LoRA 논문 분석

작성자: 20기 김정찬

배경 및 동기

- **대규모 언어 모델의 부상**

  언어 모델(특히 Transformer 계열)은 모델 크기가 커질수록 다양한 자연어 처리(NLP) 작업에서 높은 성능을 보이지만, 매개변수(파라미터)가 수십억~수천억 개에 달할 경우(예: GPT-3 175B), 모델 전체를 미세조정(fine-tuning) 하는 것은 메모리, 저장 공간, 계산 비용 측면에서 매우 비효율적임.

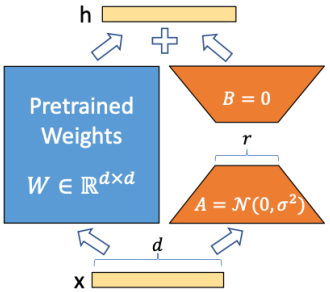
- **매개변수 효율적 학습(PEFT, Parameter-Efficient Fine-Tuning)의 필요성**

  큰 언어 모델(PLM: Pre-trained Language Model)에 대해 모든 파라미터를 학습하는 대신, 일부 파라미터 혹은 추가 모듈만 학습하는 방법들이 제안 (Adapters, Prefix Tuning 등). 하지만 기존 방식들은 아래와 같은 한계를 갖는 경우가 많았음.

  - Inference 시에 추가 지연
  - Forward Pass에서 시퀀스 길이를 축소 (프롬프트 기반의 방법)
  - 모델 품질 하락

- **LoRA(Low-Rank Adaptation)**

  모델의 원본 가중치(pre-trained weights)는 고정(freeze)하고, 저랭크 행렬을 추가로 학습함으로써 큰 언어 모델을 효율적으로 미세조정하면서, 기존 방식 대비 학습해야 할 파라미터 수 감소, GPU 메모리 사용량 절감, 추론 지연 없는 통합(merge) 가능, 높은 성능 유지 등을 달성하는 방안을 제시



A, B만 학습하겠다!

Rank: 행렬이 얼마나 많은 독립적인 정보를 가지고 있는지를 나타내는 값

Low Rank: 큰 행렬을 두 개의 저차원 행렬 곱으로 분해하는 것

문제 정의

- **전형적인 언어 모델의 Fine-tuning**

  입력 x와 목표 레이블(생성해야 할 텍스트) y가 주어졌을 때, 조건부 언어 모델 P(y|x)를 최대화하는 것이 목표

$$\max_{\Phi} \sum_{(x,y)\in Z} \sum_{t=1}^{|y|} \log\big(P_{\Phi}(y_t \mid x, y_{<t})\big),$$

  여기서 Z는 다운스트림(하위 작업) 데이터셋의 샘플 집합 $\{(x_i, y_i)\}$이고, $\Phi = \Phi_0 + \Delta\Phi$로서 Φ_0는 사전학습된 가중치, $\Delta\Phi$는 미세조정 시 학습되는 가중치 변화분임.

  하지만, Φ가 수십~수백억 개 이상의 파라미터를 가진 경우, 모든 파라미터를 학습할 때 저장 공간, GPU 메모리, 최적화 비용 등이 기하급수적으로 증가.

- **LoRA에서의 문제 설정**

  ΔΦ를 직접 전체 차원으로 두는 대신, 훨씬 차원이 작은 새로운 파라미터 Θ를 학습하고, 이를 통해 ΔΦ를 간접적으로 표현하는 방식을 택함:

$$\Delta\Phi = \Delta\Phi(\Theta) \quad \Rightarrow \quad \max_{\Theta} \sum_{(x,y)\in Z} \sum_{t=1}^{|y|} \log\big(P_{\Phi_0+\Delta\Phi(\Theta)}(y_t \mid x, y_{<t})\big).$$

  ⇒ ㅣΘㅣ가 ㅣΦ0ㅣ에 비해 매우 작도록 하여 학습 효율성과 파라미터 효율성을 모두 달성하는 것이 목표.

