

HELLO

<https://arxiv.org/abs/2305.11627>

LLM-Pruner

On the Structural Pruning of Large Language

김민성

| 02-826-3741

| minsung1066@gmail.com |

목 차

01

Pruning 이란?

02

Parameter 수 감소 장점

03

Parameter 수 감소 단점

04

pruning 예시

05

Introduction

06

method

07

Experiments

08

Conclusion

Pruning이란?



의미 = 가지치기

영어로 Pruning은 가지치기라는 의미입니다. 모델 경량화의 방법중 하나이며, On device ai를 위한 방법입니다.

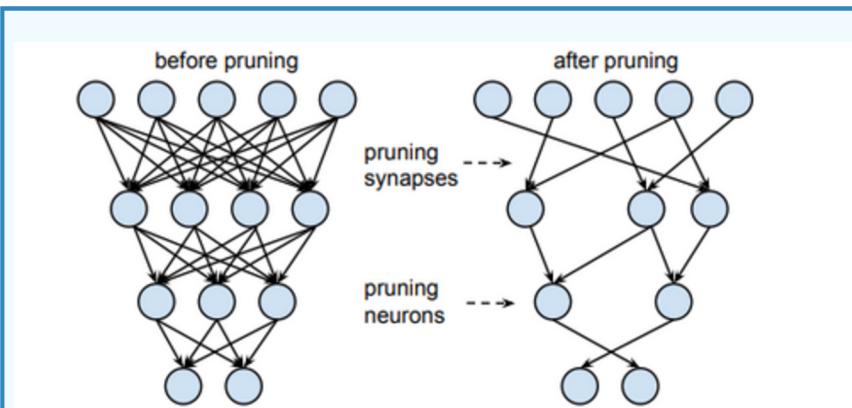


Figure 3: Synapses and neurons before and after pruning.

Pruning 원리

모델의 weight(가중치)들 중 중요도가 낮은 weight의 연결을 제거하여 모델의 파라미터를 줄이는 방법.



Pruning 방법

- 1) Magnitude Pruner : 어떠한 값을 기준으로 thresholding
- 2) Sensitivity : 모델에 대해 영향력을 분석, 가중치 표준편차로 thresholding

모델의 파라미터가 줄어들에 따라 미치는 장점

1

모델의 크기 감소 : 불필요한 가중치를 제거하면 모델의 저장 공간이 줄어듭니다.

2

추론 속도 향상 : 연산량이 줄어들어 실행 속도가 빨라집니다.

3

메모리 절약 : GPU / CPU 메모리 사용량이 감소하여 더 적은 자원으로 모델을 실행 할 수 있습니다.

4

전력 소비 감소 : 임베디드 시스템이나 모바일 환경에서 배터리 효율성이 향상됩니다.

네트워크 가중치를 가지치기(pruning)하여 경량화하는 방법을 제안한 초기 논문 중 하나

. <https://arxiv.org/abs/1506.02626>

모델의 파라미터가 줄어들에 따라 미치는 단점

1

성능 저하 : 너무 많은 파라미터를 제거하면 모델의 표현력이 감소하여 정확도가 떨어질 수 있어, 복잡한 패턴을 학습하는 데 필요한 정보가 손실될 가능성이 큽니다.

2

과도한 Pruning에 의한 정보 손실 : pruning 기법이 완벽하지 않아서 유용한 가중치까지 제거할 가능성이 있음.

3

재학습 비용 증가 : 모델의 pruning한 후에는 다시 학습(fine-tuning)해야 합니다. 이 과정에서 추가적인 계산 비용이 발생할 수 있습니다.

4

너무 공격적으로 pruning을 하면 특정 데이터셋에서는 성능이 좋지만, 새로운 데이터에 대한 일반화 능력이 떨어질 수 있습니다.

