Capturing the Temporal Dependence of Training Data Influence

Wang, Jiachen T., et al. "Capturing the Temporal Dependence of Training Data Influence." arXiv preprint arXiv:2412.09538 (2024).

Hyunwoo Jung

POSTECH

hyunwoojung@postech.ac.kr

Content

Content

Data influence estimation

- Data influence estimation?
  - Aims to provide insights into the impact of specific data points on the model’s predictive bahaviors.
  - How would the model’s behavior change if we removed a specific training data point?
- Why?
  - Model transparency

    ⇛How it works?
  - Model accountability

    ⇛Can we believe model?
  - Significant role in addressing AI copyright debate

    ⇛Who made this model?

Influence function

- How small difference on training data influences test loss?
- Influence on test loss
  - $\mathbf{IF}(z_i) \coloneqq -\nabla_\theta \ell\left(\theta, z^{(val)}\right)^\top \mathbf{H}^{-1} \nabla_\theta \ell(\theta, z_i)$
    where $\mathbf{H} = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta^2 \ell(\theta, z_i)$ is Hessian of total loss,
    $\nabla_\theta \ell(\theta, z_i)$: gradient of loss regard to data point z

- Huge computation on Hessian inverse
- Neglects training procedure

Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." International conference on machine learning. PMLR, 2017.

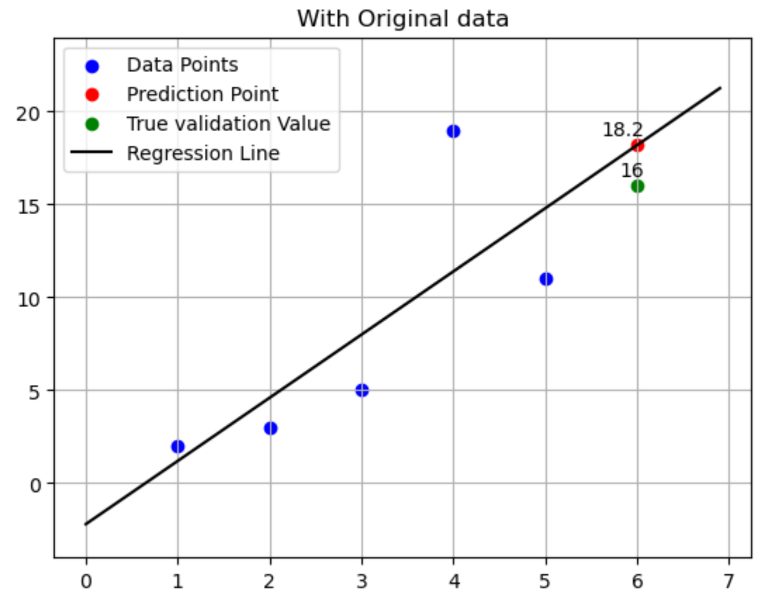
Data shapley

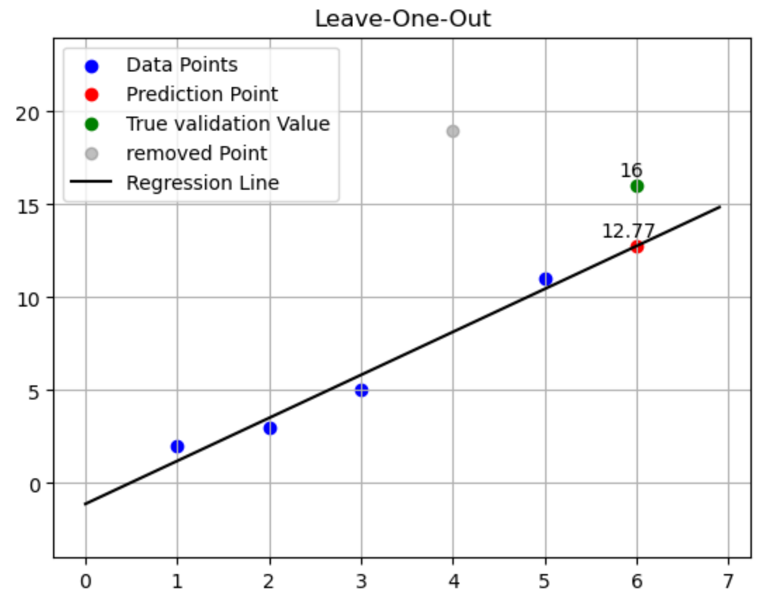
- Check data point's marginal contribution on final model.
- $\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \left(v(S \cup \{i\}) - v(S) \right)$
  - N: all dataset, S: subsets doesn't have i, $v(S)$: score only with S, $v(S \cup \{i\}) - v(S)$: how much score increase with i
- For all combinations, calculate average contribution when i is included.
  - E.g.)Want to check C's contribution on data set $\{A, B, C\}$
    - Make all combinations without C $\Rightarrow \emptyset, \{A\}, \{B\}, \{A, B\}$
    - Put C in that combination $\Rightarrow \{C\}, \{A, C\}, \{B, C\}, \{A, B, C\}$
    - And compare the score to calculate marginal contribution.
- Considering and computing all combination is hard.

