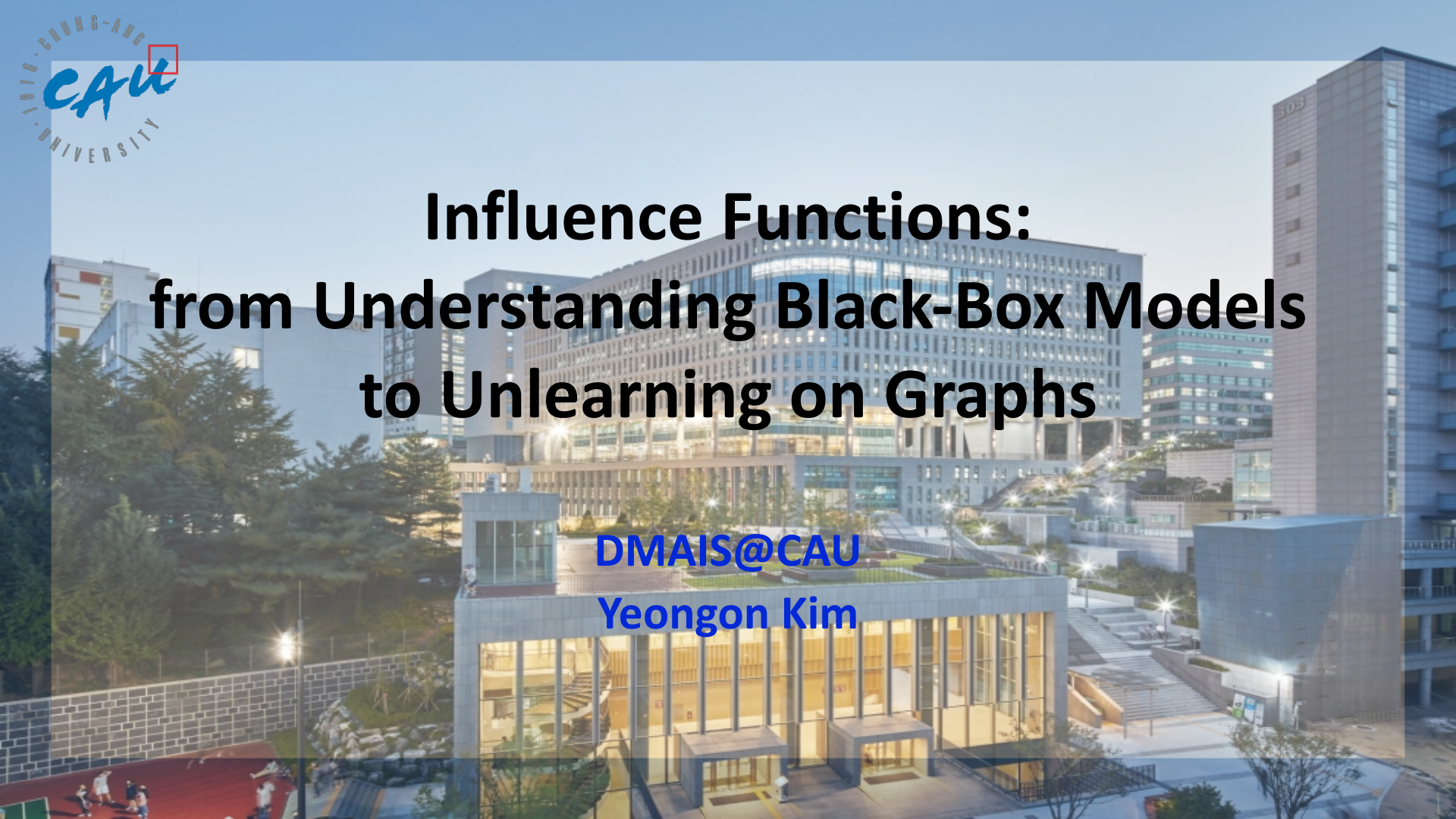


Influence Functions: from Understanding Black-Box Models to Unlearning on Graphs

DMAIS@CAU
Yeongon Kim

Papers

Understanding Black-box Predictions via Influence Functions

Pang Wei Koh [1] **Percy Liang** [1]

GIF: A General Graph Unlearning Strategy via Influence Function

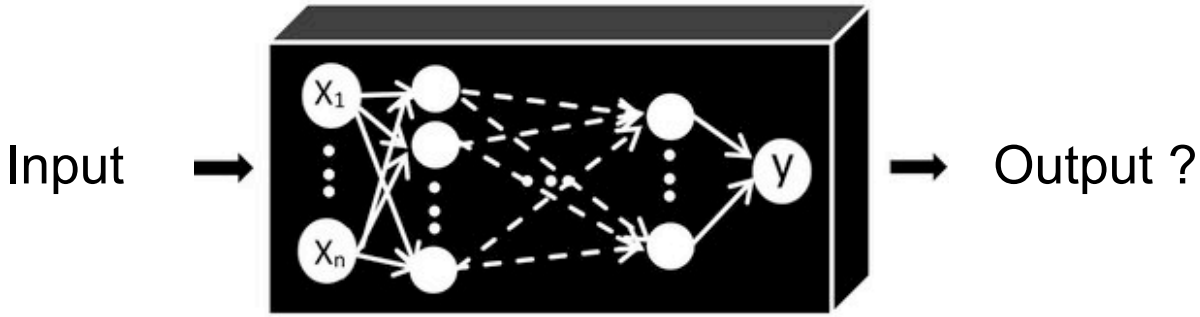
Jiancan Wu[1*], Yi Yang[1*], Yuchun Qian[1], Yongduo Sui[1], Xiang Wang[1†], Xiangnan He[1†]
[1]University of Science and Technology of China, China,
wujcan@gmail.com,{yanggnay,daytoy_qyc,syd2019}@mail.ustc.edu.cn,{xiangwang1223,xiangnanhe}@gmail.com

