The Combinatorial Brain Surgeon:

Pruning Weights That Cancel One Another in Neural Networks




Xin Yu @ Univ. of Utah

Thiago Serra @ Bucknell Univ.

Srikumar Ramalingam @ Google Research

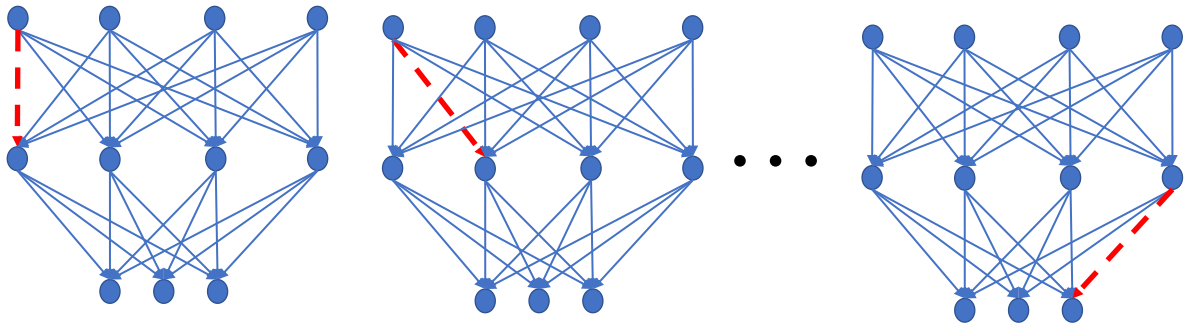
Shandian Zhe @ Univ. of Utah

Network Pruning / Sparsification

Given a well-trained neural network architecture, can we **carefully remove an excess of weights** which will produce a model with **similar or even improved accuracy**?

Impact-based methods:

Get impact of a single weight on the NN



Remove the weights with smallest impact

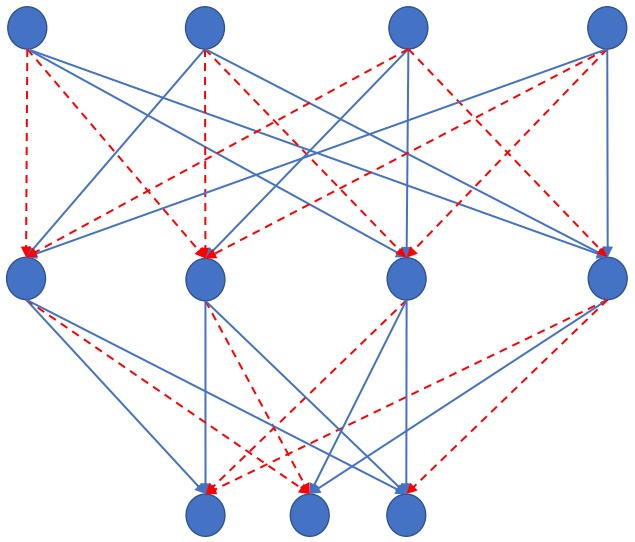
$$I_{s1} \geq I_{s2} \geq I_{s3} \geq \ldots \geq I_{sN}$$

$$I_{\{i,j\}} \neq I_i + I_j$$

Network Pruning / Sparsification

CBS(ours): study **the combined effect of removing multiple weights simultaneously** and obtain significantly better performance than removing each weight alone.

CBS:



Dependency between weights are more important!

Network Pruning / Sparsification

Optimal Brain Damage(LeCun's 1990) and Optimal Brain Surgeon(Hassibi 1992)

Both OBD & OBS use the **functional Taylor expansion of the loss function**:

$$L(w) - L(\bar{w})$$

$$= \cancel{(w - \bar{w})^T \nabla L(\bar{w})} + \frac{1}{2}(w - \bar{w})^T \nabla^2 L(\bar{w})(w - \bar{w}) + \cancel{O(||w - \bar{w}||^3)}$$