Ghorbani, Amirata, and James Zou. "Data shapley: Equitable valuation of data for machine learning." International conference on machine learning. PMLR, 2019.

Leave-one-out influence

- Leave-one-out influence (LOO)
  - The model's loss change on a validation data $z^{(val)}$ when the training data point z^* is removed from the training set $\mathcal{D}$
  - As formula ($\mathcal{A}$ is learning algorithm),
    $$\mathrm{LOO}\left(z^*; z^{(val)}\right) \coloneqq \ell\left(\mathcal{A}(\mathcal{D}), z^{(val)}\right) - \ell\left(\mathcal{A}(\mathcal{D}\backslash\{z^*\}), z^{(val)}\right)$$
  - E.g) $\mathcal{A}$: Linear Regression





$$\mathrm{LOO}\left(z^*; z^{(val)}\right) = (16 - 18.2) - (16 - 12.77) = -5.43$$

Data influence estimation

- Traditional literature usually assumes that learning altorithm $\mathcal{A}$ is permutation invariant.
  - The order of data point does not affect the outcome.
  - This assumption holds for strongly convex loss functions.

- Does Mordern training algorithms are permutation invariant?
  - No!!

    ⇒Non-convex nature of neural networks

    ⇒Multi-stage training curricula

Data influence estimation

- Additionally, LLM(Large Language Models) often undergo just one training epoch, meaning each data point is encdountered only once during training.

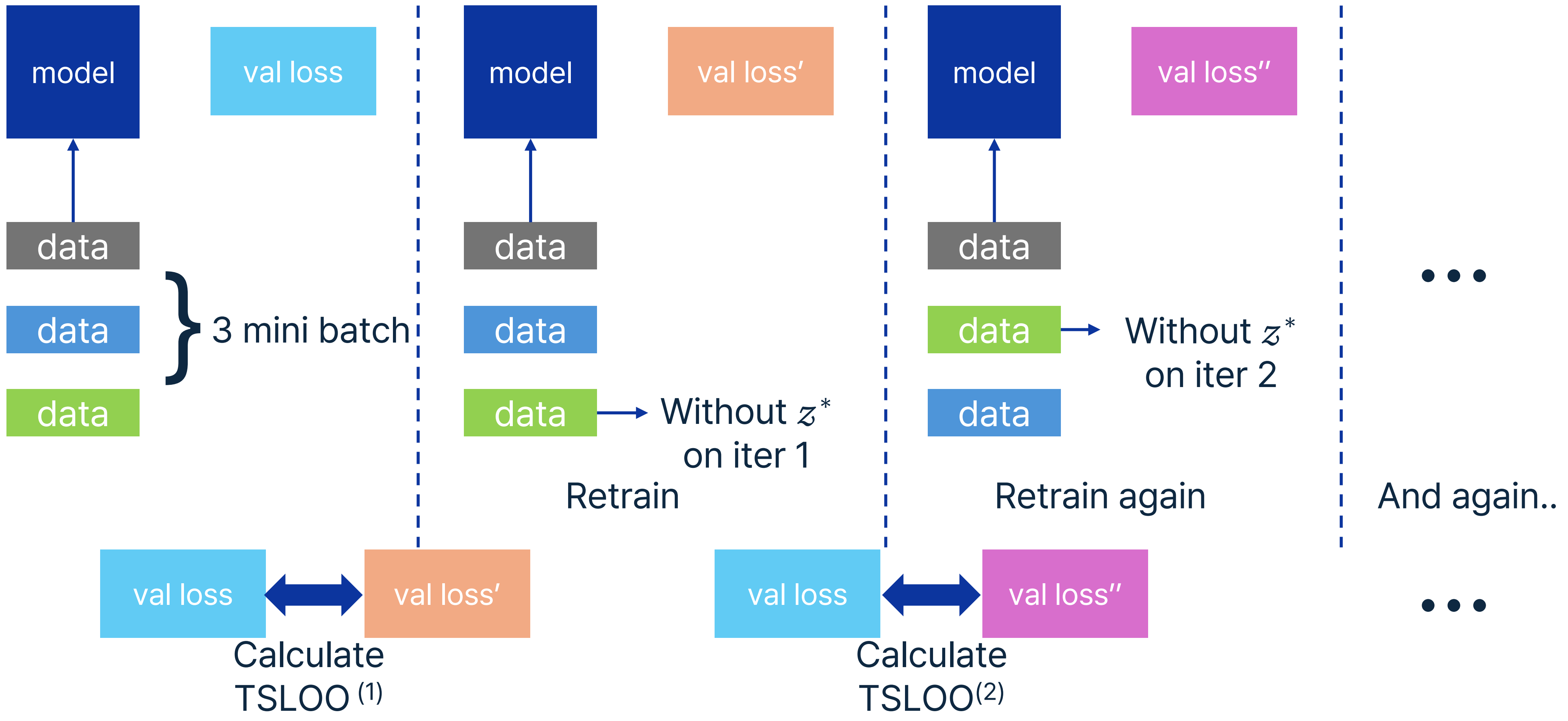
- For that reason, data order make influence of data points!

