도 학습에 전체 데이터를 사용하느냐, 1%만 사용하느냐에 따른 차이가 있다. 전체 데이터를 지도 학습에 사용한 경우 약 90%의 굉장히 높은 정확도를 보여준 반면, 1%만 사용했을 경우 68.4%의 정확도를 보여주었다. 하지만, 전체 데이터를 지도 학습한다는 것은, 모든 데이터에 라벨링이 되어있어야 하고, 학습 시간 역시 30 secs와 0 sec로 극명한 차이가 나는 것을 확인할 수 있다.

두번째 실험은 오토 인코더에서 '코드'에 해당하는 부분의 크기와 인코더, 디코더의 크기를 조절해가며 결과를 비교해보았다. 인코더, 디코더의 크기가 크면 클수록, '코드'의 크기가 크면 클수록 당연히 정확도가 올라가는 것을 확인할 수 있었지만, '코드'나 인코더, 디코더의 크기가 조금만 바뀌어도 학습 속도 면에서 시간차이가 월등하게 나는 것을 확인할 수 있었다. 학습에 많은 비용이 드는 만큼 하나의 잘 학습된 오토 인코더로 여러 분야에서 사용하는 것이 현명하다.