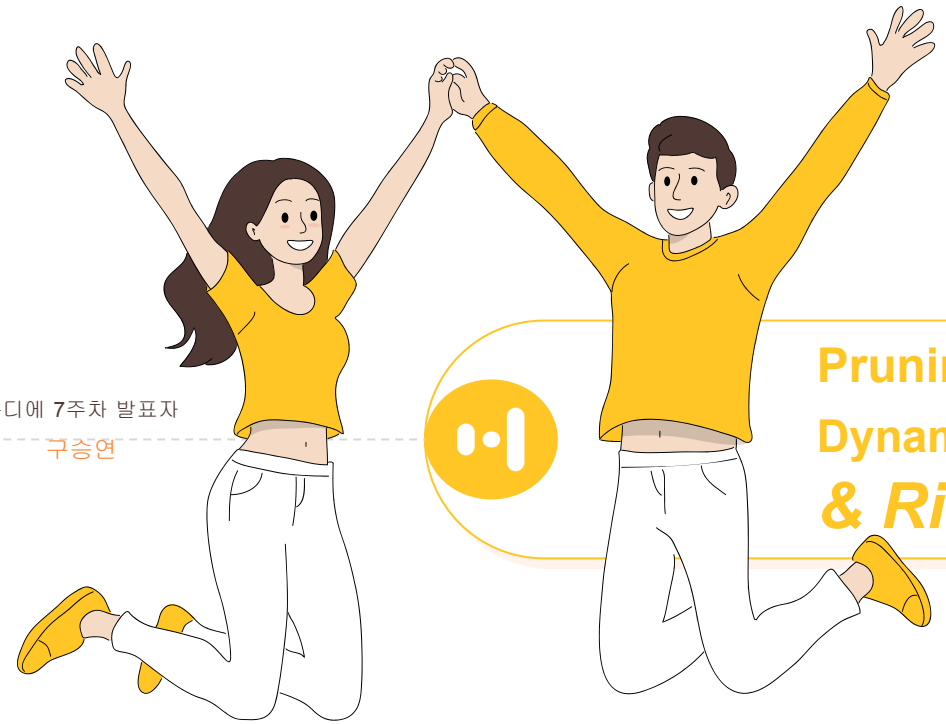


온디에 7주차 발표자

구승연

Pruning During Training:
Dynamic Sparse Training based Methods
& RigL

RigL :: Introduction

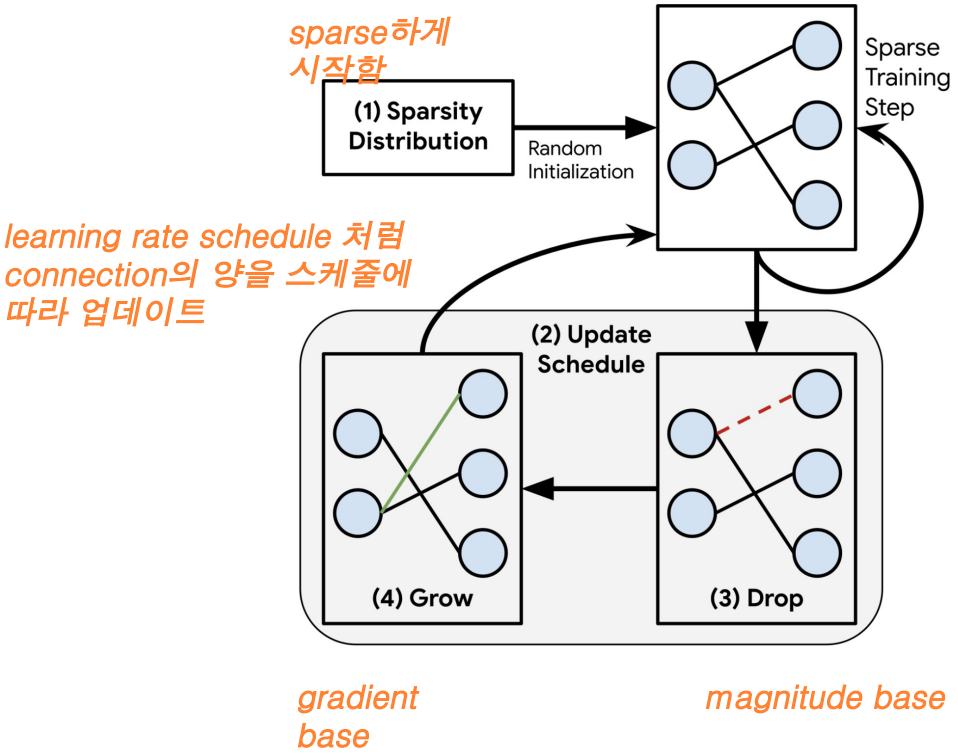
- 지난 주의 Pruning During Training은 group regularization term (L1-norm)을 사용한 method
- RigL은 **학습 중에 네트워크의 "토폴로지"를 업데이트 한다**