- So, how can we calculate influence of data on non-convex, sequence dependent scenario?
  - Trajectory-specific LOO

  ⇒Characterizes the loss change resulting from removing a data point from the specific iteration.

Data influence estimation

- But, Computationally challenging evaluating trajectory-specific LOO...



- To avoid this, author introduced data value embedding ⇒ approximation

Content

Trajecctory-specific LOO

- Originally introduced as ‘SGD-influence’ [1]
- In standard SGD, parameters are updated as
  - $\theta_{t+1} = \theta_t - \eta_t \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta_t, z)$
  - η_t: learning rate

- In this scenario, parameters are updated as
  - $\theta_{t+1} = \theta_t - \eta_t \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta_t, z)$ for $t = 0, \dots, t_s - 1$
  - $\theta'_{t_s+1} = \theta_{t_s} - \eta_{t_s} \sum_{z \in \mathcal{B}_{t_s} \setminus \{z^*\}} \nabla \ell(\theta_t, z)$ for iteration t_s which z^* is removed
  - $\theta'_{t+1} = \theta'_t - \eta_t \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta'_t, z)$ for $t = t_s + 1, \dots, T - 1$

  ⇒It just says that we use standard SGD for update,
  simply removing specific datapoint on specific iteration.

[1] Hara, Satoshi, Atsushi Nitanda, and Takanori Maehara. "Data cleansing for models trained with SGD." Advances in Neural Information Processing Systems 32 (2019)

Trajecctory-specific LOO

- $TSLOO^{(t_s)}(z^*; z^{val}) := \ell(\theta'_T, z^{val}) - \ell(\theta_T, z^{val})$
    - z^*: data point, t_s: iteration, $z^{(val)}$: validation point
    - θ_T: final model in original SGD, θ'_T: final model in this scenario

  ⇒Means change in the validation loss $\ell(\cdot, z^{(val)})$, when the data point $z^* \in \mathcal{B}_{t_s}$ is removed from iteration t_s.

- TSLOO depends on the "timing" that data is used.

Data value embedding

- Like said previous, TSLOO's computation is significant challenge.

  ⇒Data value embedding!

- We will establish approximation for

$$TSLOO^{(t_s)}\left(z^*; z^{val}\right) = \ell\left(\theta_T', z^{val}\right) - \ell\left(\theta_T, z^{val}\right)$$

- First, define interpolation between θ_T, θ_T'
  - $\theta_{k+1} = \theta_k - \eta_k \sum_{z\in\mathcal{B}_k} \nabla\ell(\theta_k, z)$: before t_s
  - $\theta_{t_s+1}(\varepsilon) \coloneqq \theta_{t_s} - \eta_{t_s} \sum_{z\in\mathcal{B}_{t_s}\setminus\{z^*\}} \nabla\ell\left(\theta_{t_s}, z\right) - \eta_{t_s}(1-\varepsilon)\nabla\ell\left(\theta_{t_s}, z^*\right)$: t_s
  - $\theta_{k+1}(\varepsilon) \coloneqq \theta_k(\varepsilon) - \eta_k \sum_{z\in\mathcal{B}_k} \nabla\ell(\theta_k(\varepsilon), z)$: after t_s

    $\therefore \theta_T(0) = \theta_T,\ \theta_T(1) = \theta_T'$

Data value embedding

- Then apply first-order Taylor expansion around $\varepsilon = 0$,
  - $\ell\left(\theta_T', z^{(val)}\right) - \ell\left(\theta_T, z^{(val)}\right) \approx \nabla\ell\left(\theta_T, z^{(val)}\right)^{\top} \left.\frac{\partial\theta_T(\varepsilon)}{\partial\varepsilon}\right|_{\varepsilon=0}$
- Proof.
  - Let $\ell\left(\theta_T(\varepsilon), z^{(val)}\right) = f(\varepsilon)$
  - $f(\varepsilon) \approx f(0) + \left.\frac{df(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} \times (\varepsilon - 0)\ (\because near\ \varepsilon = 0)$
  - $f(1) \approx f(0) + \left.\frac{df(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} \times (1 - 0)$

    $\therefore f(1) - f(0) \approx \left.\frac{df(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0}$

  $\Rightarrow \ell\left(\theta_T(1), z^{(val)}\right) - \ell\left(\theta_T(0), z^{(val)}\right) \approx \left.\frac{\partial}{\partial\varepsilon}\ell\left(\theta_T(\varepsilon), z^{(val)}\right)\right|_{\varepsilon=0}$

