Neural Network Pruning

(literature review)

이종진

Seoul National University

ga0408@snu.ac.kr

September 07, 2020

Table of Contents

1. Learning Both Weights and Connections for Effiecient Neural Networks. (2015,NIPS)

2. Pruning Filters for Efficient Convnets (2017, ICLR)

3. The Lottery Ticket Hypothesis Finding Sparse, Trainable Neural Network.(2018,ICML)

4. Optimal Brain Damage (1990, NIPS)

5. Second Order Derivatives for Network Pruning, Optimal Brain Surgeon (1993, NIPS)

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

7. SNIP, Single-shot Pruning Based on Connection Sensitivity. (2018, ICML)

8. Picking Winning Tickets Before Training by Preserving Gradient Flow. (2020, ICML)

9. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. (2020, arXiv)

Sparse

- Neural network pruning
- By removing unimportant weights from a network, several improvements can be expected.
  - Better generalization error.
  - Reduce hardware or storage requirements without affecting their accuracy.

Neural Network Pruning

- Pruning strategies
- Pre-train(Growing) / Prune / Fine-tune.
  1. Train a large, over-parameterized model.
     - Pre-trained is required/not required.
  2. Calculate the importance measure of each parameters, and prune each parameters based on the calculated measure.
     - Importance measures(magnitudes/Increment of loss,)
     - Structured pruning/Unstructured pruning.
  3. Fine-tune the pruned model.
     - Retain/Reinitialize/Rewind
  4. Repeat 1$\sim$3.
     - Single-shot/Iterative.

Notation

- Parameters: $\boldsymbol{\theta}$
- Initial value: $\boldsymbol{\theta}^0$, pre-trained: $\boldsymbol{\theta}^*$
- q-th parameters: θ_q, parameters with 0 in q-th component: $\boldsymbol{\theta}_{-q}$.
- Neural network $f(x; \boldsymbol{\theta})$
- $\boldsymbol{m}$ indicates masking variables which consists with 0 or 1.
- Pruned network $f(x; \boldsymbol{m} \odot \boldsymbol{\theta})$
- Cross-Entropy: $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$
- Hessian (Fisher information matrix)

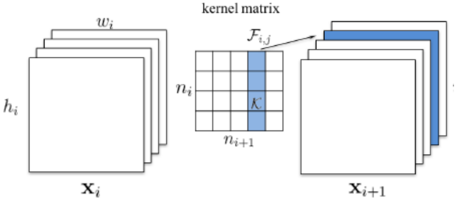
  $\mathbf{H} = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log p(y|x; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(y|x; \boldsymbol{\theta})^\top]$

1. Learning Both Weights and Connections for Effiecient Neural Networks.(2015,NIPS)

- Pre-trained/magnitude/unstructured/retain/iterative
- Method:
  1. Pre-train a given architecture.
  2. Prune the connections with their magnitude.($|\theta^*|$)
     (Threshold: $C \times$ standard deviation of the layer's weights)
  3. Fine-tuning with retaining pre-trained values.
     - $f(x; \boldsymbol{m} \odot \boldsymbol{\theta}^*)$
  4. Iteratively implemented.

2. Pruning Filters for Efficient Convnets (2017, ICLR)

- Pre-trained/magnitude/structured/retain/iterative
- Pruning filters in Convolution layer.
- n_i: the number of channels for the ith convolutional layer
- h_i/w_i be the height/widt of the input feature maps.
- $\mathcal{F}_{i,j} \in \mathbb{R}^{n_i \times k \times k}$, $\mathcal{F}_i \in \mathbb{R}^{n_{i+1} \times n_i \times k \times k}$
- When a filter $\mathcal{F}_{i,j}$ is pruned, its corresponding feature map $\boldsymbol{x}_{i+1,j}$ is removed. $n_i k^2 h_{i+1} w_{i+1}$



2. Pruning Filters for Efficient Convnets (2017, ICLR)

- The importance of a filter in each layer is calculating by the sum of its absolute weights.
- We prune same ratio for all layers in the same stage.
- Pruning filter results in pruning the corresponding activation.
  - Independent pruning
  - Greedy pruning(It does not consider the kernels for the previously pruned feature maps while calculating the importance)
- Retraining
  - Prune once and retrain
  - Prune and retrain iteratively
- We prune same ratio for all layers in the same stage.