2025/04/16	Pruning During Training: Sparsity Regularization based Methods	박예리	온라인	W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in NIPS, 2016.
2025/04/23	Pruning During Training: Dynamic Sparse Training based Methods	구승연	오프라인	U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in ICML, 2020.

지난 주 주제도 Pruning During Training

channel-wise structured sparsity can be defined as

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda_n \cdot \sum_{l=1}^{L} \left(\sum_{n_l=1}^{N_l} ||\boldsymbol{W}^{(l)}_{n_l,:,:,:}||_g \right) + \lambda_c \cdot \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} ||\boldsymbol{W}^{(l)}_{:,c_l,:,:}||_g \right).$$



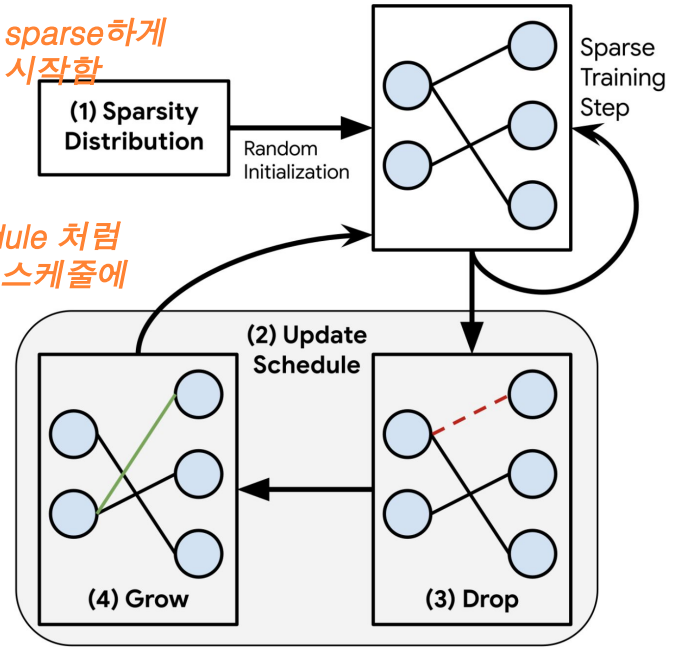
- 뒤에서 ***왜 이런 방식을 제안하게 됐는지*** , ***어떤 장점이 있는지*** 논문을 톺아보는 방식으로 소개

참고 : 이 논문에서는 CNN, RNN, Unstructured와 Structured pruning에 대해 검증하였습니다.

목차

- RigL :: Introduction
  - *Pruning During Training*
  - *한 줄 요약*
- 왜 Sparse 하게 시작할까 —(1)
  - *개념적 이해*
  - *실험 결과*
- 왜 Connection을 Schedule에 따라서 Update 할까? —(2)
  - *개념적 이해*
  - *실험 결과*
- 왜 Connection을 Grow&Drop 할까? —(3),(4)
  - *개념적 이해*
  - *실험 결과*
- 1주차 논문과는 무슨 관계인가?
  - *복습 : The Lottery Ticket Hypothesis (LTH)*
  - *LTH와 Rigging the Lottery (RigL) 사이의 관계*

sparse하게 시작함



learning rate schedule 처럼 connection의 양을 스케줄에 따라 업데이트

gradient base

magnitude base

1주차에서 'Lottery' Ticket Hypotesis에 대해 다뤘다

왜 Sparse 하게 시작할까

- “There is large body of work on training dense networks to yield sparse networks for inference”
  - 대부분의 기존 방법들은 (pruning된) sparse network를 얻기 위해 dense network를 먼저 학습시켜야 한다
- “The maximum size of sparse model is limited to the largest dense model that can be trained”
  - 이러한 방식은 ‘학습 가능한’ dense network의 크기로 최대 크기가 한정 된다.
- “Large amount of computation must be performed for parameters that are zero valued or that will be zero during inference.”
  - zero가 될 파라미터도 학습 중에는 계산량을 소모하기 때문에 비효율적이다.
- 그리고 이 논문에서 추가로 의심 중인 부분은, “It remains unknown if the performance of the current best pruning algorithms is an upper bound on the quality of sparse models.”
  - 세 가지 알고리즘에 대해 sparsity-accuracy 트레이드오프가 있다는 것을 확인한 논문이 있지만 더 나은 성능이 가능하다는

