|  |  |  |  |
| --- | --- | --- | --- |
| **HW#01** | **딥러닝 과제보고서** | **학번: 2018741035** | **이름: 한준호** |
| #1. 과제 목표 | | | |
| 딥러닝의 일반화 성능을 높이기 위해 Regularization을 해봄으로써 Overfitting을 줄이고 Test accuracy을 높힌다. 그 과정에서 Dropout, Batch Normalization, Data Augmentation와 같은 대표적인 Regularization 기법을 학습한다. | | | |
| #2. 코드 수행 결과 (사진 첨부) | | | |
| 1. Data Generation  받아온 mnist dataset의 60000개의 Training Set 중 Overfitting을 유도하기 위해 60000개 Data중 일부의 Data만 이용하려고 한다.  텍스트이(가) 표시된 사진  자동 생성된 설명  일부 Data만 이용하기 위해 Python의 ‘슬라이스(Slice)’ 기법 사용하였다.  0~59999의 범위로 이루어진 training data에서 0~199의 범위로 200개의 Data로 이루어진 x\_validation와 y\_validation를 만들어주었고, 다른 범위인 200~2199의 범위로 2000개의 Data로 이루어진 large\_x\_train, large\_y\_train를 만들어주었다. 마지막으로 다른 범위인 2200~3199의 범위로 1000개의 Data로 이루어진 x\_train, y\_train을 만들어주었다.  텍스트이(가) 표시된 사진  자동 생성된 설명  <결과>  일부 Data만 이용할 수 있도록 원하는 개수로 Training Data를 설정해주었다.  2.모델 생성  • Flatten    Flatten Layer는 추출된 주요 특징을 전결합층에 전달하기 위해 1차원 자료로 바꿔주는 Layer이다. 이미지 형태의 데이터를 배열 형태로 Flatten하게 만들어준다.  • Dense  도표이(가) 표시된 사진  자동 생성된 설명  Keras에서 사용되는 레이어(Layer, 층)는 신경망 모델을 구성하는 주요한 요소이다. 이 중 Dense Layer는 다층 퍼셉트론 신경망에서 사용되는 레이어로 뉴런의 입력과 출력을 모두 연결해준다.  테이블이(가) 표시된 사진  자동 생성된 설명  이와 같은 MLP 분류 모델을 만들기 위해 다음과 같이 Coding 하였다.  텍스트이(가) 표시된 사진  자동 생성된 설명  모델에 처음 입력되는 x Data가 이미지로 28\*28 형태이므로 Flatten( ) 함수의 인자인input\_shape = (28, 28)로 설정해주었다.  위와 같은 MLP 분류 모델을 만들기 위해 Dense( ) 함수의 첫 번째 인자인 출력 뉴런의 수를 위와 같이 설정해주었다.  모델의 마지막 레이어에는 10개의 출력을 가지기 위해 Dense( ) 함수의 첫 번째 인자인 출력 뉴런의 수를 10으로 설정해주었고, 활성화 함수로 다중 클래스 분류 문제에서 출력층에 주로 사용하는 ‘softmax’ 함수를 사용하였다.  테이블이(가) 표시된 사진  자동 생성된 설명  <결과>  3. Learning MLP    <결과>  기본 MLP 분류 모델을 학습해 보았다. 기본 MLP 분류 모델은 Training data에 맞추어 과도하게 학습이 이루어져 Test data에서 높은 성능이 나지 않는 현상, 즉 Generalization 성능이 높지 않게 나타나는 현상인 Overfitting이 발생하였다. Overfitting이 발생하면 Training accuracy는 높지만 Test accuracy는 높지 않게 나타난다.  <결과> 사진을 보면 Training accuracy와 Test accuracy가 크게 차이가 나지는 않았지만, Training accuracy보다 Test accuracy가 높지 않게 나타났다.  4. Regularization  Overfitting이 발생한 모델에 다양한 Regularization 기법을 적용해보았다.  