딥러닝

신경망 알고리즘의 장단점을 설명

- 장점
  1. 수학적으로 해결 불가능한 복잡한 문제들을 분석하는데 유용하다.
  2. 모델링을 위한 계산이 가능하다.
  3. 은닉층의 결합함수나 은닉층과 출력층 사이의 연결 강도를 추정하기 쉽다.
  4. 학습을 통해 분석하기 때문에 기존 통계학적 분석방법에 비교하여 분석 시간이 짧고, 비용이 적게 든다.
  5. 패턴인식, 예측, 분류 등에 효과적이다.
  6. 범용 근사치 추정 모형으로 사용이 용이하다.
  7. 은닉단위 수를 조절하여 모든 함수를 정밀하게 모사 가능하다.
- 단점
  1. 학습을 통해 찾은 최적의 가중치들은 신뢰도가 낮다.
  2. 새로운 데이터가 추가될 경우 학습을 처음부터 다시 시작해야 한다.
  3. 초기 가중치의 선택이 학습의 성능에 영향을 끼친다.
  4. 다수의 은닉 마디를 필요로 한다.
  5. 입력 변수의 수가 많아질수록 필요한 은닉 마디의 수가 급속하게 증가한다.
  6. 불필요하거나 중복된 입력 자료에 대하여 민감하다.
  7. 실제 상황에 정확히 부합하는 작동을 학습하기 위해 수많은 경우에 대한 다양성을 가진 자료 수집을 필요로 한다.
  8. 인간의 뇌와 같은 신경 그래프를 구축하기 위해 수백, 수천만개의 데이터 베이스 행을 채워야 하는데 이는 엄청난 양의 컴퓨터 메모리와 하드디스크 공간을 소비하게 한다.

트레이닝 데이터와 테스트 데이터의 차이

- Training data: 학습을 할 때 사용하는 데이터
- Test data: 학습한 모델의 성능을 테스트하는 데이터

Sigmoid보다 ReLU를 많이 쓰는 이유

- Vanishing gradient 때문임.
- Backpropagation을 진행할 때 gradient 값이 줄어들게 되면서 train이 잘 되지 않음.
- Sigmoid는 gradient의 절대값이 0~1사이 값으로 되면서 마지막 단계에서는 gradient가 거의 사라짐.
- ReLU는 0이하는 0으로, 그 이상 값은 그 값을 그대로 표현하기 때문에 gradient가 보존됨.

Non-Linearity

- 비선형적인 것, 직선적인 특성을 가지고 있지 않은 것을 의미한다.
- Binary한 classification을 진행하기 위해서는 linear한 함수가 아니어야 하며, True or False로 분류할 수 있어야 한다.
- Activation function에서 다음 노드에 input 값을 부여할지 말지 판단하기 때문에 non-linearity 특성이 필요함.

ReLU의 문제점

- 0보다 작은 activation 값에 대해서는 모두 동일하게 0으로 처리함.
- 이는 정보의 일정부분을 활용하지 않는 것으로 볼 수 있는데, 이를 보완하는 활성화 함수로는 selu가 있으며, 이것은 0보다 작은 값도 정보를 활용한다.

Gradient Decent

- 순간 기울기를 구한 후 오차가 가장 작은 지점을 찾아야 함. 이 때는 미분 시, 기울기가 0이 되는 지점이고, 학습률 만큼 경사를 이동하여 기울기를 변화시키면서 기울기가 0이 되는 지점을 찾아 오차를 가장 작게 함.

왜 Gradient Descent에서는 기울기를 사용하는가?

- 각 변수로의 일차 편미분 값으로 구성되는 벡터임. 이 벡터는 f의 값이 가장 가파르게 증가하는 방향을 나타내며, 벡터의 크기는 증가의 가파른 정도를 나타냄.
- Gradient의 특성을 이용하여 gradient의 반대 방향으로 조금씩 이동시키면 f(x)가 극소가 되는 지점을 찾을 수 있음.