- Assume network is properly trained:

$$\nabla L(\bar{w}) = 0$$

- Assume w is sufficiently closed to $\bar{w}$:

$$O(||w - \bar{w}||^3) \approx 0$$

The Optimization Problem in OBS and WoodFisher

OBD: assumes that the Hessian $H := \nabla^2 L(\bar{w})$ is a diagonal matrix:

$$L(w) - L(\bar{w}) \approx \frac{1}{2}\sum_i (w_i - \bar{w}_i)^2 H_{i,i}$$

OBS: **Not consider interdependency!**

$$\min_{k\in[N]} \left\{ \min_{w\in\mathbb{R}^N} \left\{ \frac{1}{2}\sum_i\sum_j (w_i - \bar{w}_i)H_{i,j}(w_j - \bar{w}_j) : w_k = 0 \right\} \right\}$$

While choosing one weight to prune…

we adjust the remaining weights to locally minimize the loss function

Woodfisher(Singh2020): Removing two weights is **combinatorically explosive:**

$$\min_{k_1,k_2\in[N]} \left\{ \min_{w\in\mathbb{R}^N} \left\{ \frac{1}{2}\sum_i\sum_j (w_i - \bar{w}_i)H_{i,j}(w_j - \bar{w}_j) : w_{k_1} = 0, w_{k_2} = 0 \right\} \right\}$$

While choosing two weight to prune…

(Their work also provides helpful guidance on approximating H^{-1} well)

The Combinatorial Brain Surgeon: Just MIP It!?

We formulate a **Mixed Integer Quadratic Program** for deciding

(i) **which weights to prune** to achieve a **given sparsity rate r**; and

(ii) how to **adjust the unpruned weights**:

$$\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (w_i - \bar{w}_i)\ H_{i,j}\ (w_j - \bar{w}_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i = \lceil r\ N \rceil$$

$$y_i = 1 \rightarrow w_i = 0 \qquad \forall i \in [N]$$

$$y_i \in \{0,1\} \qquad \forall i \in [N]$$

$$w_i \in \mathbb{R} \qquad \forall i \in [N]$$

CBS Selection (CBS-S)

Since CBS is very challenging, we focus on the **selection of weights to prune**:

$$\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \bar{w}_i y_i \; H_{i,j} \; \bar{w}_j y_j$$

$$A_{i,j} = \bar{w}_i H_{i,j} \; \bar{w}_j$$

Each term captures the combined effect of pruning both w_i and w_j

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i = \lceil r \, N \rceil$$

$$y_i \in \{0,1\} \qquad \forall i \in [N]$$

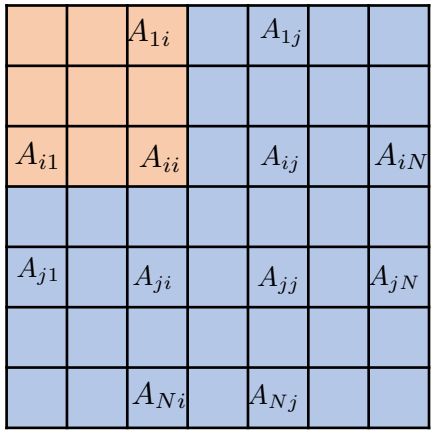
CBS-S alone is still a challenging problem, but one in which we could potentially **leverage the interdependency between pruned weights**

CBS-S: Greedy local search

Assuming we have a good initial pruning selection, we swap pruned weights and unpruned weights to optimize the CBS-S objective.

- Local Search only requires **linear times swaps** instead of quadratic!
- Each swap is computationally efficient:

  swapping a pruned weight $w_i, i \in \mathbb{P}$ with an unpruned weight $w_j, j \in \overline{\mathbb{P}}$



$$A_{i,j} = \overline{w}_i H_{i,j} \overline{w}_j$$

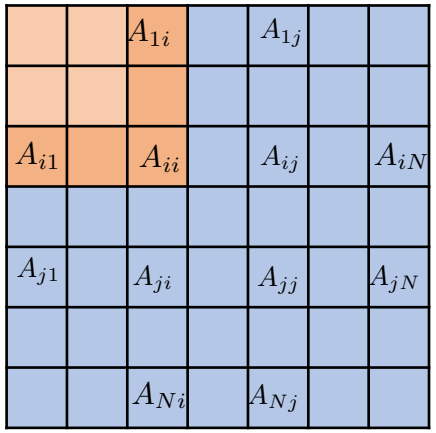
$A_{i,j}$ matters to the objective function only if both i and j are selected

CBS-S: Greedy local search

Assuming we have a good initial pruning selection, we swap pruned weights and unpruned weights to optimize the CBS-S objective.