Data value embedding

- Then apply first-order Taylor expansion around $\varepsilon = 0$,
  - $\ell\left(\theta'_T, z^{(val)}\right) - \ell\left(\theta_T, z^{(val)}\right) \approx \nabla\ell\left(\theta_T, z^{(val)}\right)^{\top} \left.\frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0}$
- Proof. (cont.)
  - $\ell\left(\theta_T(1), z^{(val)}\right) - \ell\left(\theta_T(0), z^{(val)}\right) \approx \left.\frac{\partial}{\partial \varepsilon}\ell\left(\theta_T(\varepsilon), z^{(val)}\right)\right|_{\varepsilon=0}$
  - And since ε is not directly related to $\ell\left(\theta_T(\varepsilon), z^{(val)}\right)$, we need to apply the chain rule.
  - $\left.\frac{\partial}{\partial \varepsilon}\ell\left(\theta_T(\varepsilon), z^{(val)}\right)\right|_{\varepsilon=0} = \left.\frac{\partial}{\partial \theta_T(\varepsilon)}\ell\left(\theta_T(\varepsilon), z^{(val)}\right)^{\top} \frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0}$
  - Simply, $\nabla\ell\left(\theta_T, z^{(val)}\right)^{\top} \left.\frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0} \quad \left(\because \theta_T = \theta_T(0)\right)$

$$\Rightarrow \ell\left(\theta'_T, z^{(val)}\right) - \ell\left(\theta_T, z^{(val)}\right) \approx \nabla\ell\left(\theta_T, z^{(val)}\right)^{\top} \boxed{\left.\frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0}}$$

Data value embedding

- $\left.\frac{\partial\theta_T(\varepsilon)}{\partial\varepsilon}\right|_{\varepsilon=0}$ satisfies a recursive relation!
  - Parameter update using SGD interpolation
    - $\theta_{k+1} = \theta_k - \eta_k \sum_{z\in\mathcal{B}_k} \nabla\ell(\theta_k, z)$: before t_s
    - $\theta_{t_s+1}(\varepsilon) = \theta_{t_s} - \eta_{t_s} \sum_{z\in\mathcal{B}_{t_s}\setminus\{z^*\}} \nabla\ell\left(\theta_{t_s}, z\right) - \eta_{t_s}(1-\varepsilon)\nabla\ell\left(\theta_{t_s}, z^*\right)$: t_s
    - $\theta_{k+1}(\varepsilon) = \theta_k(\varepsilon) - \eta_k \sum_{z\in\mathcal{B}_k} \nabla\ell(\theta_k(\varepsilon), z)$: after t_s

  - Derivative is 0 before t_s since ε not occurred yet.
  - Calculate partial derivative regard to ε after t_s
    - $\frac{\partial\theta_{k+1}(\varepsilon)}{\partial\varepsilon} = \frac{\partial\theta_k(\varepsilon)}{\partial\varepsilon} - \eta_k \sum_{z\in\mathcal{B}_k} \nabla^2\,\ell(\theta_k(\varepsilon), z)\,\frac{\partial\theta_k(\varepsilon)}{\partial\varepsilon}$

$= \frac{\partial\theta_k(\varepsilon)}{\partial\epsilon}\left(\boldsymbol{I} - \eta_k \mathbf{H}_k(\varepsilon)\right) \left(where\ \mathbf{H}_k(\varepsilon) = \sum_{z\in\mathcal{B}_k} \nabla^2\,\ell(\theta_k(\varepsilon), z)\right)$

Data value embedding

- $\left.\frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0}$ satisfies a recursive relation!
  - Since SGD interpolation update on iter t_s is different, get partial derivative separately.
    - $\theta_{t_s+1}(\varepsilon) = \theta_{t_s} - \eta_{t_s} \sum_{z \in \mathcal{B}_{t_s} \setminus \{z^*\}} \nabla \ell(\theta_{t_s}, z) - \eta_{t_s}(1-\varepsilon) \nabla \ell(\theta_{t_s}, z^*)$
  - Calculate partial derivative regard to ε
    - $\frac{\partial \theta_{t_s+1}(\varepsilon)}{\partial \varepsilon} = \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*)$

  - So, $\left.\frac{\partial \theta_T(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0} = \left.\left[\prod_{k=t_s+1}^{T-1}\left(\boldsymbol{I} - \eta_k \mathbf{H}_k(\varepsilon)\right)\right] \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*)\right|_{\varepsilon=0}$

$= \left[\prod_{k=t_s+1}^{T-1}(\boldsymbol{I} - \eta_k \mathbf{H}_k)\right] \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*)$

($\mathbf{H}_k(0) = \mathbf{H}_k$ since $\varepsilon = 0$ means z^* included)

Data value embedding

- Finally, the approximation of $TSLOO^{(t_s)}\left(z^*; z^{val}\right)$ is
  - $\ell\left(\theta_T', z^{(val)}\right) - \ell\left(\theta_T, z^{(val)}\right)$

    $\approx \nabla\ell\left(\theta_T, z^{(val)}\right)^\top \left.\frac{\partial\theta_T(\varepsilon)}{\partial\varepsilon}\right|_{\varepsilon=0} (\because\ first\ order\ Talyor\ approximation)$