Pruning 예제 (PyTorch 구현)

Pytorch에서는 `torch.nn.utils.prune` 모듈을 통해 프루닝 구현

```
import torch
import torch.nn.utils.prune as prune

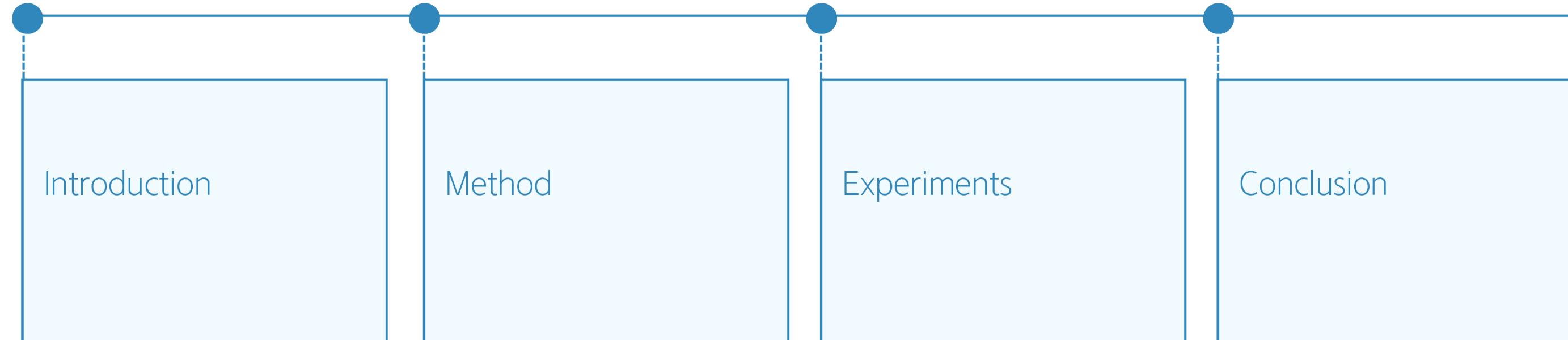
# 모델 정의
model = torch.nn.Linear(10, 10)

# 가중치의 30%를 L1 norm 기준으로 프루닝
prune.l1_unstructured(model, name='weight', amount=0.3)

# 프루닝된 가중치 확인
print(f"프루닝된 가중치 비율: {float(torch.sum(model.weight == 0)) / float(model.weight.nelement()):.2f}")
```

파이토치 링크 https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.ln_structured.html

LLM-Pruner: On the Structural Pruning of Large Language Models



INTRODUCTION

대규모 언어 모델(LLM)은 뛰어난 성능을 보였습니다. 그러나 놀라운 성능은 일반적으로 방대한 모델 크기를 필요로 하며, 이는 배포, 추론, 학습 과정에서 상당한 어려움을 초래합니다. 모델을 압축하는 방법을 탐구하여, 원본 LLM의 다중 해결 및 언어 생성 능력을 유지하는 것을 목표로 합니다.

이 논문에서 제안한 방법인 LLM-Pruner는 structural pruning(구조적 가지치기) 기법을 활용하여, gradient(기울기) 정보를 기반으로 중요하지 않는 결합 구조를 선택적으로 제거함으로써 LLM의 기능 대부분을 최대한 보존하려 합니다.

pruning 후 모델의 성능을 LoRA 기반 튜닝 기술을 활용하여 단 3시간만에 효과적으로 복구할 수 있으며, 이때 필요한 데이터는 50,000개 샘플에 불과합니다.

LLaMa, Vicuna, ChatGLM을 포함한 세 가지 LLM에서 LLM-Pruner를 검증하였고, 그 결과 압축된 모델이 여전히 Zero-shot[기존의 데이터와 지식을 바탕으로 새로운 상황 추론]분류 및 생성(task generation)에서 만족할 만한 성능을 유지함을 확인하였습니다.