Back propagation

- 신경망을 학습시키기 위한 일반적인 알고리즘 중 하나로, target 값과 실제 모델이 계산한 output이 얼마나 차이

가 나는지 구한 후 그 오차 값을 다시 뒤로 전파해가면서 각 노드가 가지고 있는 변수들을 갱신하는 알고리즘임.
- Chain rule를 이용하여 계산할 수 있으며, 이는 x가 변화했을 때 함수 g가 얼마나 변화하는 지와 그로 인해 함수 g의 변화로 인해 함수 f가 얼마나 변화하는지를 알 수 있고, 함수 f의 인자가 함수 g이면 최종값 F의 변화량에 기여하는 각 함수 f와 g의 기여도를 알 수 있음.

CNN

- 합성곱 신경망으로 불리며, 시각적 이미지를 분석하는데 사용됨.

CNN이 MLP보다 좋은 이유

- 데이터의 특징을 추출 후 그 특징을 기반으로 분류함.
- 데이터 분할 분석을 통합 과적합 가능성 감소
- 텍스트 분류 시, 다수의 단어 조합에 대한 패턴 고려 가능
- 학습 속도가 빠름

CNN 파라미터를 계산하는 방법

- Model.summary()를 통하여 파라미터 수 확인 가능
- 입력 채널 x 필터 폭 x 필터 높이 x 출력 채널 수
- 출력 데이터 크기 계산
  OutputHeight = (높이 + 2 * 패딩사이즈 - 필터 높이) / Stride 크기 + 1
  OutputWeight = (폭 + 2 * 패딩사이즈 - 필터 폭) / Stride 크기 + 1

Pooling layer를 사용하는 이유

- 학습해야할 매개변수가 없고, 채널 수가 변하지 않음.
- 계산량 감소로 인해 가중치 개수와 계산량이 줄어들며, 정보 손실도 있지만 정보 손실보다 계산량 감소에서 오는 이점이 더 크기 때문에 성능이 좋아짐.
- 합성곱 계층의 과적합을 막기 위해

Pooling 시에 Max pooling을 사용하는 이유

- 입력의 변화에 영향을 적게 받음. 입력 데이터가 조금 변해도 풀링의 결과는 잘 변하지 않는다.

Word2Vec에 대한 설명

- 단어를 벡터로 변환하는 계산 알고리즘이며, 2개의 은닉층을 활용한 단순한 계산법으로 대량의 데이터를 사용할 수 있음.
- 중심단어로 주변 단어를 맞추거나, 주변단어를 더 잘 맞추기 위해 가중치 행렬을 조금씩 업데이트 하면서 학습이 이뤄지는 구조
- 중심 단어와 주변 단어 벡터의 내적이 코사인 유사도가 되도록 단어 벡터를 벡터 공간에 임베딩함.
- 은닉층이 하나인 신경망 구조이나, 하이퍼볼릭탄젠트, 시그모이드 등 비선형 활성함수를 사용하지 않아 선형모델임.
- word2vec은 출력층이 내놓은 스코어 값에 소프트맥스 함수를 적용해 확률 값으로 변환한 후 이를 정답과 비교해 역전파 하는 구조임.
- Subsampling frequent words, negative sampling 학습 방법을 사용함.
  - subsampling frequent words: 텍스트 자체가 가진 문제를 해결하는 방법으로, 예를 들어 'a', 'the' 같은 자주 등장하는 단어를 제거함으로써 학습 속도를 향상 시킴.
  - negative sampling: 소프트맥스를 적용하려면 중심단어와 나머지 모든 단어의 내적을 한 뒤 exp를 취해야하기 때문에 계산량이 매우 큼. 이 때문에 소프트 맥스 확률을 구할 때 전체 단어를 대상으로 구하지 않고, 일부 단어만 뽑아서 계산을 함. 사용자가 지정한 윈도우 사이즈 내에 등장하지 않는 단어를 뽑고 이를 target 단어와 합쳐 전체 단어처럼 소프트맥스 확률을 구함.