INDEX

Introduction

❑ Black-Box of Modern Models

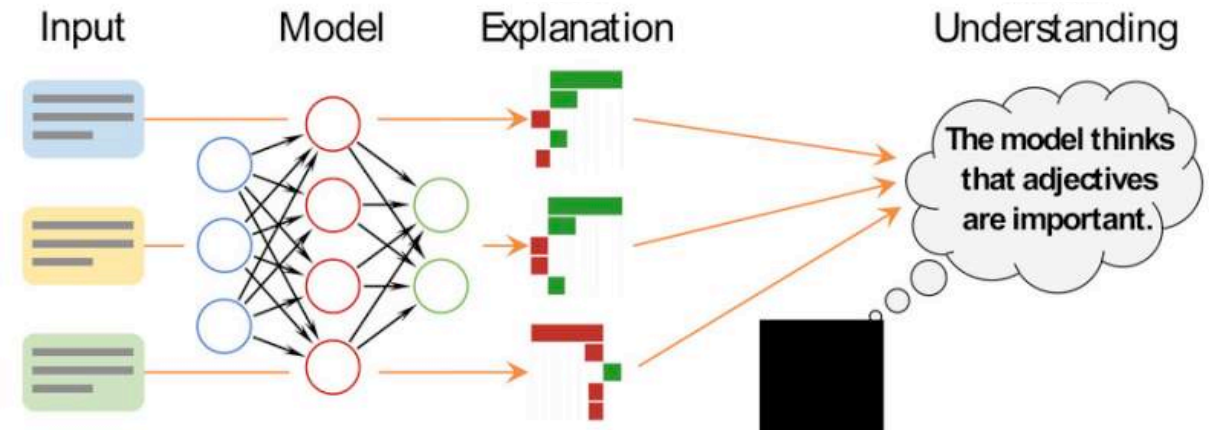
- Modern machine learning models show much higher performance than models before
- Complexity seen in modern models leads to black-box outputs
- Black-box makes it difficult to improve or maintain the models
- **Importance of Explainability is growing**



Introduction

❑ Interpreting Black-Box Models

- Focused on post-hoc explanation
- **Cannot explain where the model came from**

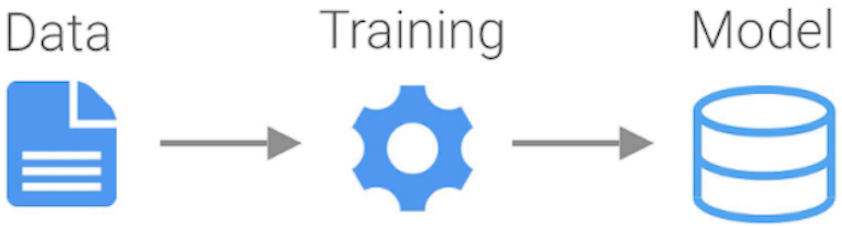


Introduction

❑ Models Derive from Training Data

- Parameters of a model are rooted in its training data
- Analyzing via data perturbation and repeated retraining can be costly

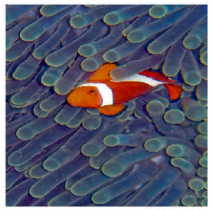
➡ **Utilize influence functions for a practical approach**



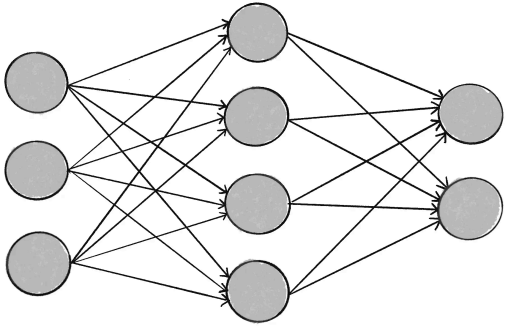
Influence Functions: Methodology

- **“What would happen if the training point were changed slightly?”**
  - Measure the influence of a training point on a model by influence function
  - Compute the **rate of change of the parameters** when a **training point z is upweighted** by some small ε
  - Utilizing small perturbations for an efficient, calculus-based approximation of influence

Training Data



ϵ



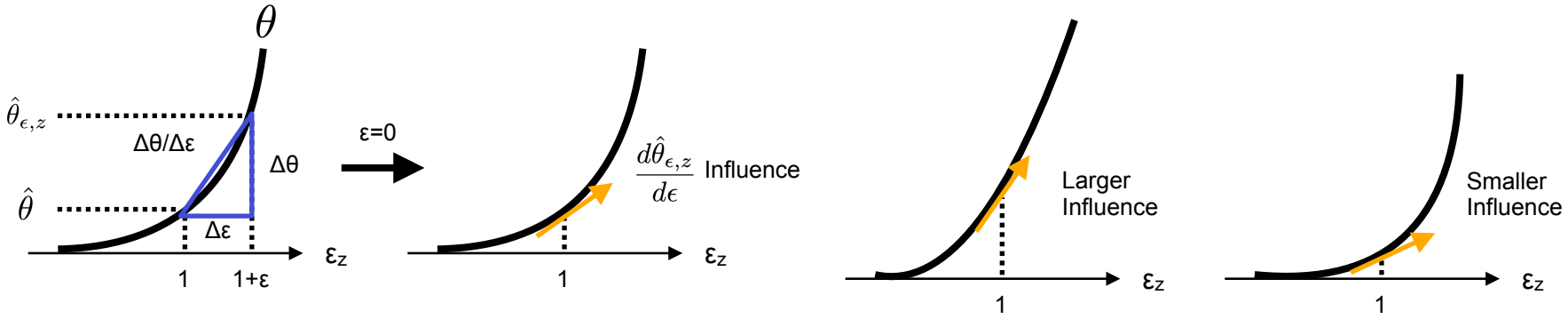
Influence

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$$

$$\hat{\theta} \longrightarrow \hat{\theta}_{\epsilon,z}$$

Influence Functions: Methodology

- **"What would happen if the training point were changed slightly?"**
  - Measure the influence of a training point on a model by influence function
  - Compute the **rate of change of the parameters** when a **training point z is upweighted** by some small ε
  - Utilizing small perturbations for an efficient, calculus-based approximation of influence