2. Pruning Filters for Efficient Convnets (2017, ICLR)

- The procedure of pruning m filters from the ith convolutional layer is as follows:
  1. Pre-train a given architecture.
  2. Calculate the importance of jth filter $(\mathcal{F}_{i,j})$ $s_j = \sum_{l=1}^{n_i} \sum |\mathcal{K}_l|$.
  3. Prune m filters with the smallest values and their corresponding feature maps.

3. The Lottery Ticket Hypothesis Finding Sparse, Trainable Neural Network. (2018,ICML)

- Pre-trained/magnitude/unstructured/retain/iterative
- Method:
  1. Pre-train the given architecture
  2. Sort the connections($\boldsymbol{\theta}$) by its magnitudes within layer.
  3. Prune the connections with their magnitude.
     (with certain ratio p)
  4. Fine-tuning by rewinding original initial values.
     - $f(x; \boldsymbol{m} \odot \boldsymbol{\theta_0})$
  5. Iteratively implemented.

3. The Lottery Ticket Hypothesis Finding Sparse, Trainable Neural Network. (2018,ICML)

- There exists the sub-networks with favorable training properties within larger over-parameterized models.
  - Winning tickets $f(x; m \odot \theta_0)$
  - $\|m\|_0 \ll |\theta|$(fewer parameters) commensurate training time and accuracy.
- To find this lucky sub-network, they iteratively prune the lowest magnitude unpruned connections.
- Pruning with rewinding is much better than reinitialization
- Many papers struggle to figure it out.
  - Reinitializing with preserving sign of parameters shows comparable performance
  - Prune weight based on their change in magnitude between initialization and the end of training.
  - Prune weight globally in larger model.

4. Optimal Brain Damage (1990, NIPS)

- Pre-trained/Hessian based method/unstructured/retain/iterative
- The importance of the parameter is measured by the effect on the training loss by enforcing that parameter to be zero.
- From a pre-trained $\boldsymbol{\theta}^*$,

$$\mathbf{I}(\theta_q^*) = \mathcal{L}(\boldsymbol{\theta}^*) - \mathcal{L}(\boldsymbol{\theta}_{-q}^*)$$

- Appoximate the importance(saliency) of θ_q^* by second derivative approiximation.

4. Optimal Brain Damage (1990, NIPS)

- Local maximum $\boldsymbol{\theta}^*$
- Approximate the objective function at $\boldsymbol{\theta}^*$ (Hessian based methods)
- $\Delta\mathcal{L} = \underbrace{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}^\top \Delta\boldsymbol{\theta}}_{\approx 0} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top \mathbf{H} \Delta\boldsymbol{\theta} + \mathcal{O}\left(\|\Delta\boldsymbol{\theta}\|^3\right)$
- Then, $\mathbf{I}(\theta_q^*)$ can be approximated by $\frac{1}{2}{\theta_q^*}^2\mathbf{H}_{qq}$
- Diagonal assumption on $\mathbf{H}$(Independence assumption.)

4. Optimal Brain Damage (1990, NIPS)

1. Pre-train the given architecture to estimate local maximum $\boldsymbol{\theta}^*$
2. Compute the second derivatives H_{qq} for each parameters
3. Compute the saliences for each parameters: $s_k = \frac{1}{2} H_{qq} {\theta_q^*}^2$
4. Sort the parameters by saliency and prune the paramters
5. Iteratively implement

5. Second Order Derivatives for Network Pruning, Optimal Brain Surgeon (1993, NIPS)

- Pre-trained/Hessian based method/unstructured/retain/single-shot
- No restrictive assumptions on the Hessian.(Dependence assumption)
- Does not require retraining after the pruning of a weight.
- $\Delta\boldsymbol{\theta} \neq \boldsymbol{\theta}^* - \boldsymbol{\theta}^*_{-q}$
- $e_q^\top \Delta\boldsymbol{\theta} + \theta_q = 0$

5. Second Order Derivatives for Network Pruning, Optimal Brain Surgeon (1993, NIPS)

- The importance(saliency) of each weight is calculated by solving the following constrained optimization problem
- $e_q^\top \Delta\theta + \theta_q = 0$

$$\min_{\Delta\boldsymbol{\theta}} \frac{1}{2}\Delta\boldsymbol{\theta}^\top \mathbf{H}\Delta\boldsymbol{\theta} \text{ s.t. } \mathbf{e}_q^\top \Delta\boldsymbol{\theta} + \boldsymbol{\theta}_q^* = 0$$

