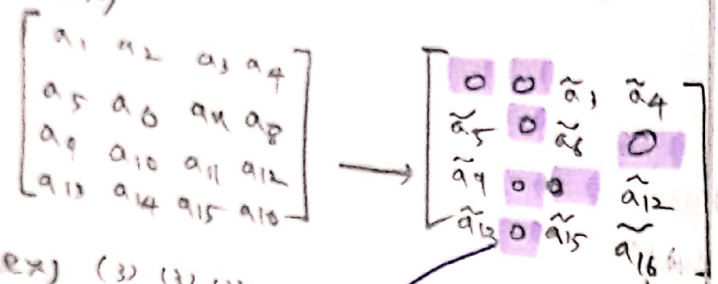
10/20

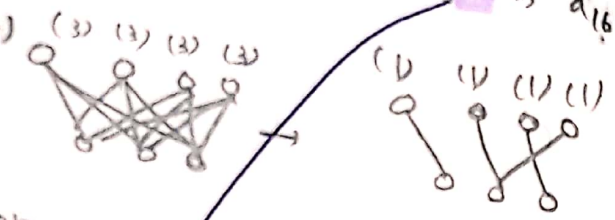
## ↳ pruning

<1>

ex)



ex)



「pruning = model compression technique」

Zero-ing out the weights.

$$\begin{bmatrix} a_1 & 0 \\ 0 & a_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 \cdot b_1, & a_1 \cdot b_2 \\ a_4 \cdot b_3, & a_4 \cdot b_4 \end{bmatrix}$$

( 4 multiplication
  0 addition.

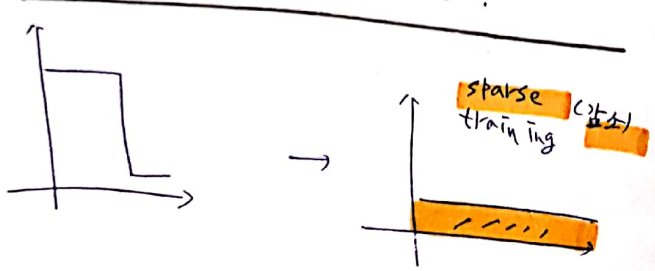
만일 pruning 안 했으면,

( 8 multiplication ex) 「$a_1 \odot b_1 + a_2 \odot b_3$」 ↦ 1 addition.
  4 addition. 2 multiplication

a)

★ How should we select the weight to prune? 「pruning 기준」

① Magnitude-based pruning
② Loss-based pruning
③ Training-based pruning




When should we prune the model, and by how much?

Early works - 'Loss-based pruning'

① generalization
② speed-up ③ interpretability
= Less compute → Faster speed.

less decision tree

$\min_{\tilde{\theta} \in \mathcal{N}_K(\theta)} L(\tilde{\theta})$ 「$L(\theta)$ : training loss,

↳ the set

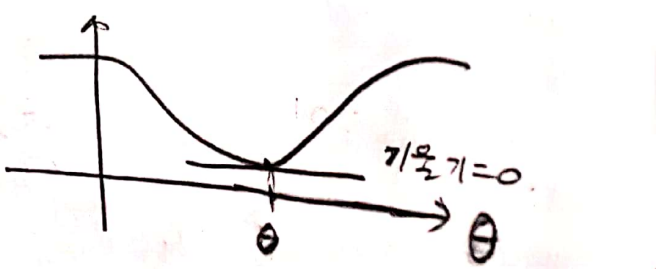
★ 「pruning weight = training risk is minimized right after removing them」 → 가중치 값 & 「중요하지 않은 값」.

$a \odot b = \begin{bmatrix} a_1 \cdot b_1 \\ a_2 \cdot b_2 \\ \vdots \\ a_n \cdot b_n \end{bmatrix}$, $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$

「$a \odot b$」

$$10^{12} C_{10^9} = \binom{10^{12}}{10^9} = \frac{10^{12}!}{10^9!\,(10^{12}-10^9)!} \leq (10^{12})^{10^9}$$

$$L(\tilde{\theta}) \simeq L(\theta) + (\tilde{\theta} - \theta)^T \cdot G_\theta + \frac{1}{2} (\tilde{\theta} - \theta)^T \cdot H_\theta (\tilde{\theta} - \theta).$$



Remove all weights, except for the top-k elements in terms of $h_{ii} \cdot \theta_i^2$.