Influence Functions: Methodology

❑ Derivation of the Influence Function

Loss term Upweighting a training point

$$\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z,\theta)$$

Derivative

$$\nabla_\theta\left(\frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \varepsilon L(z,\theta)\right)\Big|_{\theta=\hat{\theta}_{\varepsilon,z}} = \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta L(z_i,\hat{\theta}_{\varepsilon,z}) + \varepsilon\nabla_\theta L(z,\hat{\theta}_{\varepsilon,z}) = 0$$

ε = 0

$$\left[\frac{1}{n}\sum_{i=1}^{n}\nabla_\theta^2 L(z_i,\hat{\theta}_{\varepsilon,z}) + \varepsilon\nabla_\theta^2 L(z,\hat{\theta}_{\varepsilon,z})\right]\frac{d\hat{\theta}_{\varepsilon,z}}{d\varepsilon} + \nabla_\theta L(z,\hat{\theta}_{\varepsilon,z}) = 0$$

$$\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta})$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta^2 L(z_i,\hat{\theta})$

Influence Functions: Methodology

❑ Influence on the loss

- Analysis tool derived from influence functions by chain rule
- Quantifies the influence of a single training point on a specific test prediction's loss

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) \overset{\text{def}}{=} \left. \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon} \right|_{\epsilon=0}$$

Chain rule

$$= \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0}$$

$$= -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

$$\mathcal{I}_{\text{up,params}}(z)$$

Influence Functions: Limitations

❑ Problems of Traditional Influence Function

- Forming and inverting the hessian is costly
  - With n training points and $\theta \in R^p$, this requires $O(np^2 + p^3)$ operations

- Calculating the influence on the loss for all training points is often needed
  - Inefficient to calculate at every training point for a test sample

Influence Functions: Efficient Calculation

❑ Inverse Hessian Approximation by Hessian-Vector Product

- HVP: A method to compute a hessian-vector product without forming the hessian itself
- When approximating inverse hessian, the HVP is computed whenever a hessian-vector product is needed
- **Efficient implementation to scale up influence functions for modern, complex models**

$$v \xrightarrow[\text{HVP}]{} H_{\hat{\theta}} v \xrightarrow[\text{CG, Taylor}]{} H_{\hat{\theta}}^{-1} v$$

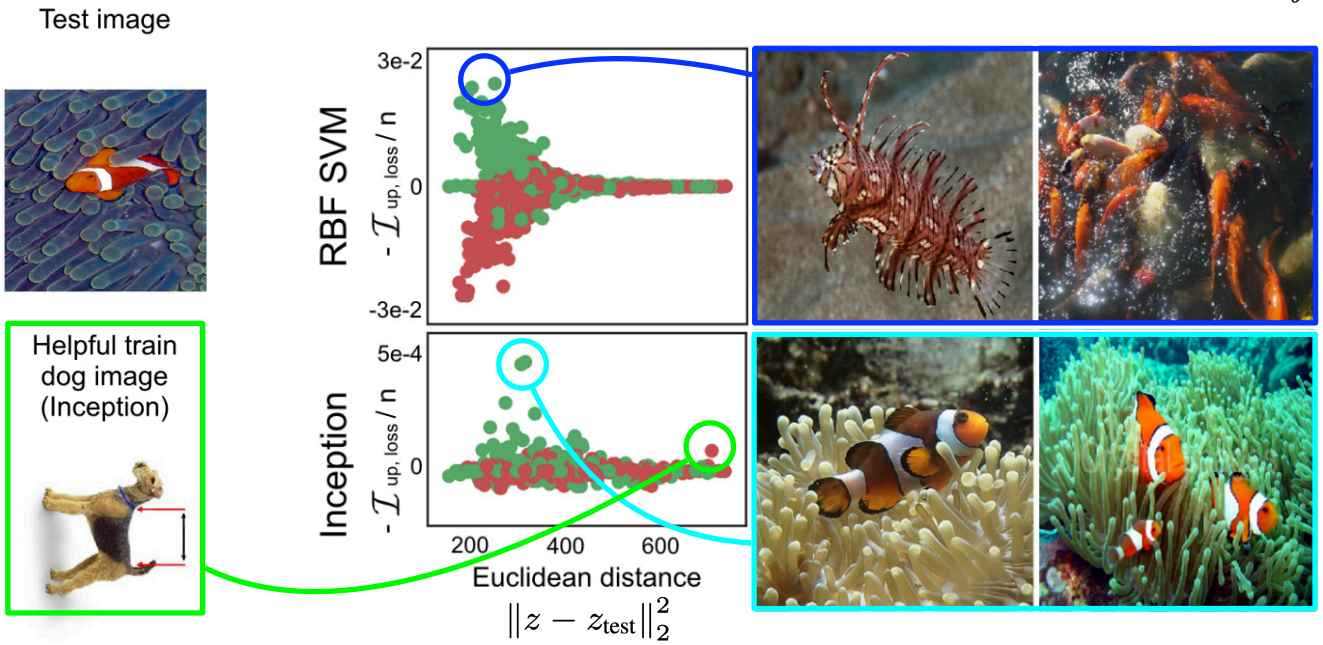
$$s_{\text{test}} \overset{\text{def}}{=} H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{\text{test}}, \hat{\theta}) \qquad \mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_\theta L(z, \hat{\theta})$$