- Lagrangian form

$$L = \frac{1}{2}\Delta\boldsymbol{\theta}^\top \mathbf{H}\Delta\boldsymbol{\theta} + \lambda(e_q^\top \Delta\theta + \theta_q)$$

5. Second Order Derivatives for Network Pruning, Optimal Brain Surgeon (1993, NIPS)

- By taking functional derivatives,

$$\Delta\boldsymbol{\theta} = -\frac{\boldsymbol{\theta}_q^*}{[\mathbf{H}^{-1}]_{qq}}\mathbf{H}^{-1}\mathbf{e}_q \text{ and } \Delta\mathcal{L}_q = \frac{1}{2}\frac{\left(\boldsymbol{\theta}_q^*\right)^2}{[\mathbf{H}^{-1}]_{qq}}$$

5. Second Order Derivatives for Network Pruning, Optimal Brain Surgeon (1993, NIPS)

- $H \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial f_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^{\top} \frac{\partial^2 \mathcal{L}(y_i, f_i(\boldsymbol{\theta}))}{\partial f_i^2} \frac{\partial f_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
- Denote $\boldsymbol{X}_i = \frac{\partial f_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, $\boldsymbol{A}_i = \frac{\partial^2 \mathcal{L}(y_i, f_i(\boldsymbol{\theta}))}{\partial f_i^2}$
- with $\boldsymbol{H}_0 = \alpha \boldsymbol{I}$ and $\boldsymbol{H}_n = \boldsymbol{H}$ and $10^{-8} \le \alpha \le 10^{-4}$

$$\boldsymbol{H}_{i+1} := \boldsymbol{H}_i + \frac{1}{n}\boldsymbol{X}_{i+1}^{\top}\boldsymbol{A}_i\boldsymbol{X}_{i+1} \quad i = 0, \ldots, n-1$$

- Recursively,

$$\boldsymbol{H}_{i+1}^{-1} = \boldsymbol{H}_i^{-1} - \boldsymbol{H}_i^{-1}\boldsymbol{X}_{i+1}^{\top}\left(n\boldsymbol{A}_i + \boldsymbol{X}_{i+1}\boldsymbol{H}_i^{-1}\boldsymbol{X}_i^{\top}\right)^{-1}\boldsymbol{X}_{i+1}\boldsymbol{H}_i^{-1}$$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Pre-trained/Hessian based method/structured/retain/iterative.
- High computational cost of computing second derivatives.
- Kronecker-factored approximate culvature.(K-FAC)
- Structured prunning
- Present two methods
  - Kron-OBD, Kron-OBS.
  - EigenDamage.

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Kronecker-Factored Approximation Curvature.
- Kronecker product($mp \times nq$) of $\boldsymbol{M}(m \times n)$, $\boldsymbol{N}(p \times q)$

$$\boldsymbol{M} \otimes \boldsymbol{N} = \begin{pmatrix} \boldsymbol{M}_{11}\boldsymbol{N} & \cdots & \boldsymbol{M}_{1n}\boldsymbol{N} \\ \vdots & & \vdots \\ \boldsymbol{M}_{m1}\boldsymbol{N} & \cdots & \boldsymbol{M}_{mn}\boldsymbol{N} \end{pmatrix}$$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Layerwise independence.
- Input activations $a \in \mathbb{R}^n$, weight matrix $W \in \mathbb{R}^{n \times m}$, output $\mathbb{R}^m (s = W^T a)$
- $\nabla_{\boldsymbol{W}} \mathcal{L} = \boldsymbol{a}(\nabla_{\boldsymbol{s}} \mathcal{L})^\top$
- K-FAC decomposes this layer's Fisher matrix.