4.1 Large Dataset  Training data가 충분하다면 Overfitting 현상이 발생할 가능성이 줄어드므로 기본 MLP 분류 모델에 사용한 1000개의 Data로 이루어진 x\_train, y\_train 대신 2000개의 Data로 이루어진large\_x\_train, large\_y\_train을 사용하였다.    <결과>  Large Training Data를 사용한 Large Model의 Generalization 성능이 증가한 것을 확인하였다.  4.2 Dropout  •Drop-out  차트이(가) 표시된 사진  자동 생성된 설명  Drop-out은 서로 연결된 연결망(layer)에서 0부터 1사이의 확률로 뉴런을 제거(drop)하는 기법이다.  테이블이(가) 표시된 사진  자동 생성된 설명  이와 같은 MLP 분류 모델을 만들기 위해 다음과 같이 Coding 하였다.  텍스트이(가) 표시된 사진  자동 생성된 설명  기본 MLP 분류 모델에 Dropout Layer만 추가해주었다. Dropout( ) 함수의 첫 번째 인자인 각각의 뉴런이 드랍 될 확률인 dropout\_rate = 0.2로 설정해주었다.  테이블이(가) 표시된 사진  자동 생성된 설명  <결과>  Dropout Model을 학습해보았다.    <결과>  Large Model보다 Dropout Model의 Generalization 성능이 증가한 것을 확인하였다.  4.3 Batch Normalization  •Normalization (정규화)  도표이(가) 표시된 사진  자동 생성된 설명  각 레이어의 입력의 분산을 평균 0, 표준편차 1인 입력 값으로 정규화 시키는 방법이다.  기본적으로 정규화를 하는 이유는 학습을 더 빨리 하기 위해서 또는 Local Optimum 문제에 빠지는 가능성을 줄이기 위해서 사용한다.  •Batch Normalization  배치 정규화는 평균과 분산을 조정하는 과정이 별도의 과정으로 떼어진 것이 아니라, 신경망 안에 포함되어 학습 시 평균과 분산을 조정하는 과정 역시 같이 조절된다. 즉, 각 레이어마다 정규화 하는 레이어를 두어, 변형된 분포가 나오지 않도록 조절하게 하는 것이 배치 정규화이다.  테이블이(가) 표시된 사진  자동 생성된 설명  이와 같은 MLP 분류 모델을 만들기 위해 다음과 같이 Coding 하였다.  텍스트이(가) 표시된 사진  자동 생성된 설명  Layer 사이에 BN(Batch Normalization) Layer를 추가해주었다. 각 레이어에는 활성화 함수로 ‘softmax’를 사용하였다.  **테이블이(가) 표시된 사진  자동 생성된 설명**  <결과>  모든 레이어에 활성화 함수로 ‘Softmax’를 사용한 BN Model을 학습해보았다.    <결과>  Large Model보다 BN Model의 Generalization 성능이 크게 떨어진 것을 확인하였다.  이는 ‘Softmax’ 함수는 출력의 각 원소가 0.0 이상 1.0 이하의 실수이기 때문에 여러 레이어에서 활성화 함수를 계속 거치면서 기울기가 거의 0에 수렴하는 기울기 소실(Vanishing Gradient) 문제가 발생한 것으로 예상된다.  기울기 소실 문제를 해결하기 위해 새롭게 Batch Normalization 모델을 설계하였다.  텍스트이(가) 표시된 사진  자동 생성된 설명  중간 레이어의 활성화 함수는 ‘ReLu’ 함수를 사용하였고, 마지막 출력 레이어의 활성화 함수만 ‘Softmax’ 함수를 사용하였다.    <결과>  Large Model보다 BN Model의 Generalization 성능이 증가한 것을 확인하였다.  4.4 Final Model  테이블이(가) 표시된 사진  자동 생성된 설명  이와 같은 MLP 분류 모델을 만들기 위해 다음과 같이 Coding 하였다.  텍스트이(가) 표시된 사진  자동 생성된 설명  테이블이(가) 표시된 사진  자동 생성된 설명  <결과>  Final Model을 학습해보았다.    <결과>  Large Model, Dropout Model보다는 Final Model의 Generalization 성능이 증가했지만, 앞서 설계한 BN Model보다 Final Model의 Generalization 성능이 아주 작은 값이긴 하지만 낮아진 것을 확인하였다.  