Influence Functions: Use Cases

- **Understanding Model Behavior**

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) \stackrel{\text{def}}{=} \left. \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon} \right|_{\epsilon=0}$$

$$= -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$



Influence Functions: Use Cases

- **Fixing Mislabeled Examples**

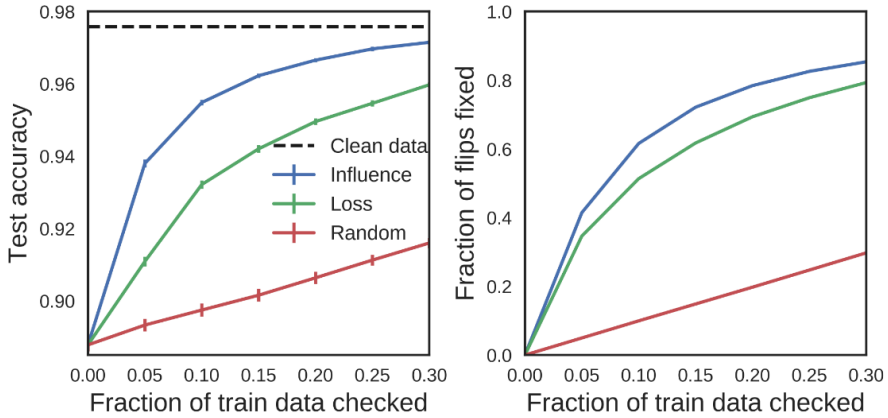


Figure 6. **Fixing mislabeled examples.** Plots of how test accuracy (left) and the fraction of flipped data detected (right) change with the fraction of train data checked, using different algorithms for picking points to check. Error bars show the std. dev. across 40 repeats of this experiment, with a different subset of labels flipped in each; error bars on the right are too small to be seen.

Summary

- Influence functions, a practical interpretability tool for modern ML

- Scalable to large neural networks by using HVP to approximate inverse hessian

$$\mathcal{I}_{\text{up,params}}(z) \overset{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\,\nabla_\theta L(z,\hat{\theta})$$

$$\begin{aligned}\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) &\overset{\text{def}}{=} \left.\frac{dL(z_{\text{test}},\hat{\theta}_{\epsilon,z})}{d\epsilon}\right|_{\epsilon=0} \\ &= -\nabla_\theta L(z_{\text{test}},\hat{\theta})^\top H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta})\end{aligned}$$

$$s_{\text{test}} \overset{\text{def}}{=} H_{\hat{\theta}}^{-1}\,\nabla_\theta L(z_{\text{test}},\hat{\theta}) \qquad \mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_\theta L(z,\hat{\theta})$$

INDEX

- Introduction
- Influence Functions
  - Methodology
  - Limitations
  - Efficient Calculation
  - Use Cases
- **Unlearning**
  - Needs for Systems to Forget
  - via Influence Functions
  - Tasks and Difficulties in Graphs
- **Graph Influence Functions**
  - Background
  - Methodology
  - Experiments
- **Conclusion**
- **Appendix**

Unlearning: Needs for Systems to Forget

- **Privacy**
  - "**Right to be Forgotten**" (e.g., GDPR in the EU, CCPA in California, and PIPEDA in Canada)
  - When a user requests deletion, the company should delete the requested data
- **Utility**
  - Poisoned data, out-of-date data or incorrect datas can degrade model quality
- **Efficiency**
  - Retraining from scratch for every unlearning request is impractical for large models

Unlearning: via Influence Functions

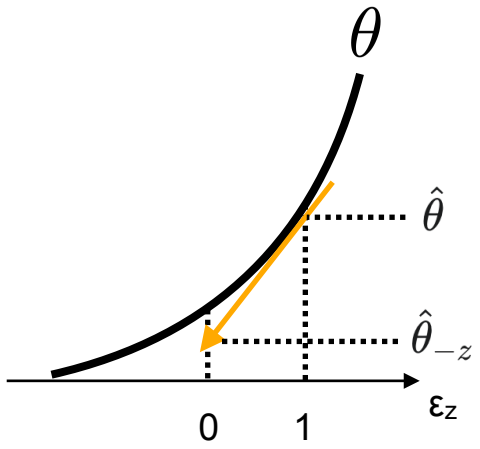
❑ Idea for Unlearning

- "If influence functions can calculate the impact of a sample, why not reverse it to **unlearn** it?"
- Upweight the unlearning point by an amount equal to one sample
- **However, there are more things to consider in graphs**

$$\hat{\theta}_{-z} - \hat{\theta} \approx \epsilon \cdot \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}$$

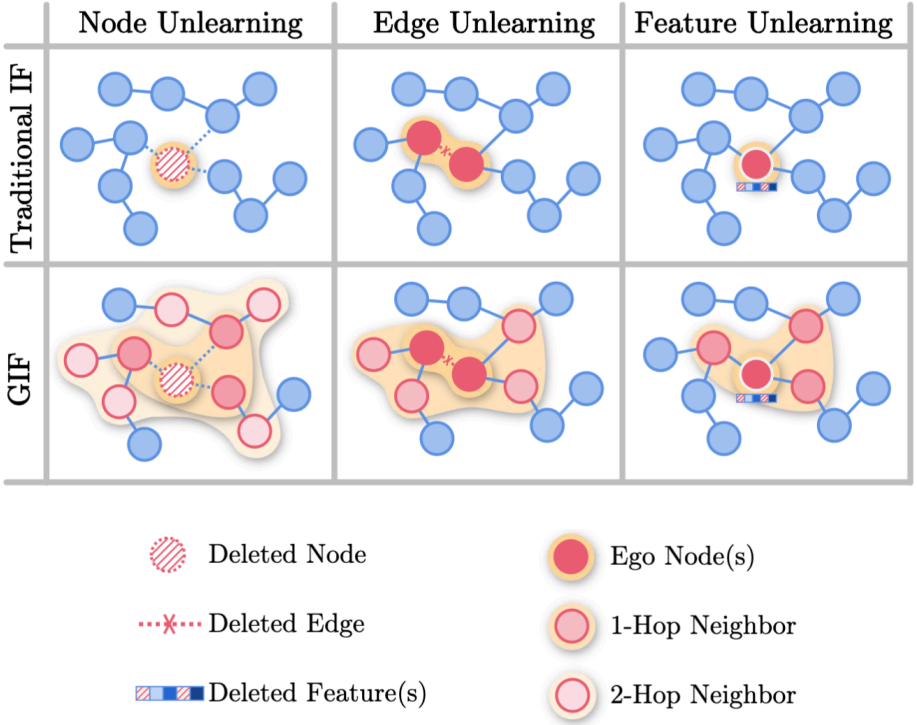
$$\hat{\theta}_{-z} \approx \hat{\theta} + \left(\epsilon \cdot \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right)$$