「optimal brain surgeon」

= Hassibi & Stork (1998)

↳ know to perform better but requires much more compute.

↓

* approximate Hessian with a Fisher matrix of the model

---

Hessian

$\nabla f \rightarrow$ vector $(f_x, f_y)$

$f(x,y)$에 대한 second derivative (2번 미분한 것에 대한 설명은 생략)

$f''(x,y) = f_{xx}, f_{xy}, f_{yx}, f_{yy}$.

이것을 행렬로 나타낸 것 = Hessian

$$\begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$$

$D(a,b)$ = discriminant $= f_{xx} \cdot f_{yy} - f_{xy} \cdot f_{yx}$.

= a, b에서의 discriminant 값.

critical point 찾기 => 극값을 찾을 때, 주어진 범위 내에서 $f'(x)=0$이 되는 x값을 찾는다.

* $f'(x)=0$을 만족시키는 모든 x값이 극대 또는 극소의 x값이 아님.

↳ 항상 극대·극소인지를 확인하기 위해 이계도함수 $f''(x)$를 구하여 $f'(x)=0$을 만족시키는 x값의 전후의 부호를 확인하여 극솟값, 극댓값을 찾을 수 있다.

「이계도함수 = 여변수 함수에서의 Hessian matix = discriminant」.

## Training-based pruning

↳ rather, they modify the loss function, gradient update to encourage having many zeros.

Monte carlo approximation to update $\theta$, $\phi$ simultaneously.

gaussian) $\tilde{\theta} = M \odot \theta$, $M_i \sim N(1, \lambda)$

---

* Magnitude-based pruning.

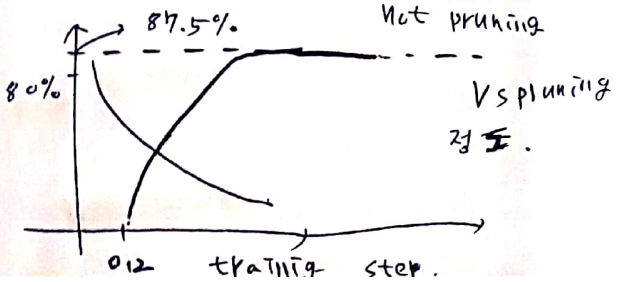
At the same time, easier to use

① no extra compute / memory

② no data

pruning scheduling

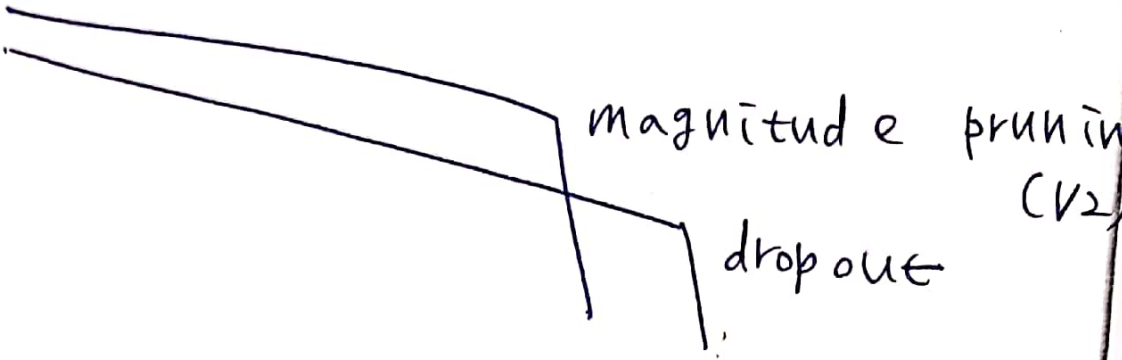
= training / retraining steps, fraction to remove at each step best to have gradual scheduling.



cubic scheduling.

pruning scheduling 같은 것 예시

= ??




☆

magnitude-based = ① easiest and cheapest to use. ② requries gradual pruning and retraining. ③ requires well-tuned hyperparmeters.

Loss-based = more compute/ memory-intensive.

training-based = very compute / memory-intensive.

↳ known to be unstable for large-scale tasks.