    $= \eta_{t_s} \nabla\ell\left(\theta_T, z^{(val)}\right)^\top \left[\prod_{k=t_s+1}^{T-1}(\boldsymbol{I} - \eta_k \mathbf{H}_k)\right] \nabla\ell\left(\theta_{t_s}, z^*\right)$

- Next, extract the test-data-independent components and define "data value embedding" for a training point $z^* \in \mathcal{B}_{t_s}$
  - $\text{DVEmb}^{(t_s)}(z^*) \coloneqq \eta_{t_s} \left[\prod_{k=t_s+1}^{T-1}(\boldsymbol{I} - \eta_k \mathbf{H}_k)\right] \nabla\ell\left(\theta_{t_s}, z^*\right)$
  - Of course, we can quickly determine training data z^*'s influence by
    - $\nabla\ell\left(\theta_T, z^{(val)}\right)^\top \text{DVEmb}^{(t_s)}(z^*)$

Content

Efficient computation and storage

- Still, there are some problem…
  - Computing data value embedding
    - Need gradient per sample
    - Need Hessian per step
  - Storing data value embedding
    - $\dim\left(\mathrm{DVEmb}_t(z^*)\right) = \dim(model)$
    - Hard to storing individual embeddings for each training data point.
- How to deal with it?

Recursive approximation of DVEmb

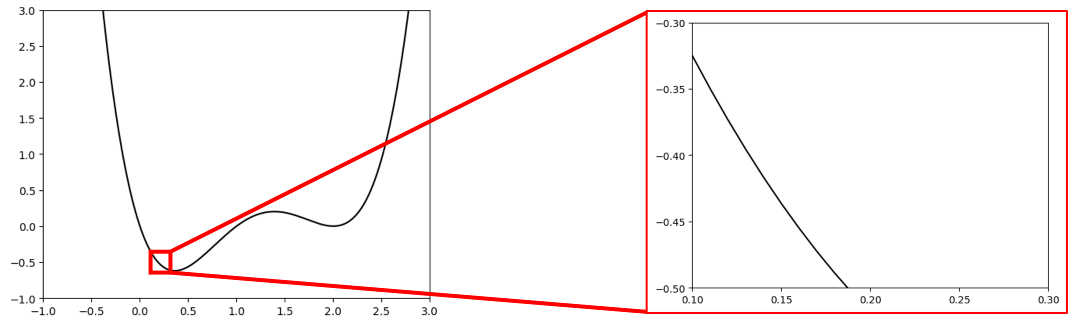
- Use Generalized Gauss-Newton (GGN) for Hessian matrix $\mathbf{H}_k$
  - $\mathbf{H}_t \approx \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta_t, z) \nabla \ell(\theta_t, z)^\top$ (in context of cross-entropy loss)
- Proof.
  - Assume that using cross entropy loss function with one-hot encoded label

$$L(y, f) = -\sum_{i=1}^{C} y_i \log(f_i)$$

  - $\boldsymbol{y} = [y_1, y_2, \dots, y_C]^\top$ is true label vector
    $\boldsymbol{f} = [f_1, f_2, \dots, f_C]^\top$ is predicted probability vector
  - $\nabla_\theta L = \sum_{i=1}^{C} \frac{\partial L}{\partial f_i} \frac{\partial f_i}{\partial \theta} = \sum_{i=1}^{C} \left(-\frac{y_i}{f_i} \frac{\partial f_i}{\partial \theta} \right) = -\frac{1}{f_k} \frac{\partial f_k}{\partial \theta}$ $(\because \textit{onehot encoded label})$
    - $y_k = 1$ for correct class k, $y_i = 0$ for $i \neq k$

Recursive approximation of DVEmb

- Use Generalized Gauss-Newton (GGN) for Hessian matrix $\mathbf{H}_k$
  - $\mathbf{H}_t \approx \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta_t, z) \nabla \ell(\theta_t, z)^\top$ (in context of cross-entropy loss)
- Proof. (cont.)
  - $\mathbf{H} = \nabla_\theta^2 L = \frac{\partial}{\partial \theta}\left(-\frac{1}{f_k}\frac{\partial f_k}{\partial \theta}\right) = \frac{1}{f_k^2}\frac{\partial f_k}{\partial \theta}\left(\frac{\partial f_k}{\partial \theta}\right)^\top - \frac{1}{f_k}\frac{\partial^2 f_k}{\partial \theta^2}$
  - we update parameter near the current parameter.
  - Plus, f_k is approximately linear in θ near the current parameter values.
  - Thus, $-\frac{1}{f_k}\frac{\partial^2 f_k}{\partial \theta^2}$ can be removed. ((linear function)″ = 0)



Recursive approximation of DVEmb

- Use Generalized Gauss-Newton (GGN) for Hessian matrix $\mathbf{H}_k$
  - $\mathbf{H}_t \approx \sum_{z \in \mathcal{B}_t} \nabla \ell(\theta_t, z) \nabla \ell(\theta_t, z)^\top$ (in context of cross-entropy loss)
- Proof. (cont.)
  - So $\mathbf{H} \approx \frac{1}{f_k^2} \frac{\partial f_k}{\partial \theta} \left(\frac{\partial f_k}{\partial \theta} \right)^\top$
  - $\nabla_\theta L = -\frac{1}{f_k} \frac{\partial f_k}{\partial \theta} \Rightarrow \mathbf{H} \approx \nabla_\theta L \nabla_\theta L^\top$

    $\Rightarrow$Outer product of the gradient of loss, approximates Hessian.