This paradigm has two main limitations. First, the maximum size of sparse models is limited to the largest dense model that can be trained; even if sparse models are more parameter efficient, we can’t use pruning to train models that are larger and more accurate than the largest possible dense models. Second, it is inefficient; large amounts of computation must be performed for parameters that are zero valued or that will be zero during inference. Additionally, it remains unknown if the performance of the current best pruning algorithms is an upper bound on the quality of sparse models. Gale et al. (2019) found that three different dense-to-sparse training algorithms all achieve about the same sparsity / accuracy trade-off. However, this is far from conclusive proof that no better performance is possible.

논문에서 다음과 같은 이유를 제시하고 있다

*

(1) Sparsity Distribution

- Sparsity Distribution은 모델의 'Sparsity'를 각 layer 에 따라 분배하는 방법을 의미
- 크게 두(세)가지 방법을 제안함

1. Uniform
   - 각 layer 별 sparsity가 동일함
   - 단, input layer는 dense(sparsity=0)
     전체 모델에 끼칠 수 있는 영향이 큰 것에 비해 크기에 미치는 영향은 작기 때문에

2. Erdos-Renyi (에르되시-레니?)
   - 그래프 엣지를 랜덤하게 생성하는 이론 : 어떤 노드 쌍을 골라도 엣지가 생성될 확률이 동일하도록 한다
   - 수식은 오른쪽
   - 이 방법을 사용하면 input+output 채널 수에 비례한 sparsity가 s^l scales with $1 - \frac{n^{l-1}+n^l}{n^{l-1}*n^l}$
   - 결과적으로 파라미터(node) 수가 많으면 더 많은 파라미터를 살린다

3. Erdos-Renyi-Kernel(ERK)
   - Erdos-Renyi에 Kernel dimension 정보를 추가: $1 - \frac{n^{l-1}+n^l+w^l+h^l}{n^{l-1}*n^l*w^l*h^l}$
   - FCN은 2번 방법을, convolution layer는 ERK.

*

(1) Sparsity Distribution

input : 2, output:100
input:40, output: 5
이 두가지 경우를 비교했을 때, 엣지의 수는 같지만 노드의 수는 다릅니다.
이 케이스로 왜 파라미터 수가 더 많은 층에 많은 연결을 배분하면 좋은지 설명해 주세요

10초 동안 생각함 >

왜 "노드(채널) 수가 많은 층"에 더 많은 연결(파라미터)을 배분해야 하나?

두 예시를 통해 직관적으로 살펴보겠습니다.

사례	입력 노드 수	출력 노드 수	총 노드	가능한 연결 수 (dense 기준)	평균 fan-in/out (=연결 수 / 노드 수)
A	2	100	**102**	2 × 100 = **200**	**≈ 1.96**
B	40	5	**45**	40 × 5 = **200**	**≈ 4.44**

노드 당 2개의 연결 → 노드 하나 당 학습할 수 있는 기회 작음

s^l scales with $1 - \frac{n^{l-1}+n^l}{n^{l-1}*n^l}$

> 두 층 모두 **파라미터(엣지) 수는 200개로 동일**하지만, **노드 수는 크게 다르다**는 점이 핵심입니다.

*

(1) Sparsity Distribution

다른 dynamic pruning method와 비교 (SET, SNFS, SNIP)