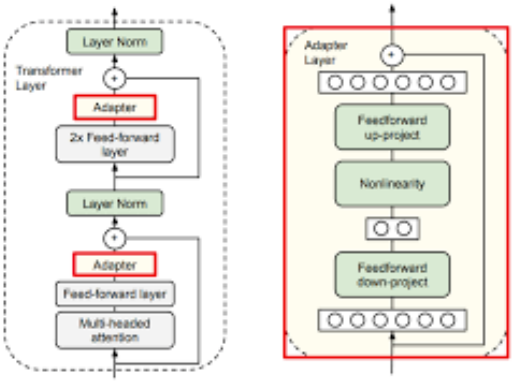
기존 접근법 및 한계

- **Adapter 계열**

  Transformer 각 층에 작은 MLP('어댑터 레이어')를 삽입해 학습. 매개변수가 비교적 적지만, 각 층마다 추가 연산이 들어가므로 추론 지연이 증가. 특히 모델 병렬화 시 추가적인 통신(예: AllReduce) 발생 가능.

Adapter 계열 방법

기존 모델 파라미터는 그대로 유지하고, 각 레이어에 작은 크기의 어댑터 모듈을 삽입하는 방법으로 Adapter 모듈은 주로 두 개의 linear layers으로 구성되며, 중간에 비선형 활성화 함수가 추가
적응 과정: 입력 → 차원 축소(down-projection) → 비선형 변환 → 차원 복원(up-projection) → 원래 출력에 추가(skip connection)

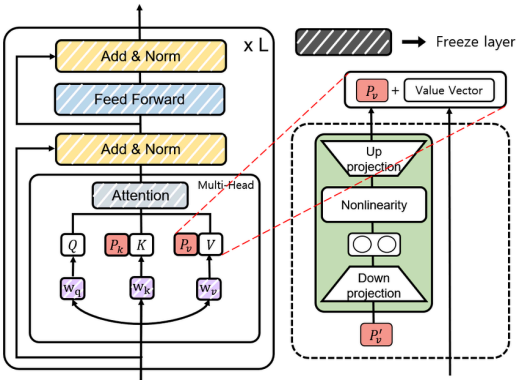


- **프롬프트(prefix) 튜닝 계열**

  입력 토큰 앞뒤에 학습 가능한 벡터를 삽입해, 파라미터 업데이트 없이 언어 모델의 '컨텍스트'만 조정.

  - 시퀀스 길이를 소모
  - 최적화가 어려운 경우가 많음(작은 랭크일 때 불안정한 성능)
  - 많은 학습 토큰을 쓰면 입력 분포가 크게 바뀌어 성능 저하.

프롬프트(prefix) 튜닝 계열 방법
사전학습된 대형 언어 모델(Pretrained Language Models, PLMs) 을 특정 태스크에 맞게 적응시키기 위해
모델 파라미터를 고정하고, 입력 프롬프트(prompt)만 최적화하는 방법



⇒ 추론 시 별도 지연이 없고, 시퀀스 길이를 고정하며, 낮은 메모리 요구를 만족하는 새로운 방법이 필요하다!

LoRA: Low-Rank Adaptation

- **Low-Rank 기반 업데이트 ⇒ 매개변수 효율성**

  Transformer 내부의 대부분의 가중치는 완전결합 행렬 W0 ∈ R^{d×k} 형태인데, fine tuning 중의 가중치 변화 ΔW가 사실상 rank가 낮을 것이라는 가정 아래,

  $$\Delta \mathbf{W} = \mathbf{BA}, \quad \text{where } \mathbf{B} \in \mathbb{R}^{d \times r},\ \mathbf{A} \in \mathbb{R}^{r \times k},\ \text{그리고 } r \ll \min(d, k).$$

  원래 학습은 W0+ΔW를 직접 업데이트해야 하지만, LoRA에서는 A, B만 학습하면 됨. 따라서, 학습해야 할 파라미터가 크게 줄어듦.

- **Forward pass에서의 재정의 ⇒ 학습 메모리 효율성**

  Transformer에서 어떤 입력x에 대해 h=W0x가 원래의 출력이라면, LoRA 적용 시,

  $$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{BAx}.$$

  초기화 시에는 B = 0, A는 정규분포와 같은 작은 분산으로 초기화하므로, 초기 출력은 원본과 동일하게 시작하게 된다. 또, α / r과 같은 스칼라 벡터를 곱해서 learning rate 조정 가능.