Structed pruning 특징

1

단위 : Structed pruning은 그룹(예: 필터, 채널) 단위로 가중치를 제거. 이는 모델의 특정 부분을 전체적으로 제거하는 것을 의미

2

어떤 기준으로 pruning을 할 것인가: 가중치의 크기(L1 노름, L2 노름), 배치 정규화의 스케일링 계수등 다양한 기준을 사용하여 중요하지 않는 단위를 식별하고 제거

3

적용 기준 : Pruning을 전체 네트워크에 걸쳐 일괄적으로(global) 적용할 수도 있고, 각 레이어별로(local) 다른 기준을 적용할 수도 있음.

4

어떤 시점에 pruning을 할 것인가 : pruning : 어느 정도 학습된 후에 적용할 수도 있고, 모델의 초기화 단계에서부터 적용할 수도 있음

Method

<https://arxiv.org/abs/2305.11627>

TASK-SPECIFIC COMPRESSION

특정 작업(Task)에 맞게 모델을 압축하고 별도의 파인튜닝을 진행함.
작업에 대한 추가 데이터셋이 필요하며, 범용적인 LLM 능력을 유지하기 어려움

TINY BERT

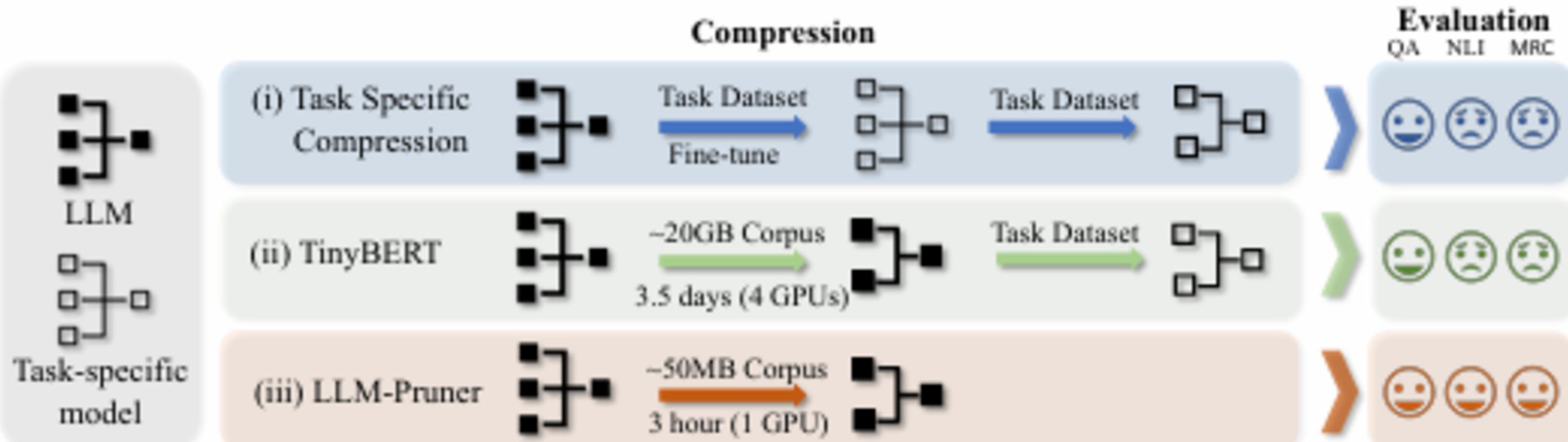
KD 기반으로 작은 모델로 압축하여, 약 20GB의 데이터를 활용하여 지속적으로 훈련
이 필요함(3.5일 4GPU). 시간도 오래 걸리고 일부 성능이 손실됨

LLM-PRUNRT (제안된 방법)

50MB의 작은 데이터셋만 사용하고, 단 3시간으로 훈련 가능(1 GPU). 여기서는 크래
디언트 기반의 구조적 프루닝(가중치 중요도를 손실 변화량을 기반으로 평가)를 사용
하고, LoRA로 성능을 복구함. 제거시에도 모델의 사전 학습된 지식을 최대로 보존함.
task-specific한 다른 파인튜닝 모델과 다르게, 제로샷 성능을 유지 가능

Method

<https://arxiv.org/abs/2305.11627>



LLaMA-5.4B by LLM-Pruner

The Leaning Tower of Pisa is known for its unusual tilt, which is a result of a number of factors. When the tower was built in the twelfth century, the soil beneath it was extremely soft, allowing the buttresses to settle unevenly. This resulted in a tilt towards one side.