Recursive approximation of DVEmb

- Using approximation of Hessian $\mathbf{H}_t \approx \sum_{z\in\mathcal{B}_t} \nabla\ell(\theta_t, z)\nabla\ell(\theta_t, z)^\top$
- $\mathrm{DVEmb}^{(t_s)}(z^*) = \underline{\eta_{t_s}\left[\prod_{k=t_s+1}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\nabla\ell\left(\theta_{t_s}, z^*\right)}$

$$= \eta_{t_s}\left[\prod_{k=t_s+2}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\left(\boldsymbol{I}-\eta_{t_s+1}\mathbf{H}_{t_s+1}\right)\nabla\ell\left(\theta_{t_s}, z^*\right)$$

$$\approx \eta_{t_s}\left[\prod_{k=t_s+2}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\left(\boldsymbol{I}-\eta_{t_s+1}\sum_{z\in\mathcal{B}_{t_s+1}}\nabla\ell\left(\theta_{t_s+1}, z\right)\nabla\ell\left(\theta_{t_s+1}, z\right)^\top\right)\nabla\ell\left(\theta_{t_s}, z^*\right)$$

$$= \eta_{t_s}\left[\prod_{k=t_s+2}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\nabla\ell\left(\theta_{t_s}, z^*\right) - \eta_{t_s}\sum_{z\in\mathcal{B}_{t_s+1}}\left(\underline{\eta_{t_s+1}\left[\prod_{k=t_s+2}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\nabla\ell\left(\theta_{t_s+1}, z\right)}\right)\nabla\ell\left(\theta_{t_s+1}, z\right)^\top\nabla\ell\left(\theta_{t_s}, z^*\right)$$

$$= \eta_{t_s}\left[\prod_{k=t_s+2}^{T-1}(\boldsymbol{I}-\eta_k\mathbf{H}_k)\right]\nabla\ell\left(\theta_{t_s}, z^*\right) - \eta_{t_s}\sum_{z\in\mathcal{B}_{t_s+1}}\left(\nabla\ell\left(\theta_{t_s+1}, z\right)^\top\nabla\ell\left(\theta_{t_s}, z^*\right)\right)\mathrm{DVEmb}^{(t_s+1)}(z)$$

$$= \ldots$$

$$= \eta_{t_s}\nabla\ell\left(\theta_{t_s}, z^*\right) - \eta_{t_s}\sum_{k=t_s+1}^{T-1}\left(\sum_{z\in\mathcal{B}_k}\left(\nabla\ell(\theta_k, z)^\top\nabla\ell\left(\theta_{t_s}, z^*\right)\right)\mathrm{DVEmb}^{(k)}(z)\right)$$

Recursive approximation of DVEmb

- $\mathrm{DVEmb}^{(t_s)}(z^*) = \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) - \eta_{t_s} \sum_{t=t_s+1}^{T-1} \left(\sum_{z \in \mathcal{B}_t} \left(\nabla \ell(\theta_t, z)^\top \nabla \ell(\theta_{t_s}, z^*) \right) \mathrm{DVEmb}^{(t)}(z) \right)$
- This provides crucial insight
  - Gradient similarity term $\nabla \ell(\theta_t, z)^\top \nabla \ell(\theta_{t_s}, z^*)$ increases when z and z^* are similar.

  ⇒The more similar data point to z^* appears, the more z^*'s influence decreases.

- Plus, this equation suggests the possibility of a back-propagation algorithm for computing data value embeddings.
  (only using gradients)

Step1: Storing gradient at each iteration

- During training, we need to store the per-sample gradient for each data point in the training batch.
- There are two challenges
  1. Storage
     - p: # of model parameters, T: number of iteration, B: batch size
     - Requireing $\mathcal{O}(TBp)$ disk space
  2. Efficiency
     - Computing per-sample gradients need to separate backpropagation for each $z \in \mathcal{B}_t$

     $\Rightarrow$Increasing computational cost by a factor of B

Step1: Storing gradient at each iteration

- Gradient decomposition technique [1]
  - Illustrate this technique with a simple linear layer where the output is $\boldsymbol{s} = \mathbf{aW}$
  - $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$: weight matrix, $\mathbf{a} = \left(\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(B)}\right)^{\top} \in \mathbb{R}^{B \times d_1}$: input, $\boldsymbol{s} = \left(\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(B)}\right)^{\top} \in \mathbb{R}^{B \times d_2}$: pre-activation tensor.
  - Since we can express the gradient of an individual loss $\ell^{(i)} \coloneqq \ell(\theta, z_i)$ with respect to $\mathbf{W}$ as

$$\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial \boldsymbol{s}^{(i)}} \otimes \frac{\partial \boldsymbol{s}^{(i)}}{\partial \mathbf{W}} = \frac{\partial \ell^{(i)}}{\partial \boldsymbol{s}^{(i)}} \otimes \mathbf{a}^{(i)} = \frac{\partial \ell}{\partial \boldsymbol{s}^{(i)}} \otimes \mathbf{a}^{(i)}$$

$$\left(\ell \coloneqq \textstyle\sum_{j=1}^{B} \ell^{(j)}, \otimes: Kronecker\ product\right), \left(\because \mathbf{s} = \mathbf{aW} \because j \neq i\ then\ \frac{\partial \ell^{(j)}}{\partial \boldsymbol{s}^{(i)}} = 0\right)$$