ε: amount of an single sample



Unlearning: Tasks in Graphs

- **Node Unlearning**
  - Deleting a node and connected edges
- **Edge Unlearning**
  - Deleting a edge
- **Feature Unlearning**
  - Deleting the influence of a feature in a node

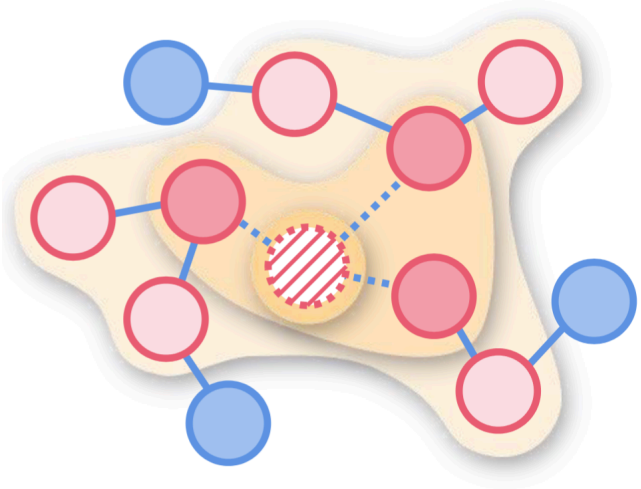


Unlearning: Difficulties in Graphs

❑ Message Passing in GNNs

- Need to remove not only the influence itself, but also the effect on multi-hop neighbors

➡ **Introduce an influence function tailored for graph unlearning**



Graph Influence Functions: Background

❑ Similarities to Influence Functions

$$\mathcal{I}_{\text{up,params}}(z) = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

- Hessian–vector products for efficient approximation of inverse Hessian

$$\hat{\theta} - \theta_0 = \theta_{\epsilon=-1} - \theta_0 \approx H_{\theta_0}^{-1} \nabla_{\theta_0} \mathcal{L}_{\Delta\mathcal{G}}$$

$\Delta\mathcal{G}$:Unlearning request
(nodes, edges and features to be removed)

❑ Differences from Influence Functions

- Include the neighbors affected by the perturbation of a single sample
- For affected nodes, uses the difference between the loss before and after removal

$$\hat{\theta}_{\epsilon} = \arg\min_{\theta}(\mathcal{L}_0 + \epsilon \Delta\mathcal{L}_{(\mathcal{G}\setminus\Delta\mathcal{G})}), \quad \Delta\mathcal{L}_{(\mathcal{G}\setminus\Delta\mathcal{G})} = \sum_{z_i \in \mathcal{D}} \hat{l}\,(z_i, y_i)$$

$$\hat{l}\,(z_i, y_i) = \begin{cases} l(f_{\mathcal{G}}(z_i), y_i), & z_i \in \Delta\mathcal{G} \\ l(f_{\mathcal{G}}(z_i), y_i) - l(f_{\mathcal{G}\setminus\Delta\mathcal{G}}(z_i), y_i), & z_i \text{ is influenced by } \Delta\mathcal{G} \\ 0, & \text{other nodes} \end{cases}$$

Graph Influence Functions: Methodology

❑ Influenced Region

- K-hop neighbors of the nodes influenced by the removal
- Unlearning a node also involves removing connected edges
- K-hop of the connected nodes is equivalent to the (k+1)-hop of the unlearned node

$$\mathcal{N}_k(z_i) = \{z_j | 1 \leqslant \mathrm{SPD}(z_j, z_i) \leqslant k\},$$

$$\mathcal{N}_k(e_i) = \mathcal{N}_k(z_1') \bigcup \mathcal{N}_k(z_2') \bigcup \{z_1', z_2'\},$$

- *For Node Unlearning request* $\Delta\mathcal{G} = \{\mathcal{V}^{(rm)}, \varnothing, \varnothing\}$, *the influenced region is* $\mathcal{N}_k(\mathcal{V}^{(rm)}) = \bigcup_{e_i \in \mathcal{V}^{(rm)}} \mathcal{N}_{k+1}(e_i)$;
- *For Edge Unlearning request* $\Delta\mathcal{G} = \{\varnothing, \mathcal{E}^{(rm)}, \varnothing\}$, *the influenced region is* $\mathcal{N}_k(\mathcal{E}^{(rm)}) = \bigcup_{z_i \in \mathcal{E}^{(rm)}} \mathcal{N}_k(z_i)$;
- *For Feature Unlearning request* $\Delta\mathcal{G} = \{\varnothing, \varnothing, \mathcal{X}^{(rm)}\}$, *the influenced region is* $\mathcal{N}_k(\mathcal{X}^{(rm)}) = \bigcup_{z_i \sim \mathcal{X}^{(rm)}} \mathcal{N}_k(z_i)$, *where* $z_i \sim \mathcal{X}^{(rm)}$ *indicates that the feature of node* z_i *is revoked.*