LLaMA-7B

The Leaning Tower of Pisa is known for being tilted and unstable. However, its story is much more fascinating. Although construction began in 1173, the tower was never meant to be tilted. It simply became that way because it was built on unstable ground.

Method

<https://arxiv.org/abs/2305.11627>

PRUNING 과정은 두 단계로 나뉩니다.

1. DISCOVER ALL COUPLED STRUCTER IN LLMS

모델에서 어떤 부분이 서로 연관성이 가지는지 찾아야 합니다. LLM-Pruner은 각 뉴런의 입/출력 차수를 정의하여 coupled structures를 자동으로 찾아냅니다. 두 뉴런의 종속성은 아래 식으로 정의 합니다.

Structure Dependency in LLMs. Similar to [II], the pruning begins by building the dependency for LLMs. Assume N_i and N_j are two neurons in the model, $\text{In}(N_i)$ and $\text{Out}(N_i)$ represents all the neurons that point towards or point from N_i . The dependency between structures can be defined as:

$$N_j \in \text{Out}(N_i) \wedge \text{Deg}^-(N_j) = 1 \Rightarrow N_j \text{ is dependent on } N_i \quad (1)$$

where $\text{Deg}^-(N_j)$ represents the in-degree of neuron N_j . Noting that this dependency is directional, we can therefore correspondingly obtain another dependency:

$$N_i \in \text{In}(N_j) \wedge \text{Deg}^+(N_i) = 1 \Rightarrow N_i \text{ is dependent on } N_j \quad (2)$$

where $\text{Deg}^+(N_i)$ represents the out-degree of neuron N_i . The principle of dependency here is, if a current neuron (e.g., N_i) depends solely on another neuron (e.g., N_j), and the neuron N_j is subjected to pruning, it follows that the neuron N_i must also undergo pruning.

Method

<https://arxiv.org/abs/2305.11627>

PRUNING 과정은 두 단계로 나뉩니다.

2. GROUPED IMPORTANCE ESTIMATION OF COUPLED STRUCTURE

그룹화가 끝나면, importance 기반으로 어떤 구조를 삭제할 것인지 정해야 합니다. Vector-wise Importance와 Element-wise Importance를 모두 고려하여 Group Importance를 측정합니다.

Vector-wise Importance. Suppose that given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, where N is the number of samples. In our experiments, we set N equal to 10 and we use some public datasets as the source of \mathcal{D} . A group (as previously defined as a set of coupled structures) can be defined as $\mathcal{G} = \{W_i\}_{i=1}^M$, where M is the number of coupled structures in one group and W_i is the weight for each structure. While pruning, our goal is to remove the group that has the least impact on the model's prediction, which can be indicated by the deviation in the loss. Specially, to estimate the importance of W_i , the change in loss can be formulated as [24]:

$$I_{W_i} = |\Delta \mathcal{L}(\mathcal{D})| = |\mathcal{L}_{W_i}(\mathcal{D}) - \mathcal{L}_{W_i=0}(\mathcal{D})| = \left| \underbrace{\frac{\partial \mathcal{L}^\top(\mathcal{D})}{\partial W_i} W_i}_{\neq 0} - \frac{1}{2} W_i^\top H W_i + \mathcal{O}(\|W_i\|^3) \right| \quad (3)$$

where H is the hessian matrix. Here, \mathcal{L} represents the next-token prediction loss. The first term is typically neglected in prior work [24, 52, 12], as the model has already converged on the training dataset, where $\partial \mathcal{L}^\top / \partial W_i \approx 0$. However, since \mathcal{D} here is not extracted from the original training data, which means that $\partial \mathcal{L}^\top / \partial W_i \neq 0$. This presents a desirable property for determining the importance of W_i by the gradient term under LLMs, since computation of the second term, the Hessian matrix, on the LLM is impractical with $\mathcal{O}(N^2)$ complexity.