Training data와 Test data를 분리하는 이유

- 주어진 모델에 대해서만 학습을 할 경우, 학습 데이터에만 성능이 좋은 과적합이 일어남. 이를 방지하기 위해 일반화된 모델을 만들기 위해 학습에 사용하지 않은 데이터로 예측을 수행하여 모델의 성능을 확인해야함.

Validation 데이터가 따로 있는 이유

- 모델의 학습 과정에서 성능을 확인하고, 하이퍼파라미터를 튜닝하는데 사용함. 딥러닝의 하이퍼파라미터 튜닝은 임의로 하는 것이 더 성능이 좋기 때문에 validation data를 통해 하이퍼파라미터 튜닝을 진행하고, 최고의 모델을 선택하여 test data로 성능 확인을 할 수 있도록 함.

Batch Normalization에 대한 설명

- 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화하는 것.
- Gradient를 전체 데이터를 사용하기에는 데이터 학습 소요 시간이 너무 길고, 계산량도 많아지기 때문에 batch 단위로 나눠 학습을 함. 다만 일부의 데이터만을 사용하기 때문에 학습 과정에서 계층 별로 입력의 데이터 분포가 달라지는 Internal Covariant Shift 현상이 나타남. Batch 간의 데이터가 상이하면 연산 전 후가 달라지 수 있기 때문에 Batch normalization을 적용해야함.
- 학습 단계와 추론 단계에서 다르게 적용되어야 함. 학습 단계에서는 배치별로 계산되어야 각 배치들이 표준 정규분포를 각각 따르게 되어있음. 추론 단계에서는 적용할 평균과 분산에 고정값을 사용함. 학습 단계에서는 데이터가 배치 단위로 들어오기 때문에 배치의 평균, 분산을 구하는 것이 가능하지만, 테스트 단계에서는 배치 단위로 평균, 분산을 구하기가 어려워 학습 단계에서 배치 단위의 평균/분산을 저장해놓고 테스트 시에는 평균/분산을 사용함.

Optimizer 중 SGD, RMSProp, Adam에 대해 아는 대로 설명

- SGD: 매 스텝에서 한 개의 샘플을 무작위로 선택하고 그 하나의 샘플에 대한 gradient를 계산함. 매 반복에서 다뤄야 할 데이터가 매우 적기 때문에 한 번에 하나의 샘플을 처리하여 학습 속도가 빠름. 다만 무작위로 샘플을 선택하기 때문에 불안정하고, 방향에 따라 성질, 기울기가 달라지는 함수 이기 때문에 탐색 경로가 비효율적임.
- RMSProp: 과거의 기울기를 제곱하여 계속 더해가 학습을 진행할수록 갱신 강도가 약해지는 AdaGrad의 문제점을 갱신한 기법임. 과거의 모든 기울기를 균일하게 더해가는 것이 아니라, 먼 과거의 기울기는 서서히 잊고 새로운 기울기 정보를 크게 반영함. AdaGrad 계산식에 지수이동 평균을 적용하여 최소 step은 유지할 수 있도록 함.
- Adam: RMSProp과 Momentum 기법을 합친 기법으로, 매개변수 공간을 효율적으로 탐색해주고, 하이퍼파라미터의 편향 보정이 진행되는 기법임.

AutoML에 대한 설명

- 기계학습 파이프 라인에서 수작업과 반복되는 작업을 자동화하는 프로세스.
- 모델을 학습하고 평가할 때 다양한 알고리즘들과 연관된 하이퍼 파라미터들을 실험하고 성능을 비교하여 최상의 성능을 갖는 모델을 찾는 과정을 자동화하는 것, 그리고 인공 신경망 기술을 활용함에 있어서 문제에 적합한 architecture를 찾는 과정을 자동화 하는 문제를 해결하려고 함.

딥러닝 시 GPU를 사용하는 이유

- 딥러닝 알고리즘은 많은 양의 단순 연산을 수행하는데 GPU는 병렬 수치 연산을 고속으로 처리할 수 있음.