Graph Influence Functions: Methodology

❑ Resulting Formula

$$\hat{\theta}_\epsilon = \arg\min_\theta(\mathcal{L}_0 + \epsilon\Delta\mathcal{L}_{(\mathcal{G}\setminus\Delta\mathcal{G})}), \quad \Delta\mathcal{L}_{(\mathcal{G}\setminus\Delta\mathcal{G})} = \sum_{z_i\in\mathcal{D}} \hat{l}\,(z_i, y_i)$$

$$\hat{l}\,(z_i, y_i) = \begin{cases} l(f_{\mathcal{G}}(z_i), y_i), & z_i \in \Delta\mathcal{G} \\ l(f_{\mathcal{G}}(z_i), y_i) - l(f_{\mathcal{G}\setminus\Delta\mathcal{G}}(z_i), y_i), & z_i \text{ is influenced by } \Delta\mathcal{G} \\ 0, & \text{other nodes} \end{cases}$$

- *For Node Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i\in\mathcal{N}_k(\mathcal{V}^{(rm)})\cup\mathcal{V}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right)) - H_{\theta_0}^{-1} \sum_{z_i\in\mathcal{N}_k(\mathcal{V}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G}\setminus\Delta\mathcal{G}), y_i\right)).$$

- *For Edge Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i\in\mathcal{N}_k(\mathcal{E}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right)) - H_{\theta_0}^{-1} \sum_{z_i\in\mathcal{N}_k(\mathcal{E}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G}\setminus\Delta\mathcal{G}), y_i\right)).$$

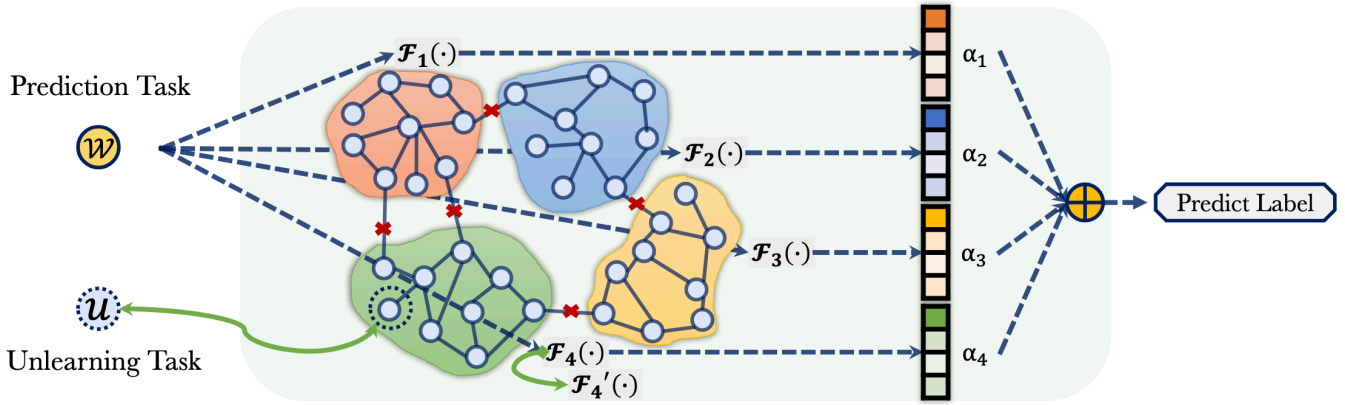
- *For Feature Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i\sim N_k(\mathcal{X}^{(rm)})\cup\mathcal{X}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right)) - H_{\theta_0}^{-1} \sum_{z_i\sim N_k(\mathcal{X}^{(rm)})\cup\mathcal{X}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G}\setminus\Delta\mathcal{G}), y_i\right)).$$

Graph Influence Functions: Experiments

❑ Utility and Efficiency

- Compared with graph eraser, SISA based graph unlearning method



Graph Influence Functions: Experiments

❑ Utility and Efficiency

Table 3: Comparison of F1 scores and running time (RT) for different graph unlearning methods for edge unlearning with 5% edges deleted from the original graph. 'LPA' and 'Kmeans' stand for GraphEraser [6] using balanced label propagation and balanced embedding k-means algorithm for community detection, respectively. The bold indicates the best result for each GNN model on each dataset.

Model		Dataset					
		Cora		Citeseer		CS	
Backbone	Strategy	F1 score	RT (second)	F1 score	RT (second)	F1 score	RT (second)
GCN	Retrain	0.8210±0.0055	6.33	0.7318±0.0096	7.52	0.9126±0.0055	55.95
	LPA	0.6790±0.0001	1.35	0.5556±0.0001	1.69	0.7732±0.0001	3.04
	Kmeans	0.5535±0.0001	1.89	0.5045±0.0001	1.56	0.7754±0.0001	9.97
	GIF	**0.8218±0.0066**	**0.16**	**0.6925±0.0060**	**0.13**	**0.9137±0.0016**	**0.26**
GAT	Retrain	0.8804±0.0060	15.72	0.7643±0.0049	19.50	0.9305±0.0011	110.39
	LPA	0.3432±0.0001	3.04	0.6997±0.0001	3.92	0.7650±0.0001	6.43
	Kmeans	0.6900±0.0001	3.19	0.7628±0.0001	3.41	0.8794±0.0001	17.36
	GIF	**0.8649±0.0072**	**0.86**	**0.7663±0.0072**	**0.59**	**0.9325±0.0015**	**1.02**
SGC	Retrain	0.8236±0.0142	6.63	0.7132±0.0091	7.16	0.9165±0.0045	58.12
	LPA	0.3247±0.0001	2.03	0.3934±0.0001	1.70	0.5267±0.0001	3.08
	Kmeans	0.3690±0.0001	1.41	0.3874±0.0001	1.56	0.6532±0.0001	10.59
	GIF	**0.8129±0.0110**	**0.12**	**0.6892±0.0082**	**0.12**	**0.9164±0.0051**	**0.24**
GIN	Retrain	0.8051±0.0144	8.48	0.7294±0.0207	9.94	0.8822±0.0074	70.38
	LPA	0.6605±0.0001	2.65	0.6096±0.0001	2.21	0.6510±0.0001	3.82
	Kmeans	0.7491±0.0001	2.53	0.6517±0.0001	2.11	0.8336±0.0001	10.29
	GIF	**0.8059±0.0234**	**0.48**	**0.7315±0.0185**	**0.48**	**0.8884±0.0083**	**0.57**