- **Inference 시에 추가 지연 없음 ⇒ 추론 속도 유지**
  - 학습을 마치면 'W = W0 + BA'를 미리 계산해 merge하여 저장 가능

- 결과적으로 추론 단계에서 파라미터가 늘지 않고, 추가 연산도 발생하지 않아 전체 Latency가 증가하지 않음

LoRA, Transformer에 적용

- Attention은 Self-Attention 모듈(쿼리 W_q, 키 W_k, 밸류 W_v, 아웃풋 W_o)과 Feed-forward 모듈로 구성. 예를 들어, Attention 모듈에서

$$\mathbf{W}_q \in \mathbb{R}^{d\times d}, \mathbf{W}_v \in \mathbb{R}^{d\times d}$$

  각각 low rank 행렬 부가하면 MLP, LayerNorm 등은 고정

- 메모리 이점: 예를 들어 GPT-3 175B 전체 파라미터(약 1,750억 개) 중 일부(쿼리·밸류 등)만 저랭크로 학습하면,
  - GPU 메모리를 최대 1/3로 감소(옵티마이저 states를 훨씬 적게 유지)
  - 체크포인트(학습 완료 후 저장 파일)도 수천~수만 배 작게 저장 가능

실험 결과

- **GLUE 벤치마크(NLU 과제)**
  - LoRA을 적용한 RoBERTa (125M, 355M), DeBERTa(1.5B) 등의 결과, 일반 full fine-tuning 수준 혹은 그 이상의 성능 달성.
  - 같은 파라미터 크기를 사용할 경우, Adapters 혹은 Prefix Tuning 등 다른 기법 대비 성능이 높거나 학습 안정성이 좋았음.
- **언어 생성(NLG) 과제**
  - GPT-2(중형, 대형)에 대해 E2E NLG Challenge, WebNLG, DART 등의 벤치마크에서, LoRA가 Adapter, Prefix Layer Tuning 등 대비 높은 BLEU, ROUGE, CIDEr 지표 달성.
  - 하이퍼파라미터를 크게 튜닝하지 않아도 수월하게 좋은 성능 달성.
- **GPT-3(175B) 대규모 실험**
  - WikiSQL, MNLI, SAMSum 등에 대해 LoRA를 적용.
  - 파라미터를 크게 줄였음에도 풀 파인튜닝 이상의 성능을 보이거나 거의 동등.
  - 추가 확인: Adapter, Prefix 기법은 파라미터 수를 무작정 늘리면 오히려 성능이 하락하는 현상도 관찰됨. 반면 LoRA는 저랭크 r를 적절히 택해 파라미터를 늘려도 안정적으로 성능이 향상되거나 최소 유지.

Low-rank에 대한 추가 분석

1. **어떤 부분에 LoRA를 적용할까?**

쿼리 Wq vs. 밸류 Wv 등 다양한 부분에 랭크 r를 동일하게 할당해 실험한 결과, Wq,Wv 등 필요한 여러 부분을 동시에 low-rank 학습하면, 하나의 행렬에만 큰 랭크를 부여하는 것보다 더 좋은 성능을 보이는 경우가 많음.

2. **실제 학습된 ΔW는 정말 랭크가 낮나?**
   - GPT-3 실험에서, ΔW를 분해했을 때 **1~4 정도의 작은 랭크만으로도** 기존 성능에 근접하거나 오히려 향상.
   - ΔW의 고유벡터/고윳값(특잇값)을 관찰한 결과, 상위 1~2개 특잇값 벡터가 대부분의 정보량을 차지.
   - 이는 학습 시 downstream task에서 필요한 방향만 증폭하는 모습을 보이는 것으로 해석 가능.
3. **ΔW와 기존 W의 상관 관계?**

   ΔW가 임의의 잡음처럼 작동하는 것이 아니라, W가 가진 특정 하위공간(subspace)을 amplify 한다는 사실을 실험적으로 확인. 즉, "사전학습된 가중치가 이미 알고 있는(미약하게 학습된) 특성을 저랭크 업데이트를 통해 부각한다"라는 가설을 뒷받침.

결론

논문에서 제시된 결과를 통해, 기존 어댑터(adapter)나 프롬프트 튜닝 방식보다 다양한 과제에서 더 나은(또는 유사한) 성능을 보이면서도, 합리적인 메모리·모델 크기로 활용 가능함을 보였다!

추가 연구 방향:

- LoRA와 다른 PEFT 기법의 결합(예: Prefix-Tuning + LoRA) 시너지
- 저랭크 업데이트가 실제 언어 모델 내부에서 작동하는 기제 해석
- 더 큰 모델 구조(MLP·LayerNorm·Bias 등)에도 LoRA 확장 적용
- 저랭크 기법을 다른 딥러닝 영역(비전·멀티모달 등)으로의 일반화