이는 모든Regularization기법을 적용하여 레이어를 깊게 쌓을 수록 성능이 좋아지는 것은 아니라는 것을 확인할 수 있었다.  •Data Augmentation  Data Augmentation는 갖고 있는 데이터셋을 여러 가지 방법으로 Augment하여 실질적인 학습 데이터셋의 규모를 키울 수 있는 방법이다.  방법으로는 Rotating, Flipping, Scaling, Shifting, Mirroring, Random Cropping, Shearing, Local Warping 등이 있다. | | | |
| #3. 고찰  - Large Model을 설계하면서, Training Data가 충분할수록 Overfitting 현상이 발생할 가능성이 줄어드는 것을 확인할 수 있었다. 과제는 Overfitting을 유도하기 위해 전체 Data 중 일부만 사용했지만 전체 Data의 양이 한정적이고 모든 Data를 Training Data로 사용해도 Overfitting이 발생한다면 여러 Regularization 기법들을 이용해 Overfitting을 방지할 수도 있고, 여러 Data Augmentation기법을 사용해 기존 Data들을 의도적으로 변형시켜 Training Data에 추가해 Data 양을 늘리는 방법도 Overfitting을 방지할 수 있는 방법이라는 것을 알 수 있었다.  - Dropout Model을 설계하면서, Dropout( ) 함수의 첫 번째 인자인 각각의 뉴런이 드랍 될 확률인 dropout\_rate = 0.2로 설정해주었다. Drop-out을 적용했을 때 적용하지 않았을 때보다 좋은 결과가 나오는 것을 확인할 수 있었고, 이 과정에서 dropout\_rate의 값에 따라 다른 결과가 나오는 것을 확인할 수 있었다. 그렇기 때문에 모델의 성능을 높이기 위해 적절한 dropout\_rate을 설정하여 Drop-out을 적용하는 것이 중요하다는 것을 알 수 있었다.  - Batch Normalization 모델을 설계하면서, 모든 레이어의 활성화 함수를 ‘Softmax’로 설정했을 때, BN Model의 Generalization 성능이 눈에 띄게 낮아진 것을 확인할 수 있었다. 이는 ‘Softmax’ 함수는 출력의 각 원소가 0.0 이상 1.0 이하의 실수이기 때문에 여러 레이어에서 활성화 함수를 계속 거치면서 기울기가 거의 0에 수렴하는 기울기 소실(Vanishing Gradient) 문제가 발생한 것으로 예상된다. 그렇기 때문에 모델의 성능을 높이기 위해 각 레이어에서 적절한 활성화 함수를 설정하는 것도 중요하다는 것을 알 수 있었다.  - Final Model을 설계하면서, 모든Regularization기법을 적용하여 레이어를 깊게 쌓을 수록 모델의 성능이 좋아지는 것은 아니라는 것을 확인할 수 있었다. 단순히 층을 깊게 쌓는다고 항상 좋은 결과가 나오는 것은 아니라는 것을 알 수 있었고, Data와 학습 목적에 맞게 적절한 Regularization 기법을 선택하여 모델에 적용하는 것이 중요하다는 것을 알 수 있었다.  - Final Model의 Generalization 성능이 아주 작은 값이긴 하지만 낮아진 이유를 찾아본 결과, 드롭아웃 층과 배치 정규화 층은 모두 정규화 기능을 포함하고 있으므로 두 가지 층을 같이 사용한다면 오히려 학습 성능이 좋지 않을 수 있다는 의견이 있었다. | | | |
| #4. 결론  - Learning MLP    - Large Dataset    - Drop-out    - Batch Normalization    - Final Model을 학습해보았다.    Training Data가 많거나 Data와 학습 목적에 맞게 적절한 Regularization 기법을 선택하여 모델에 적용한다면 Overfitting을 방지하여 모델의 성능이 좋아질 수 있지만, 모든Regularization기법을 적용하여 단순히 레이어를 깊게 쌓을 수록 모델의 성능이 좋아지는 것은 아니라는 것을 확인할 수 있었다. | | | |