FASTIF
Scalable Influence Functions for Efficient Model Interpretation and Debugging

Nguyễn Đức Thắng

Resident at FSOFT AI LAB

Ngày 4 tháng 7 năm 2021

FASTIF
Scalable Influence Functions for Efficient Model Interpretation and Debugging

Nguyễn Đức Thắng

Background

FASTIF

Analysis

Content

Background

- Influence of upweighting z on the loss at a test point z_{test}:

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{1}$$

- For each test data point, we can pre-compute and cache the following quantity:

$$s_{\text{test}} = H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right) \tag{2}$$

- Compute the influence for each training data-point z:

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_\theta L(z, \hat{\theta}) \tag{3}$$

Background

- Approximation $s_{\text{test}} = H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)$:

$$\tilde{H}_0^{-1} v = v_0 = \nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right) \tag{4}$$

$$\tilde{H}_j^{-1} v = v + \left(I - \nabla_\theta^2 L\left(z_{s_j}, \hat{\theta}\right)\right) \tilde{H}_{j-1}^{-1} v \tag{5}$$

$$= v + I H_{j-1}^{-1} v - \nabla_\theta^2 L(z_{s_j}, \hat{\theta}) H_{j-1}^{-1} v \tag{6}$$

- Approximation Hv corresponding to $\nabla_\theta^2 L\left(z_{s_j}, \hat{\theta}\right) \tilde{H}_{j-1}^{-1} v$:

$$\mathbf{H}\mathbf{v} = \nabla_\theta \left(\mathbf{v} \cdot \nabla_\theta L\right) \tag{7}$$

- Repeat (4) and (5) for T times independently, and return the averaged inverse HVP estimations.

Background

Cost and approximation quality depends:

- J: the number of recursive iterations.
- T: the number of independent runs.
- B: the batch size of sample points from training data (number of z_{s_j}).

Calculate influence on the full training data.

$$z^* = \arg\max_{z \in \mathcal{Z}} \mathcal{I}\left(z, z_{\text{test}}\right) \tag{8}$$

FASTIF

1. Speeding up the *argmax* using kNN.
2. Speeding up the Inverse Hessian.
3. Details on the Parallelization.

Speeding up the argmax using kNN

- Search to a subset of promising data points, $\hat{\mathcal{Z}} \subseteq \mathcal{Z}$:

$$z^* = \arg\max_{z \in \hat{\mathcal{Z}}} \mathcal{I}\left(z, z_{\text{test}}\right) \tag{9}$$

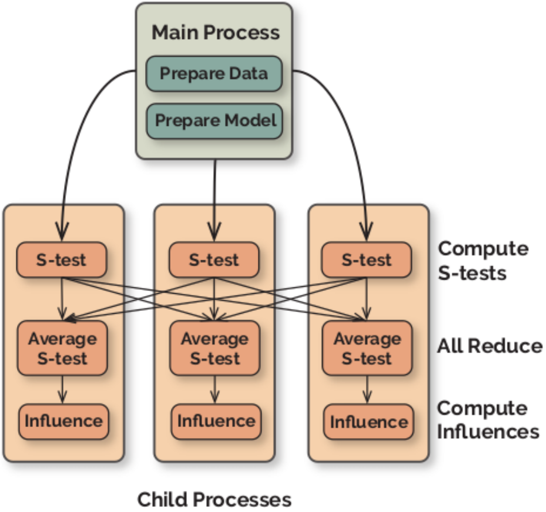
- Select subset $\hat{\mathcal{Z}}$ as the top-k nearest neighbors of z_{test} based on the ℓ_2 distance between extracted features of the data-points.
- Can using libraries such as FAISS.

Speeding up the Inverse Hessian

Propose a few simple changes:

- Choose a J so that approximation converges.
- Choose a small batch size. In our experiments, we found that even $B = 1$ suffices.
- Make up for the noisiness of small batch size using larger T , which can be distributed over multiple GPUs.

Details on Parallelization



Details on Parallelization

	Original s_{test} (1 GPU)	**Fast s_{test} (1 GPU)**	**Fast s_{test} (4 GPUs)**
No kNN	≥ 2 hours (1X)	/	/
k**NN** $k = 1e4$	14.72 ± 0.21 min (8X)	6.61 ± 0.03 min (18X)	2.36 ± 0.02 min (50X)
k**NN** $k = 1e3$	10.93 ± 0.04 min (10X)	3.13 ± 0.05 min (38X)	1.46 ± 0.07 min (82X)

Table 1: Speed of influence functions. All experiments are measured on MultiNLI dataset. The run-times are averages and standard deviations over 10 examples.

Analysis

1. Recall of kNN
2. Inverse-Hessian-Vector-Product Approximation
   Speed-Quality Trade-Off
3. Quality of Influence Estimations

Recall of kNN

- If a data-points in influential, will it be included in the subset selected by the kNN?
- Define the recall score R@m as the percentage of top-m ground-truth influential data-points that are selected by the kNN.

$$R@m = \frac{|\,\{\text{ retrieved }\} \cap \{\text{ top- } m \text{ influential }\}\,|}{|\,\{\text{ top- } m \text{ influential }\}\,|} \tag{10}$$

Recall of kNN

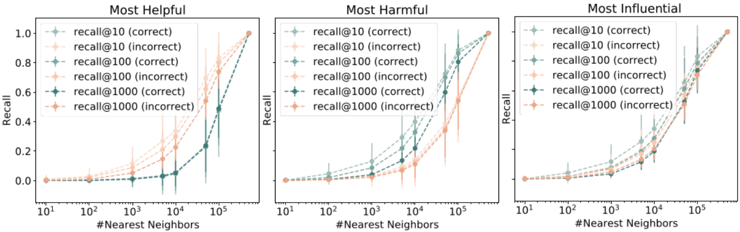


Figure 3: The recall of kNN in terms of finding influential data-points. The lines and error bars represent the means and standard deviations across 100 correct/incorrect predictions.

Inverse-Hessian-Vector-Product Approximation Speed-Quality Trade-Off

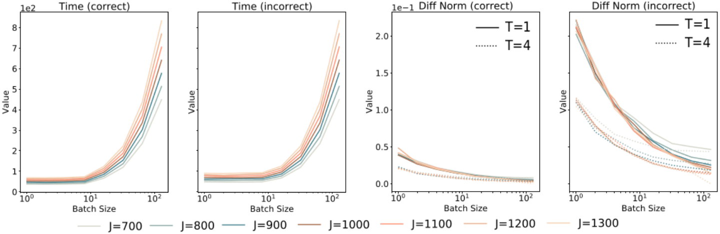


Figure 4: **Left Half:** computational time of Hessian approximation as a function of batch size and recursive iterations J. We further break down the figure into two sub-figures: cases when the prediction is correct and those when it is incorrect. **Right Half:** estimation error norm as a function of batch size, recursive iterations, whether the prediction is correct, and additionally the number of independent runs T.

Quality of Influence Estimations

Comparison	**Pearson**	**Spearman**	**Kendall**
Fast ($k = 10^3$) vs Full	99.9 ± 0.05	99.8 ± 0.39	97.4 ± 1.69
Fast ($k = 10^4$) vs Full	99.9 ± 0.04	99.9 ± 0.09	98.0 ± 0.77
Fast ($k = 10^3$) vs Fast ($k = 10^4$)	99.9 ± 0.08	99.8 ± 0.35	97.3 ± 1.66

Table 3: Correlations between influence values using various measures. The means and standard deviations are computed with 6 evaluation points (balanced between correct and incorrect predictions).

Thank you for your attention!