$$\begin{aligned}
\boldsymbol{H} &= \mathbb{E}[\text{vec}\{\nabla_W \mathcal{L}\}\text{vec}\{\nabla_W \mathcal{L}\}^T] \\
&= \mathbb{E}[\{\nabla_s \mathcal{L}\}\{\nabla_s \mathcal{L}\}^T \otimes aa^T] \\
&\approx \mathbb{E}[\{\nabla_s \mathcal{L}\}\{\nabla_s \mathcal{L}\}^T] \otimes \mathbb{E}[aa^T] = S \otimes A
\end{aligned}$$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Efficient computation.($\mathbf{Q}$ and $\mathbf{\Lambda}$)

$$\mathbf{H} \cdot X = \mathbf{S} \otimes \mathbf{A} \cdot \mathbf{X} = \mathbf{A}\mathbf{X}\mathbf{S}^\top$$

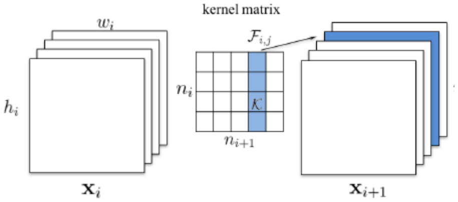
$$\mathbf{H}^{-1} = (\mathbf{S} \otimes \mathbf{A})^{-1} = \mathbf{S}^{-1} \otimes \mathbf{A}^{-1}$$

$$\mathbf{H} = (\mathbf{Q_S} \otimes \mathbf{Q_A})(\mathbf{\Lambda}_S \otimes \mathbf{\Lambda}_A)(\mathbf{Q}_S \otimes \mathbf{Q_A})^\top$$

- $\mathbf{Q_S} \otimes \mathbf{Q_A}$ gives the eigen basis of the Kronecker product.

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- For ith convolution layer,
- n_i: number of channels, h_i: height size, w_i: weight size, k: filter size.
- $\boldsymbol{x}_i \in \mathbb{R}^{n_i \times h_i \times w_i}$, $\boldsymbol{x}_{i+1} \in \mathbb{R}^{n_{i+1} \times h_{i+1} \times w_{i+1}}$, $\mathcal{F}_{i,j} \in \mathbb{R}^{n_i \times k^2}$, $\mathcal{F}_i \in \mathbb{R}^{n_i \times n_{i+1} \times k^2}$



6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- $\mathcal{I}_{i+1} = [h_{i+1}] \times [w_{i+1}]$, spatial location index for output channel.
- $\ell \in \mathcal{I}_{i+1}$, $\boldsymbol{x}_{i+1}^{(\ell)} \in \mathbb{R}^{n_{i+1}}$
- Like in the fully connected layer case, $\Delta_{\boldsymbol{W}} \mathcal{L} = \boldsymbol{a}(\Delta_{\boldsymbol{s}} \mathcal{L})^{\top}$
  the gradient of the kernel matrix $\mathcal{F}_i = \sum_{\ell \in \mathcal{I}_{i+1}} x_i^{p(\ell)} \nabla_{x_{i+1}^{(\ell)}} \mathcal{L}^{\top}$
- $\boldsymbol{x}_i^{p(\ell)} \in \mathbb{R}^{n_i k^2}$ is corresponding patch for l-th spatial location.
- $\nabla_{\mathcal{F}_{i,j}} \mathcal{L} = \sum_{l \in \mathcal{I}_{i+1}} \boldsymbol{x}_i^{p(\ell)} \nabla_{\boldsymbol{x}_{i+1}^{(\ell)}} \mathcal{L}^{T}, j = 1, \ldots, n_{i+1}$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Then, Hessian matrix can be computed.

$$
\begin{aligned}
\boldsymbol{H}_i &\approx \sum_{l \in \mathcal{I}_{i+1}} \mathbb{E}[\{\nabla_{x_{i+1}^{(\ell)}} \mathcal{L}\}\{\nabla_{x_{i+1}^{(\ell)}} \mathcal{L}\}^T] \otimes \mathbb{E}[x_i^{p(\ell)} x_i^{p(\ell)\,T}] \\
&\approx \left(\frac{1}{|\mathcal{I}_{i+1}|} \sum \mathbb{E}[\{\nabla_{x_{i+1}^{(\ell)}} \mathcal{L}\}\{\nabla_{x_{i+1}^{(\ell)}} \mathcal{L}\}^T]\right) \otimes \left(\sum_{\ell \in \mathcal{I}_{i+1}} \mathbb{E}[x_i^{p(\ell)} x_i^{p(\ell)\,T}]\right) \\
&:= \boldsymbol{S} \otimes \boldsymbol{A}
\end{aligned}
$$

where $\boldsymbol{S}$ is $n_{i+1} \times n_{i+1}$ and $\boldsymbol{A}$ is $n_i k^2 \times n_i k^2$ matrix.