Attention

- 기존의 RNN, LSTM에서 발생한 기울기 소실(vanishing gradient) 문제를 해결하기 위해 등장한 모델입니다. 이전 모델에서는 문장이 길어질수록 처음 단어의 가중치의 전달이 없어지고, 문장의 의미를 파악하기 어려웠습니다. 그래서 query와 맵핑된 key, value를 통해 각 문장의 단어 유사도를 구하여 관련 있는 단어들을 집중할 수 있도록 합니다. 이를 통해 동음이의어를 구분하고, 문맥을 이해할 수 있도록 합니다. 다만 문장 전체의 벡터값을 각 token 마다 계산을 하니 bottleneck(병목현상) 문제가 생겨 정보 손실이 이루어지고, 학습이 느리다는 단점이 있습니다.

Transformer

- 기존 Attention 모델을 발전시킨 Self attention 모델로써 정보 손실 문제를 해결하고, RNN 기술을 없앤 후 position embedding을 통해 위치를 학습시켜 문맥을 이해할 수 있도록 합니다. Self attention이란 query와 key, value가 같은 것으로 문장 자체의 위치와 문맥을 이해합니다. 특히 attention head를 여러 개 갖고 있는 multi head attention의 구조이기 때문에 병렬 처리를 하여 고도화된 학습을 할 수 있으며, 학습 시 MLM, NSP를 통해 예측 학습을 진행합니다. Encoder 부분은 문장을 이해하는 NLU의 구조이며, Decoder는 문장을 생성할 수 있는 NLG 구조입니다. Decoder는 확률을 통해 문장 생성 중 다음에 위치할 단어 중 가장 높은 확률을 가진 단어를 배치하며 문장을 생성합니다. 디코딩 방법으로는 Viterbi 알고리즘이 많이 쓰이며, 끝에서부터 역으로 계산하여 가장 확률 높은 단어들을 찾을 수 있도록 합니다.

Few Shot Learning

- 딥러닝 모델은 데이터 양에 비례하여 성능이 향상됩니다. 하지만 데이터 수집 및 레이블링 비용, 데이터 확보의

어려움 및 일반화 능력을 모델에게 갖추게 하기 위해 연구가 진행되었으며, 이를 통해 소량의 데이터만으로 학습하여 좋은 성능을 내도록 연구된 것이 Few-Shot learning입니다. 이미 학습된 pretrain data를 사용하여 가중치 업데이트 없이 약간의 데이터로도 모델에게 원하는 task를 수행할 수 있도록 합니다.

SVD

- 특잇값 분해라하며, 고유값 분해처럼 행렬을 대각화하여 분해하지만, 정방행렬이 아니어도 분해 가능함.
- 추천 시스템에서의 matrix를 분해하는 것에 적절함.
- 행렬을 차원축소하기 위해 쓰이며, 기존 고유값 분해에 비해 행렬의 제약없이 분해할 수 있습니다. 벡터의 방향은 그대로 두고 차원만 축소하여 거리를 줄이며, 특이값으로 이루어진 행렬을 얻을 수 있습니다. 특이값 분해를 이용하여 노이즈 데이터를 제거하여 차원을 축소하고 서로 연결된 상관관계를 찾기 위해 쓰입니다. 특히 데이터가 커질수록 해당 차원을 표현하기 위해 필요한 데이터가 많아지는 차원의 저주 문제를 해결할 수 있어 과적합 문제를 해결할 수 있습니다.