Method

<https://arxiv.org/abs/2305.11627>

PRUNING 과정은 두 단계로 나뉩니다.

2. GROUPED IMPORTANCE ESTIMATION OF COUPLED STRUCTURE

그룹화가 끝나면, importance 기반으로 어떤 구조를 삭제할 것인지 정해야 합니다. Vector-wise Importance와 Element-wise Importance를 모두 고려하여 Group Importance를 측정합니다.

Element-wise Importance. The above can be considered as an estimate for the weight W_i . We can derive another measure of importance at a finer granularity, where each parameter within W_i is assessed for its significance:

$$I_{W_i^k} = |\Delta \mathcal{L}(\mathcal{D})| = |\mathcal{L}_{W_i^k}(\mathcal{D}) - \mathcal{L}_{W_i^k=0}(\mathcal{D})| = \left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i^k} W_i^k - \frac{1}{2} W_i^k H_{kk} W_i^k + \mathcal{O}(\|W_i^k\|^3) \right| \quad (4)$$

Here, k represents the k -th parameter in W_i . The diagonal of the hessian H_{kk} can be approximated by the Fisher information matrix, and the importance can be defined as:

$$I_{W_i^k} = |\mathcal{L}_{W_i^k}(\mathcal{D}) - \mathcal{L}_{W_i^k=0}(\mathcal{D})| \approx \left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i^k} W_i^k - \frac{1}{2} \sum_{j=1}^N \left(\frac{\partial \mathcal{L}(\mathcal{D}_j)}{\partial W_i^k} W_i^k \right)^2 + \mathcal{O}(\|W_i^k\|^3) \right| \quad (5)$$

Group Importance. By utilizing either $I_{W_i^k}$ or I_{W_i} , we estimate the importance at the granularity of either a parameter or a weight. Remembering that our goal is to estimate the importance of \mathcal{G} , we aggregate the importance scores in four ways: (i) Summation: $I_{\mathcal{G}} = \sum_{i=1}^M I_{W_i}$ or $I_{\mathcal{G}} = \sum_{i=1}^M \sum_k I_{W_i^k}$, (ii) Production: $I_{\mathcal{G}} = \prod_{i=1}^M I_{W_i}$ or $I_{\mathcal{G}} = \prod_{i=1}^M \sum_k I_{W_i^k}$, (iii) Max: $I_{\mathcal{G}} = \max_{i=1}^M I_{W_i}$ or $I_{\mathcal{G}} = \max_{i=1}^M \sum_k I_{W_i^k}$; (iv) Last-Only: Since deleting the last executing structure in a dependency group is equivalent to erasing all the computed results within that group, we assign the importance of the last executing structure as the importance of the group: $I_{\mathcal{G}} = I_{W_l}$ or $I_{\mathcal{G}} = \sum_k I_{W_l^k}$, where l is the last structure. After assessing the importance of each group, we rank the importance of each group and prune the groups with lower importance based on a predefined pruning ratio.

Method

<https://arxiv.org/abs/2305.11627>

3. FAST RECOVERY WITH LOW-RANK APPROXIMATION

pruning 이후 모델 성능이 떨어집니다. 성능 회복을 위해서 Low-rank Approximation을 통해 다시 성능을 회복시키면 됩니다.

In order to expedite the model recovery process and improve its efficiency under limited data, it is crucial to minimize the number of parameters that need optimization during the recovery phase. To facilitate this, we employ the low-rank approximation, LoRA[19], to post-train the pruned model. Each learnable weight matrix in the model, denoted as W , encompassing both pruned and unpruned linear projection in the LLM, can be represented as W . The update value ΔW for W can be decomposed as $\Delta W = PQ \in \mathbb{R}^{d^- \times d^+}$, where $P \in \mathbb{R}^{d^- \times d}$ and $Q \in \mathbb{R}^{d \times d^+}$. The forward computation can now be expressed as:

$$f(x) = (W + \Delta W)X + b = (WX + b) + (PQ)X \quad (6)$$

where b is the bias in the dense layer. Only training P and Q reduces the overall training complexity, reducing the need for large-scale training data. Besides, the extra parameters P and Q can be reparameterized into ΔW , which would not cause extra parameters in the final compressed model.