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Without considering the interaction between filters, we can compute the importance of each filter in ith convolution layer

$$\Delta \mathcal{L}_j = \frac{1}{2} \mathcal{F}_{ij}^{*\top} H_{i,j} \mathcal{F}_{ij}^{*}$$

where $\mathcal{F}_{ij}^{*} \in \mathbb{R}^{n_i k^2}$ and $H_{i,j} \in \mathbb{R}^{n_i k_i^2 k^2}$

- (Convolution) layerwise independence.
- Kron-OBD

$$\Delta \mathcal{F}_{ij} = -\mathcal{F}_{ij}^{*} \text{ and } \Delta \mathcal{L}_j = \frac{1}{2} \boldsymbol{S}_{jj} \mathcal{F}_{ij}^{*\top} \boldsymbol{A} \mathcal{F}_{ij}^{*}$$

- Kron-OBS

$$\Delta \mathcal{F}_{ij} = -\frac{\boldsymbol{S}^{-1} \boldsymbol{e}_j \otimes \mathcal{F}_{ij}^{*}}{[\boldsymbol{S}^{-1}]_{jj}} \text{ and } \Delta \mathcal{L}_j = \frac{1}{2} \frac{\mathcal{F}_i^{*\top} \boldsymbol{A} \mathcal{F}_{ij}^{*}}{[\boldsymbol{S}^{-1}]_{jj}}$$

where $\boldsymbol{e}_j$ is the selecting vector with 1 for elements of $\mathcal{F}_{i,j}$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

1. Compute Kronecker factors $\boldsymbol{A}$ and $\boldsymbol{S}$
2. Compute the importance for each filters in each layers.
3. Compute p_{th} percentile of $\Delta\mathcal{L}$ as τ
4. Update $\boldsymbol{\theta}$ as $\Delta\boldsymbol{\theta}$(Kron-OBD or Kron-OBS)
5. Finetune the network.

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

- Eigendamage based on $\mathbf{Q_S} \otimes \mathbf{Q_A}$ the eigen basis of the Kronecker product.
- Introduce a novel network reparameterization.

$$\mathcal{F}_i = (\boldsymbol{Q_S} \otimes \boldsymbol{Q_A})\mathcal{F}_i^{'} = \boldsymbol{Q_A}\mathcal{F}_i^{'}\boldsymbol{Q_S}^{\top}, \quad \mathcal{F}_i^{'} = (\boldsymbol{Q_S} \otimes \boldsymbol{Q_A})^{\top}\mathcal{F}_i$$

6. EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis (2019, ICML)

1. Compute Kronecker factors $\boldsymbol{A}$ and $\boldsymbol{S}$
2. $Q_S, \boldsymbol{\Lambda}_S = \text{Eigen}(S)$ and $Q_A, \boldsymbol{\Lambda}_A = \text{Eigen}(A)$
3. $\boldsymbol{\Theta} = \mathcal{F}_i^{'} \cdot \text{diag}(\boldsymbol{\Lambda}_A)\,\text{diag}(\boldsymbol{\Lambda}_S) \cdot \mathcal{F}_i^{'}$
4. for all row (or column) in $\boldsymbol{\Theta}$

$$\Delta\mathcal{L}_r = \boldsymbol{\theta}_{r,.}1 \text{ or } (\Delta\mathcal{L}_c = \mathbf{1}^\top \boldsymbol{\theta}_{.,c})$$

5. Remove r_{th} row(or c_{th} column) in $\boldsymbol{W}^{'}$ and r_{th}(or c_{th}) eigenbasis in $\boldsymbol{Q}_A$(or $\boldsymbol{Q}_S$ if $\Delta\mathcal{L}_r$(or $\Delta\mathcal{L}_c \leq \tau$).
6. Compute p_{th} percentile of $\Delta\mathcal{L}$ as τ
7. Finetune the network.