Graph Influence Functions: Experiments

❑ Unlearning Efficacy

- Novel evaluation method inspired by adversarial attacks

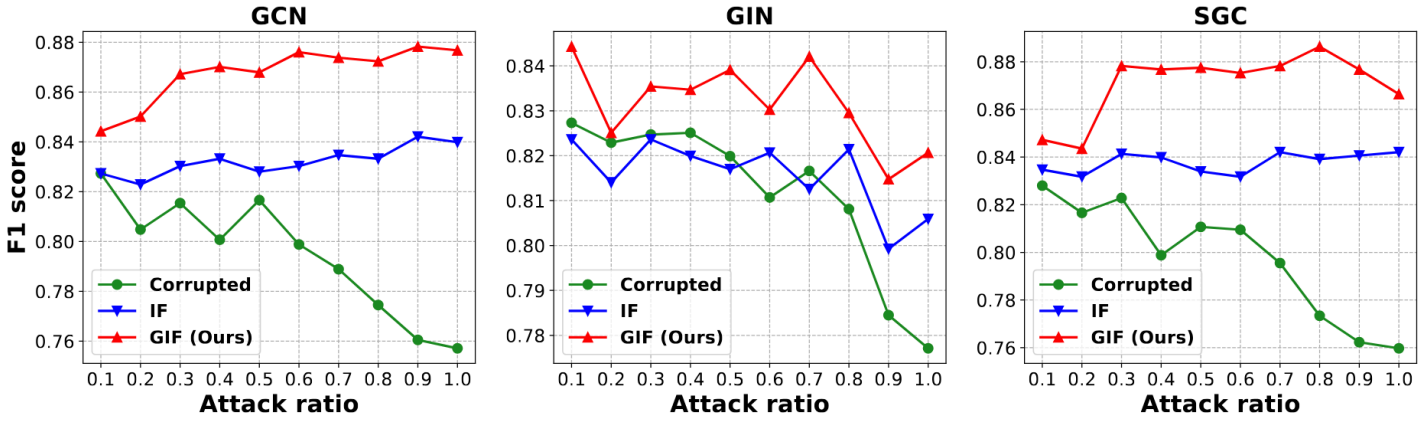


Figure 2: Comparison of unlearning efficacy over three GNN models.

Graph Influence Functions: Experiments

❑ Hyper-parameter Studies

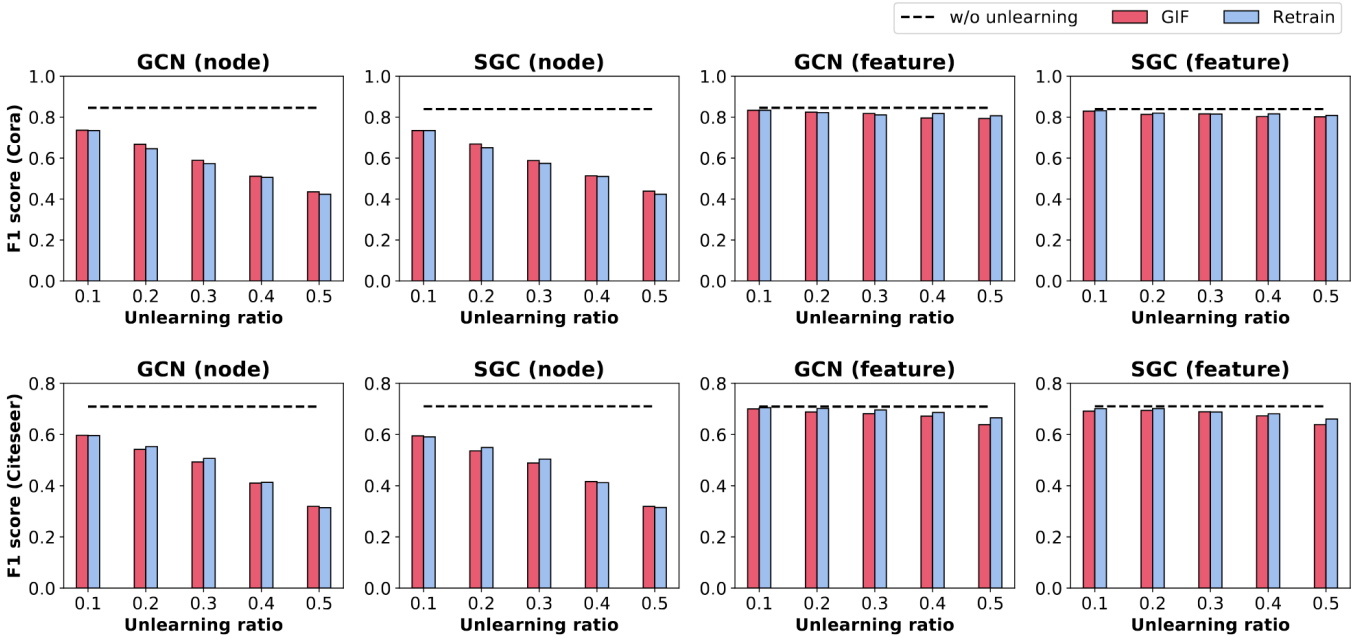
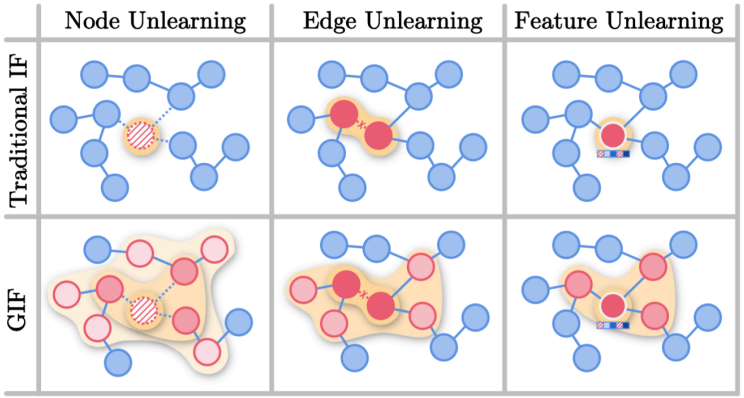


Figure 3: Impact of unlearning ratio ρ on node unlearning tasks and feature unlearning tasks.