LoRA 논문 링크 : <https://arxiv.org/abs/2106.09685>

LoRA 실제 학습 방법에 관한 내용은 유튜브에 잘 찾으면 나올겁니다. LoRA 방법에 약간 의문이 LoRA 방법 자체로 작은 fine tuning 모델을 사용해서 학습시키는 것인데, 이런 방법을 사용하고 LoRA를 사용했지만 성능 유지된다고 말을 할수있을지 의문입니다.

Experiments

<https://arxiv.org/abs/2305.11627>

LLM-PRUNER의 성능 실험

task-agnostic setting에서 일반 상식 추론 데이터셋(BoolQ, PIQA, Hella Swag 등)을 이용

Pruning Ratio : 모델에서 제거할 파라미터의 비율

Method : L 원본 모델 성능, 랜덤 선택한 파라미터 제거 방법(Random), 의존성 구조를 고려한 제거 방법(Channel) 등의 방법을 비교

Benchmark Dataset : WikiText2, PTB 언어 모델에 대한 일반적인 평가, BoolQ, PIQA 다양한 추

론 및 질문 응답

Pruning Ratio	Method	WikiText2 ↓	PTB ↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Ratio = 0%	LLaMA-7B [49]	-	-	76.5	79.8	76.1	70.1	72.8	47.6	57.2	68.59
	LLaMA-7B*	12.62	22.14	73.18	78.35	72.99	67.01	67.45	41.38	42.40	63.25
Ratio = 20% w/o tune	L2	582.41	1022.17	59.66	58.00	37.04	52.41	33.12	28.58	29.80	42.65
	Random	27.51	43.19	61.83	71.33	56.26	54.46	57.07	32.85	35.00	52.69
Ratio = 20% w/ tune	Channel	74.63	153.75	62.75	62.73	41.40	51.07	41.38	27.90	30.40	45.38
	Vector	22.28	41.78	61.44	71.71	57.27	54.22	55.77	33.96	38.40	53.25
	Element ²	19.77	36.66	59.39	75.57	65.34	61.33	59.18	37.12	39.80	56.82
	Element ¹	19.09	34.21	57.06	75.68	66.80	59.83	60.94	36.52	40.00	56.69
Ratio = 20% w/ tune	Channel	22.02	38.67	59.08	73.39	64.02	60.54	57.95	35.58	38.40	55.57
	Vector	18.84	33.05	65.75	74.70	64.52	59.35	60.65	36.26	39.40	57.23
	Element ²	17.37	30.39	69.54	76.44	68.11	65.11	63.43	37.88	40.00	60.07
	Element ¹	17.58	30.11	64.62	77.20	68.80	63.14	64.31	36.77	39.80	59.23

task-agnostic setting에서 일반 상식 추론 데이터셋(BoolQ, PIQA, Hella Swag 등)을 이용

Pruning Ratio : 모델에서 제거할 파라미터의 비율

Method : L 원본 모델 성능, 랜덤 선택한 파라미터 제거 방법(Random), 의존성 구조를 고려한 제거 방법(Channel) 등의 방법을 비교

Benchmark Dataset : WikiText2, PTB 언어 모델에 대한 일반적인 평가, BoolQ, PIQA 다양한 추론 및 질문 응답

Table 2: Zero-shot performance of the compressed LLaMA-13B. Here we adopt Element¹ as the importance estimation for ‘Channel’ and ‘Block’.