7. SNIP, Single-shot Pruning Based on Connection Sensitivity. (2018, ICML)

- ~~*Pre − trained*~~/Connection sensitivity/unstructured/retain
- Prunes a given network once at initialization prior to training.
- After pruning, the sparse network is trained in the standard way.
- Determines important connections with a mini-batch of data at single shot.
- The importance(Connection sensitivity) of each parameters is calculated as follow.

$$S(\theta_q) = \lim_{\epsilon \to 0} \left| \frac{\mathcal{L}(\theta_0) - \mathcal{L}(\theta_0 + \epsilon \delta_q)}{\epsilon} \right| = \left| \theta_q \frac{\partial \mathcal{L}}{\partial \theta_q} \right|$$

7. SNIP, Single-shot Pruning Based on Connection Sensitivity. (2018, ICML)

1. Variance scaling initialization
2. Using a mini-batch, Compute the connection sensitivity.

$$S(\theta_q) = \lim_{\epsilon \to 0} \left| \frac{\mathcal{L}(\theta_0) - \mathcal{L}(\theta_0 + \epsilon \delta_q)}{\epsilon} \right| = \left| \theta_q \frac{\partial \mathcal{L}}{\partial \theta_q} \right|$$

3. Sort the parameters by their connection sensitivity and prune the parameters
4. Training the pruned model.

8. Picking Winning Tickets Before Training by Preserving Gradient Flow. (2020, ICML)

- GraSP
- ~~*Pre − trained*~~/Preserving Gradient Flow/unstructured/retain
- It is important to preserve the training dynamics than the loss itself.
- A larger gradient norm indicates that, each gradient update achieves a greater loss reduction,

$$\Delta\mathcal{L}(\theta) = \lim_{\epsilon \to 0} \frac{\mathcal{L}(\theta + \epsilon\nabla\mathcal{L}(\theta)) - \mathcal{L}(\theta)}{\epsilon} = \nabla\mathcal{L}(\theta)^{\top}\nabla\mathcal{L}(\theta)$$

8. Picking Winning Tickets Before Training by Preserving Gradient Flow. (2020, ICML)

- Prune the weights whose removal will not reduce the gradient flow.

$$\mathbf{S}(\delta) = \Delta\mathcal{L}\left(\theta_0 + \delta\right) - \underbrace{\Delta\mathcal{L}\left(\boldsymbol{\theta}_0\right)}_{\text{Const}} = 2\delta^\top \nabla^2 \mathcal{L}\left(\theta_0\right) \nabla \mathcal{L}\left(\theta_0\right) + \mathcal{O}\left(\|\delta\|_2^2\right)$$

$$= 2\delta^\top \mathbf{H}\mathbf{g} + \mathcal{O}\left(\|\delta\|_2^2\right)$$

- The importance of weights,

$$\mathbf{S}(-\boldsymbol{\theta}) = -\boldsymbol{\theta} \odot \mathbf{H}\mathbf{g}$$

  - The larger the score of a weight the lower its importance.
  - $\mathbf{H} = \mathbf{I}$, equals to SNIP.

8 .Picking Winning Tickets Before Training by Preserving Gradient Flow. (2020, ICML)

1. $\mathcal{D}_b = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{b} \sim \mathcal{D}$
2. Compute the Hessian-gradient product Hg
3. $\mathrm{S}(-\theta_0) = -\theta_0 \odot \mathrm{Hg}$
4. Compute p_{th} percentile of $\mathrm{S}(-\theta_0)$ as τ
5. $\mathbf{m} = \mathbf{S}(-\boldsymbol{\theta}_0) < \tau$
6. Train the network $f_{\mathrm{m}\odot\theta}$ on $\mathcal{D}$ until convergence.

9. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. (2020, arXiv)

- Faithful connection sensitivity through layerwise dynamical isometry.
- Layerwise Jacobian

$$\mathbf{J}^{l-1,l} = \frac{\partial \mathbf{x}^l}{\partial \mathbf{x}^{l-1}} = \mathbf{D}^l \mathbf{W}^l$$

- Layerwise dynamical isometry is that Layerwise Jacobian's singular values are exactly 1.
- Reinitialize the weights after pruining by

$$\min_{\mathbf{W}^l} \left\| \left(\mathbf{m}^l \odot \mathbf{W}^l\right)^T \left(\mathbf{m}^l \odot \mathbf{W}^l\right) - \mathbf{I}^l \right\|_F$$

The end

The end.