BERT

- BERT는 문맥을 고려한 임베딩 모델임. 문맥과 관계없이 정적 임베딩을 생성하는 Word2vec과는 달리 문맥을 기반으로 동적 임베딩을 생성함. Transformer의 Encoding부분만 사용하고, bidirectional model임. 양방향으로 문장을 읽으며 멀티 헤드 어텐션 메커니즘을 사용해 문장의 각 단어의 문맥을 이해해 문장에 있는 각 단어의 문맥 표현을 출력으로 반환함.
- BERT에 데이터를 입력하기 전 토큰 임베딩, 세그먼트 임베딩, 위치 임베딩을 레이어를 기반으로 변환함.
  1. 토큰 임베딩: 문장을 토큰화해 토큰들을 추출하고, 첫번째 문장의 시작 부분에 [CLS] 토큰을 추가하고, 모든 문장 끝에 [SEP] 토큰을 추가함. 이 토큰들을 임베딩 레이어를 사용해 토큰을 임베딩으로 변환함.
  2. 세그먼트 임베딩: 주어진 두 문장을 구별하는데 사용됨. 세그먼트 임베딩은 입력에 대한 출력으로 EA 도는 EB로만 반환함. A문장에 속하면 EA, B문장에 속하면 EB로 반환함.
  3. 위치 임베딩: 문장에서 단어의 위치에 대한 정보를 제공함.
  이 세 임베딩을 합산해 BERT에 입력으로 제공함.
- 사전 학습 전략으로는 MLM, NSP가 있음.
  1. MLM: 주어진 입력 문장에서 전체 단어의 15%를 무작위로 마스킹하고 마스크된 단어를 예측하도록 모델을 학습시킴. 마스크된 단어를 예측하기 위해 모델은 양방향으로 문장을 읽고 마스크된 단어를 예측하려 시도함. 다만 이렇게 토큰을 마스킹하면 사전 학습과 파인 튜닝 사이에 불일치가 생기게 됨. Mask 토큰을 예측해 BERT를 사전 학습 시키고, 학습 시킨 후에는 감정 분석과 같은 다운 스트림 태스크를 위해 사전 학습된 BERT를 파인 튜닝함. 그런데 파인 튜닝에는 입력에 [MASK] 토큰이 없어 사전 학습 방식과 파인 튜닝 방식간 불일치가 발생함. 이 문제를 해결하기 위해 80-10-10% 규칙을 적용함. 15%중 80%의 토큰을 [MASK] 토큰으로 교체하고, 15% 중 10%의 토큰을 임의의 단어로 교체하고, 15% 중 10%의 토큰을 어떤 변경을 하지 않음. 이후 임베딩 레이어에 입력하여 임베딩 값으로 변환하고 각 토큰의 표현 R을 얻음. 마스크된 토큰 R[MASK]의 표현을 소프트맥스 활설화를 통해 피드포워드 네트워크에 입력하고, 피드포워드 네트워크는 R[MASK] 단어가 마스크된 단어가 될 확률을 반환함. 그리고 MASK될 확률이 가장 높은 단어를 찾아 예측함.
  2. NSP: 이진 분류 테스트로, 두 문장을 입력하고 두 번째 문장이 첫 번째 문장의 다음 문장인지 예측함. NSP 태스크를 수행함으로써 두 문장 사이의 관계를 파악함.

GPT

- Transformer의 Decoder부분을 이용한 사전학습 NLG 모델임.
- Multi layer Transformer decoder를 사용하며, token embedding값과 position embedding 값을 더해준 후 transformer block을 통과하고, vocab size만큼 linear를 하고 softmax를 취해줌.
- 문장의 시작을 알리는 start 토큰 〈s〉, 문장의 끝을 알리고 BERT의 CLS 토큰 역할을 하는 extract 토큰 〈e〉, BERT의 SEP 토큰처럼 문장을 이어주는 delimeter 〈$〉토큰이 있음.
- 문장을 생성하기 때문에 unidirectional model이며, 해당 단어 토큰 뒤에 확률적으로 가장 높게 위치할 토큰을 배치함.

Multi-modal training

- 인간의 인지적 학습법을 모방하여 다양한 형태 데이터로 학습하는 방법
- 다양한 자원으로부터 수집된 데이터가 하나의 정보를 표현하며, 보통 인간 행동 인식 분야에 많이 사용됨.