Pruning Ratio	Method	WikiText2↓	PTB↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Ratio = 0% w/o tune	LLaMA-13B*	11.58	20.24	68.47	78.89	76.24	70.09	74.58	44.54	42.00	64.97
	L2	61.15	91.43	61.50	67.57	52.90	57.54	50.13	31.14	36.80	51.08
	Random	19.24	31.84	63.33	73.18	63.54	60.85	64.44	36.26	38.00	57.09
	Channel	49.03	106.48	62.39	66.87	49.17	58.96	49.62	31.83	33.20	50.29
Ratio = 20% w/ tune	Block	16.01	29.28	67.68	77.15	73.41	65.11	68.35	38.40	42.40	61.79
	L2	20.97	38.05	73.24	76.77	71.86	64.64	67.59	39.93	40.80	62.12
	Random	16.84	31.98	64.19	76.06	68.89	63.30	66.88	38.31	40.80	59.78
	Channel	17.58	29.76	69.20	76.55	68.89	66.38	62.08	38.99	39.60	60.24
	Block	15.18	28.08	70.31	77.91	75.16	67.88	71.09	42.41	43.40	64.02

task-agnostic setting에서 일반 상식 추론 데이터셋(BoolQ, PIQA, Hella Swag 등)을 이용

Pruning Ratio : 모델에서 제거할 파라미터의 비율

Method : L 원본 모델 성능, 랜덤 선택한 파라미터 제거 방법(Random), 의존성 구조를 고려한 제거 방법(Channel) 등의 방법을 비교

Benchmark Dataset : WikiText2, PTB 언어 모델에 대한 일반적인 평가, BoolQ, PIQA 다양한 추론 및 질문 응답

Table 4: Zero-shot performance of the compressed Vicuna-7B

Pruned Model	Method	WikiText2 ↓	PTB ↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Ratio = 0%	Vicuna-7B	16.11	61.37	76.57	77.75	70.64	67.40	65.11	41.21	40.80	62.78
Ratio = 20% w/o tune	l2	3539.98	5882.21	55.90	56.15	32.37	51.85	30.01	28.41	28.20	40.41
	random	34.63	112.44	61.47	70.89	54.67	56.27	55.60	31.74	34.60	52.18
	Channel	71.75	198.88	51.77	63.93	42.58	55.17	43.94	29.27	33.40	45.72
Ratio = 20% w/ tune	Vector	27.03	<u>92.51</u>	62.17	71.44	55.80	53.43	55.77	33.28	37.80	52.81
	Element ²	<u>24.70</u>	94.34	<u>62.87</u>	<u>75.41</u>	<u>64.00</u>	<u>58.41</u>	60.98	<u>37.12</u>	<u>39.00</u>	<u>56.83</u>
	Element ¹	25.74	92.88	61.70	75.30	63.75	56.20	<u>63.22</u>	36.60	37.00	56.25
Ratio = 20% w/ tune	Vector	19.94	74.66	63.15	74.59	61.95	60.30	60.48	36.60	39.40	56.64
	Element ²	18.97	76.78	60.40	75.63	65.45	63.22	63.05	37.71	39.00	57.78
	Element ¹	19.69	78.25	63.33	76.17	65.13	60.22	62.84	37.12	39.20	57.71

conclusion

<https://arxiv.org/abs/2305.11627>

이 논문은 LLM-Pruner라는 구조적 프루닝 방법을 제안하고 있습니다. LLM-Pruner는 task-agnostic하게 압축하여, 원래의 훈련 데이터에 대한 의존도를 최소화하고 언어 능력을 보전하는 것을 목표로 합니다.

프루닝을 위해서 모델 내에서 각 뉴런을 검토하여 의존성 그룹(coupled structure)을 식별하고, LLM의 의존성 그래프를 구성합니다. 이후 group importance를 gradient 변화량 크기를 기반으로 중요도를 식별해서 일정 수치 이하를 제거합니다.

그 이후 LoRA를 활용하여 가지치기된 모델의 recovery를 빠르게 진행할 수 있습니다.

의문 : LLM 작은 모델로 만들어서, LoRA 모델로 합치는 것과 다른 것인가?

I'M SO TIRED

<https://arxiv.org/abs/2305.11627>

감사합니다