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1\mathbf{b} \\ a_2\mathbf{b} \\ \vdots \\ a_m\mathbf{b} \end{bmatrix} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \qquad \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} 1 \times 3 \\ 1 \times 4 \\ 2 \times 3 \\ 2 \times 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 6 \\ 8 \end{bmatrix}$$

$$\begin{pmatrix} \frac{\partial \ell^{(i)}}{\partial s_1^{(i)}} a_1^{(i)} & \frac{\partial \ell^{(i)}}{\partial s_1^{(i)}} a_2^{(i)} & \cdots & \frac{\partial \ell^{(i)}}{\partial s_1^{(i)}} a_{d_1}^{(i)} \\ \frac{\partial \ell^{(i)}}{\partial s_2^{(i)}} a_1^{(i)} & \frac{\partial \ell^{(i)}}{\partial s_2^{(i)}} a_2^{(i)} & \cdots & \frac{\partial \ell^{(i)}}{\partial s_2^{(i)}} a_{d_1}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \ell^{(i)}}{\partial s_{d_2}^{(i)}} a_1^{(i)} & \frac{\partial \ell^{(i)}}{\partial s_{d_2}^{(i)}} a_2^{(i)} & \cdots & \frac{\partial \ell^{(i)}}{\partial s_{d_2}^{(i)}} a_{d_1}^{(i)} \end{pmatrix}$$

[1] Wang, Jiachen T., et al. "Data shapley in one training run." arXiv preprint arXiv:2406.11011 (2024).

Step1: Storing gradient at each iteration

- Gradient decomposition technique
  - Now, for each data point z_i
    - Rather than storing full gradient vectors $\frac{\partial \ell^{(i)}}{\partial \mathbf{W}} \in \mathbb{R}^{d_1 \times d_2}$,
    - Instead store the smaller pair $\left(\mathbf{a}^{(i)}, \frac{\partial \ell}{\partial \boldsymbol{s}^{(i)}}\right) \in \mathbb{R}^{d_1 + d_2}$

    ⇒Reduces memory requirements from $\mathcal{O}(pTB)$ to $\mathcal{O}\left(\sqrt{p}TB\right)$.

    ⇒$\frac{\partial \ell}{\partial \boldsymbol{s}^{(i)}}$ is readily available during the backpropagation pass.

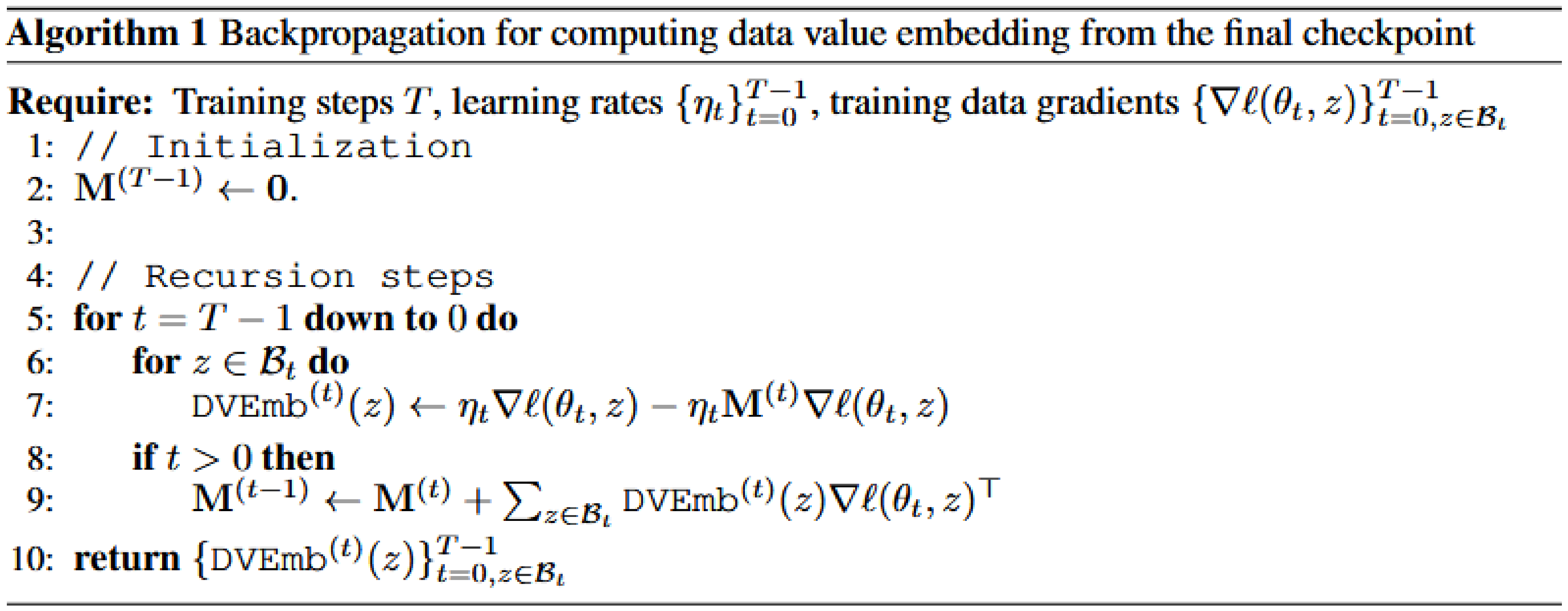
Step1: Storing gradient at each iteration