- Local Search only requires **linear times swaps** instead of quadratic!
- Each swap is computationally efficient:

  swapping a pruned weight $w_i, i \in \mathbb{P}$ with an unpruned weight $w_j, j \in \overline{\mathbb{P}}$



$A_{i,j} = \overline{w}_i H_{i,j} \overline{w}_j$

Cost of keeping i in $\mathbb{P}$: $\boldsymbol{\alpha_i \propto A_{i,i} + \sum_{j \in \mathbb{P}, j \neq i} (A_{i,j} + A_{j,i})}$

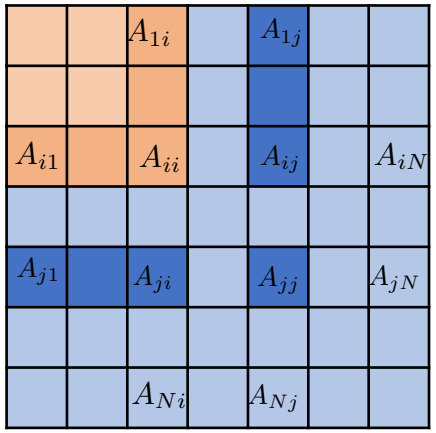
$A_{i,j}$ matters to the objective function only if both i and j are selected

CBS-S: Greedy local search

Assuming we have a good initial pruning selection, we swap pruned weights and unpruned weights to optimize the CBS-S objective.

- Local Search only requires **linear times swaps** instead of quadratic!
- Each swap is computationally efficient:

  swapping a pruned weight $w_i, i \in \mathbb{P}$ with an unpruned weight $w_j, j \in \overline{\mathbb{P}}$



$A_{i,j} = \overline{w}_i H_{i,j} \overline{w}_j$

Cost of keeping i in $\mathbb{P}$: $\boldsymbol{\alpha_i} \propto \boldsymbol{A_{i,i} + \sum_{j \in \mathbb{P}, j \neq i} (A_{i,j} + A_{j,i})}$

Cost of adding j to $\mathbb{P}$: $\boldsymbol{\beta_j} \propto \boldsymbol{A_{j,j} + \sum_{i \in \mathbb{P}} (A_{i,j} + A_{j,i})}$

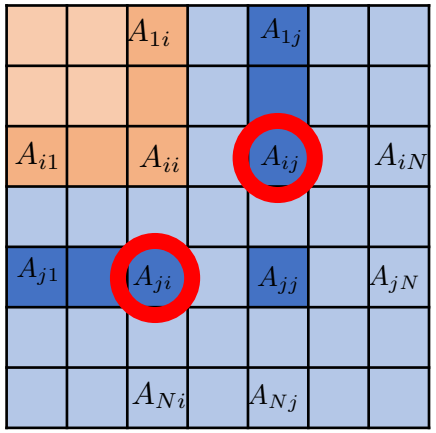
$A_{i,j}$ matters to the objective function only if both i and j are selected

CBS-S: Greedy local search

Assuming we have a good initial pruning selection, we swap pruned weights and unpruned weights to optimize the CBS-S objective.