Summary

- Graph Influence Functions, a function tailored for general graph unlearning
- Quantify the neighbor propagation effect according to the graph structure
- New evaluation method for graph unlearning



- *For Node Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i \in \mathcal{N}_k(\mathcal{V}^{(rm)}) \cup \mathcal{V}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right) - H_{\theta_0}^{-1} \sum_{z_i \in \mathcal{N}_k(\mathcal{V}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G} \setminus \Delta\mathcal{G}), y_i\right).$$

- *For Edge Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i \in \mathcal{N}_k(\mathcal{E}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right) - H_{\theta_0}^{-1} \sum_{z_i \in \mathcal{N}_k(\mathcal{E}^{(rm)})} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G} \setminus \Delta\mathcal{G}), y_i\right).$$

- *For Feature Unlearning tasks,*

$$\Delta\theta = H_{\theta_0}^{-1} \sum_{z_i \sim N_k(\mathcal{X}^{(rm)}) \cup \mathcal{X}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i), y_i\right) - H_{\theta_0}^{-1} \sum_{z_i \sim N_k(\mathcal{X}^{(rm)}) \cup \mathcal{X}^{(rm)}} \nabla_{\theta_0} l\left(f_{\mathcal{G}}(z_i; \mathcal{G} \setminus \Delta\mathcal{G}), y_i\right).$$

Conclusion

- Influence functions, a tool for understanding the black-box of predictions
- Applicable for unlearning tasks (e.g., graph influence functions)
- Potential to be used in many other fields

Thank you!

Appendix

❑ Hessian Matrix

- Second-order derivatives of a multivariate function
- Hessian shows how the gradient bends near given parameters

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Appendix

❑ Hessian-Vector Product

- Second-order information is available without forming hessian matrix

$$Hv = \nabla_\theta\left(\nabla_\theta L(\theta) \cdot v\right)$$

Pearlmutter trick

$$Hv \approx \frac{\nabla L(\theta+\epsilon v) - \nabla L(\theta)}{\epsilon}$$

Approximation

Appendix

❑ Another Method for Deriving the Influence Function

- One order taylor expansion is used to derive the influence function
- Since ΔG is a tiny subset of G, neglect the term $\nabla^2_{\theta_0}\mathcal{L}_{\Delta\mathcal{G}}$

$$\theta_\epsilon = \arg\min_\theta(\mathcal{L}_0 + \epsilon\mathcal{L}_{\Delta\mathcal{G}}) \qquad 0 = \nabla_{\theta_\epsilon}\mathcal{L}_0 + \epsilon\nabla_{\theta_\epsilon}\mathcal{L}_{\Delta\mathcal{G}}, \quad 0 = \nabla_{\theta_0}\mathcal{L}_0$$

$$0 \approx \Delta\theta\left(\nabla^2_{\theta_0}\mathcal{L}_0 + \epsilon\nabla^2_{\theta_0}\mathcal{L}_{\Delta\mathcal{G}}\right) + \left(\epsilon\nabla_{\theta_0}\mathcal{L}_{\Delta\mathcal{G}} + \nabla_{\theta_0}\mathcal{L}_0\right)$$

$$\hat{\theta} - \theta_0 = \theta_{\epsilon=-1} - \theta_0 \approx H^{-1}_{\theta_0}\nabla_{\theta_0}\mathcal{L}_{\Delta\mathcal{G}}$$

Appendix

❑ Conjugate Gradient

$$H \succ 0 \qquad H^{-1}v \;=\; \arg\min_{t} \; \tfrac{1}{2}\, t^\top H t - v^\top t$$

$$\begin{aligned}
\alpha_k &= \frac{r_k^\top r_k}{p_k^\top H p_k}, \\
t_{k+1} &= t_k + \alpha_k p_k, \\
r_{k+1} &= r_k - \alpha_k H p_k, \\
\beta_{k+1} &= \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}, \\
p_{k+1} &= r_{k+1} + \beta_{k+1} p_k.
\end{aligned}$$

Appendix

❑ Stochastic Estimation

$$H \succ 0 \qquad H^{-1}v = \arg\min_t \frac{1}{2} t^\top H t - v^\top t$$

THEOREM 5. *Suppose Hessian matrix H is positive definite. When the spectral radius of $(I - H)$ is less than 1, $H_0^{-1} = I$ and $H_n^{-1} = (I - H)H_{n-1}^{-1} + I$ for $n = 1, 2, 3, \cdots$, then $\lim_{n\to\infty} H_n^{-1} = H^{-1}$.*

$$[H_t^{-1}v] = v + [H_{t-1}^{-1}v] - H_{\theta_0}[H_{t-1}^{-1}v], \quad [H_0^{-1}v] = v$$

$$[H_j^{-1}v] = v + [H_{j-1}^{-1}v] - \lambda H_{\theta_0}[H_{j-1}^{-1}v], \quad [H_0^{-1}v] = v$$

Appendix

❑ Influence Functions vs. Leave-one-out Retraining

- While assuming convexity, it remains effective even for non-convex models