Method	Top-1 Accuracy	FLOPs (Train)	FLOPs (Test)	Top-1 Accuracy	FLOPs (Train)	FLOPs (Test)
Dense	76.8±0.09	1x (3.2e18)	1x (8.2e9)			
	S=0.8			S=0.9		
Static	70.6±0.06	0.23x	0.23x	65.8±0.04	0.10x	0.10x
SNIP	72.0±0.10	0.23x	0.23x	67.2±0.12	0.10x	0.10x
Small-Dense	72.1±0.12	0.20x	0.20x	68.9±0.10	0.12x	0.12x
SET	72.9±0.39	0.23x	0.23x	69.6±0.23	0.10x	0.10x
RigL	74.6±0.06	0.23x	0.23x	72.0±0.05	0.10x	0.10x
Small-Dense$_{5\times}$	73.9±0.07	1.01x	0.20x	71.3±0.10	0.60x	0.12x
RigL$_{5\times}$	**76.6±0.06**	1.14x	0.23x	**75.7±0.06**	0.52x	0.10x
Static (ERK)	72.1±0.04	0.42x	0.42x	67.7±0.12	0.24x	0.24x
DSR*	73.3	0.40x	0.40x	71.6	0.30x	0.30x
RigL (ERK)	75.1±0.05	0.42x	0.42x	73.0±0.04	0.25x	0.24x
RigL$_{5\times}$ (ERK)	**77.1±0.06**	2.09x	0.42x	**76.4±0.05**	1.23x	0.24x
SNFS*	74.2	n/a	n/a	72.3	n/a	n/a
SNFS (ERK)	75.2±0.11	0.61x	0.42x	72.9±0.06	0.50x	0.24x
Pruning*	75.6	0.56x	0.23x	73.9	0.51x	0.10x
Pruning$_{1.5\times}$*	**76.5**	0.84x	0.23x	**75.2**	0.76x	0.10x
DNW*	76	n/a	n/a	74	n/a	n/a

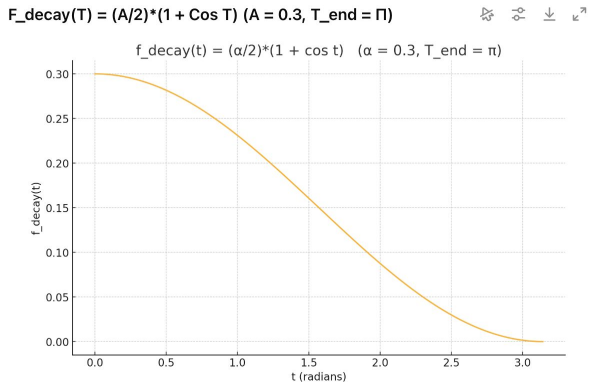
결과 : ERK가 Uniform에 비해 우수함

*

(2) Update Schedule & (3) Drop & (4) Grow

- RigL은 점점 줄어드는 cosine annealing 방식의 스캐줄 사용
- 무엇이 점점 줄어드나?
  - Update하는 파라미터의 개수
- Update? → 연결을 자르고, 연결을 다시 붙일 connection의 개수

magnitude gradient



(2) Update Schedule. The update schedule is defined by the following parameters: (1)ΔT: the number of iterations between sparse connectivity updates, (2) T_{end}: the iteration at which to stop updating the sparse connectivity, (3) α: the initial fraction of connections updated and (4) f_{decay}: a function, invoked every ΔT iterations until T_{end}, possibly decaying the fraction of updated connections over time. For the latter, as in Dettmers & Zettlemoyer (2019), we use *cosine* annealing, as we find it slightly outperforms the other methods considered.

Time = Step

update할 connection의 개수를 결정하는 함수

$$f_{decay}(t;\ \alpha,\ T_{end}) = \frac{\alpha}{2}\left(1 + cos\left(\frac{t\pi}{T_{end}}\right)\right)$$

cosine annealing : 점점 줄어드는 스캐줄

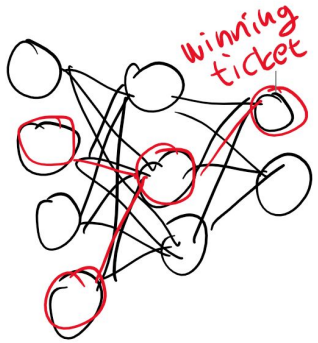
(3) Drop criterion. Every ΔT steps we drop the connections given by $ArgTopK(-|\theta^l|, f_{decay}(t;\ \alpha, T_{end})(1 - s^l)N^l)$, where $ArgTopK(v,\ k)$ gives the indices of the top-k elements of vector v.