- Local Search only requires **linear times swaps** instead of quadratic!
- Each swap is computationally efficient:

  swapping a pruned weight $w_i, i \in \mathbb{P}$ with an unpruned weight $w_j, j \in \overline{\mathbb{P}}$



$A_{i,j} = \overline{w}_i H_{i,j} \overline{w}_j$

Cost of keeping i in $\mathbb{P}$: $\boldsymbol{\alpha_i} \propto \boldsymbol{A_{i,i} + \sum_{j \in \mathbb{P}, j \neq i} (A_{i,j} + A_{j,i})}$

Cost of adding j to $\mathbb{P}$: $\boldsymbol{\beta_j} \propto \boldsymbol{A_{j,j} + \sum_{i \in \mathbb{P}} (A_{i,j} + A_{j,i})}$

Cost associated with keeping both: $\boldsymbol{\gamma_{i,j}} \propto \boldsymbol{A_{i,j} + A_{j,i}}$

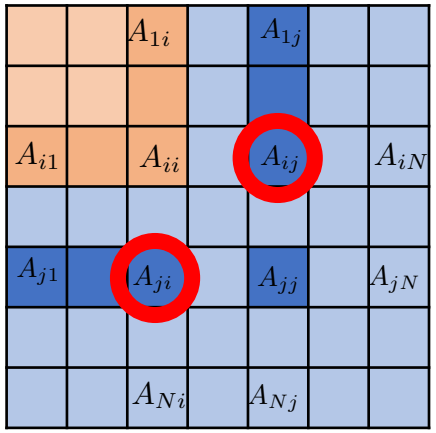
$A_{i,j}$ matters to the objective function only if both i and j are selected

CBS-S: Greedy local search

Assuming we have a good initial pruning selection, we swap pruned weights and unpruned weights to optimize the CBS-S objective.

- Local Search only requires **linear times swaps** instead of quadratic!
- Each swap is computationally efficient:

  swapping a pruned weight $w_i, i \in \mathbb{P}$ with an unpruned weight $w_j, j \in \overline{\mathbb{P}}$



$A_{i,j} = \overline{w}_i H_{i,j} \overline{w}_j$

Cost of keeping i in $\mathbb{P}$: $\boldsymbol{\alpha_i} \propto \boldsymbol{A_{i,i} + \sum_{j \in \mathbb{P}, j \neq i} (A_{i,j} + A_{j,i})}$

Cost of adding j to $\mathbb{P}$: $\boldsymbol{\beta_j} \propto \boldsymbol{A_{j,j} + \sum_{i \in \mathbb{P}} (A_{i,j} + A_{j,i})}$

Cost associated with keeping both: $\boldsymbol{\gamma_{i,j}} \propto \boldsymbol{A_{i,j} + A_{j,i}}$

Cost of swapping i with j: $\propto \boldsymbol{-\alpha_i + \beta_j - \gamma_{i,j}}$

$A_{i,j}$ matters to the objective function only if both i and j are selected

CBS Update (CBS-U)

Given a **CBS-S solution $\tilde{y}$**, the **systematic weight update** remains simple:

$$\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (w_i - \overline{w}_i)\ H_{i,j}\ (w_j - \overline{w}_j)$$

$$\text{s.t.} \quad w_i = 0 \qquad \forall i \in [N] : \tilde{y}_i = 1$$

$$w_i \in \mathbb{R} \qquad \forall i \in [N] : \tilde{y}_i = 0$$

By abstracting pruned weights altogether, CBS-U becomes an unconstrained quadratic optimization problem with closed form solution (provided H^{-1}) ☺

In OBS, we update the unpruned weights for every weight that is pruned; in CBS, we update the unpruned weights once based on all pruned weights

Computational Experiments

We compare accuracy for different sparsity rates in varying models / datasets:

- MP: Prune smallest weights
- WF-S / WF: Singh & Alistarh
- CBS-S / CBS: Ours

(Best accuracy)
(Second best)

Just prune weights

Prune + update

	Prune Selection				Weight Update		
Sparsity	*MP*	*WF-S*	*CBS-S*	**Improvement**	*WF*	*CBS*	**Improvement**
0.50	93.93	93.92	93.91	-0.02	**94.02**	93.96	-0.06
0.70	93.62	93.48	93.75	0.13	93.77	**93.98**	0.21
0.90	90.30	90.77	92.37	1.60	91.69	**93.14**	1.45
0.95	83.64	83.16	88.24	4.60	85.54	**88.92**	3.38
0.98	32.25	34.55	**66.64**	32.09	38.26	55.45	17.20

MNIST @ MLPNet

Thank you!

See you in the poster session!

xin.yu@utah.edu