(4) Grow criterion. The novelty of our method lies in how we grow new connections. We grow the connections with highest magnitude gradients, $ArgTopK_{i \notin \theta^l \setminus \mathbb{I}_{active}}(|\nabla_{\Theta^l} L_t|,\ k)$, where $\theta^l \setminus \mathbb{I}_{active}$ is the set of active connections remaining after step (3). Newly activated connections are **initialized to zero** and therefore

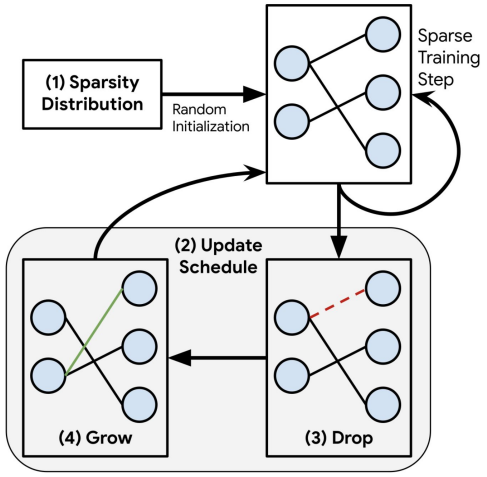
gradient :
다음 학습에서 많이 변함 → 다음 학습에서 큰 magnitude가 기대됨

1주차 논문과는 무슨 관계인가

- **LTH가 밝혀낸 것**
  - *가지치기를 한 상태에서 학습을 하기 위해 파라미터를 초기화 할 때,*
    가지치기를 하기 전의 초기값으로 초기화를 하면 학습이 성공적이다!
    (the initialization lottery!!)
  - *dense network 안에*
    *이미 **학습을 성공적으로 만드는 네트워크 토폴로지 + 초기값이 있다*** *는 가*



- ***LTH의 한계***
  - *LTH를 적용하려면 fully-connect 상태로 학습을 한 번 꼭 해야함*



Rigging the Lottery

Rigged : 조작된

<= 이미 sparse 한 상태로 시작한다

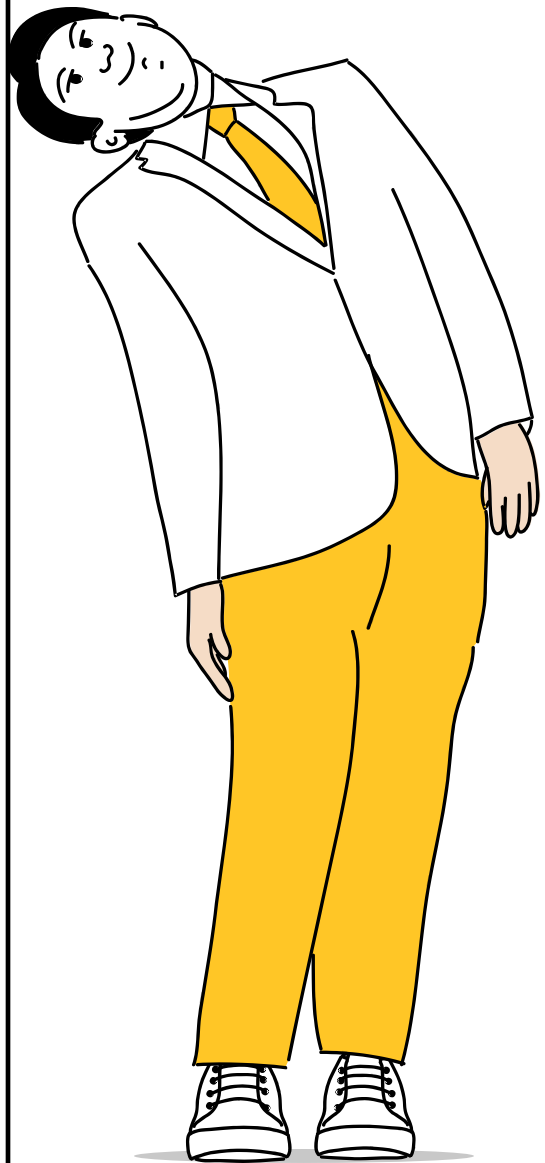
RigL은 '운 좋은' 초기화를 필요로 하지 않고
sparse network를 학습하는 방법을 제시함

the original initial conditions. In this paper we introduce a new method for training sparse models without the need of a "lucky" initialization; for this reason, we call our method "The Rigged Lottery" or *RigL*†. We make the following specific contributions:

*

감사합니다

2025. 03. 12 (Wed) 온디에 1주차 구승연



추가로,
https://github.com/google-research/lottery-ticket-hypothesis