- Random projection
  - Further compress the stored gradient information.
  - Two projection matrices $\mathbf{P_a} \in \mathbb{R}^{r \times d_1}$ and $\mathbf{P_s} \in \mathbb{R}^{r \times d_2}$ to project $\mathbf{a}$ and $\frac{\partial \ell}{\partial \boldsymbol{s}}$ to lower dimensional space $\mathbb{R}^r$
  - Because kronecker product have property like $(A \otimes B)(a \otimes b) = (Aa) \otimes (Bb)$,
  - Projected gradient can then be reconstructed directly from the projected activations and output derivatives. $(\mathbf{P_a} \otimes \mathbf{P_s})\left(\mathbf{a} \otimes \frac{\partial \ell}{\partial \mathbf{s}}\right) = (\mathbf{P_a a}) \otimes \left(\mathbf{P_s} \frac{\partial \ell}{\partial \mathbf{s}}\right)$
  - This approach reduces storage needs to $\mathcal{O}(TB\tilde{p})$ where $\tilde{p}$: projected dimension, while still capturing essential gradient geometric information.

Step2: Backpropagating DVEmb

- Let $\mathrm{DVEmb}^{(t_s)}(z^*) = \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) - \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) \mathbf{M}^{(t_s)}$,
  Where $\mathbf{M}^{(\mathrm{t_s})} := \sum_{t=t_s+1}^{T-1} \left(\sum_{z \in \mathcal{B}_t} \left(\mathrm{DVEmb}^{(t)}(z) \nabla \ell(\theta_t, z)^\top \right) \right)$
- Since data value embedding is same as training gradient for the last iteration, initialize $\mathbf{M}^{(T-1)} = \mathbf{0}$.
- And for $t_s = T-1, \ldots, 0$, $\mathrm{DVEmb}^{(t_s)}(z^*)$ is recursively computed.
  1. For each $z^* \in \mathcal{B}_{t_s}$, $\mathrm{DVEmb}^{(t_s)}(z^*) = \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) - \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) \mathbf{M}^{(t_s)}$
  2. Update $\mathbf{M}$ after computing all embedding for the current iteration.
     $M^{(t_s-1)} = M^{(t_s)} + \sum_{z^* \in \mathcal{B}_{t_s}} \mathrm{DVEmb}^{(t_s)}(z^*) \nabla \ell(\theta_k, z^*)^\top$
- Now we can calculate $\mathrm{DVEmb}^{(t)}(z)$ for all data on specific iteration.
- Plus, author argues that if we assume layer wise independence, we can compute $\mathrm{DVEmb}^{(t)}(z)$ on per layer basis.

Step2: Backpropagating DVEmb



Algorithm 1 Backpropagation for computing data value embedding from the final checkpoint

```
Require: Training steps $T$, learning rates $\{\eta_t\}_{t=0}^{T-1}$, training data gradients $\{\nabla \ell(\theta_t, z)\}_{t=0, z \in \mathcal{B}_t}^{T-1}$
 1: // Initialization
 2: $\mathbf{M}^{(T-1)} \leftarrow \mathbf{0}$.
 3:
 4: // Recursion steps
 5: for $t = T-1$ down to $0$ do
 6:     for $z \in \mathcal{B}_t$ do
 7:         $\texttt{DVEmb}^{(t)}(z) \leftarrow \eta_t \nabla \ell(\theta_t, z) - \eta_t \mathbf{M}^{(t)} \nabla \ell(\theta_t, z)$
 8:     if $t > 0$ then
 9:         $\mathbf{M}^{(t-1)} \leftarrow \mathbf{M}^{(t)} + \sum_{z \in \mathcal{B}_t} \texttt{DVEmb}^{(t)}(z) \nabla \ell(\theta_t, z)^\top$
10: return $\{\texttt{DVEmb}^{(t)}(z)\}_{t=0, z \in \mathcal{B}_t}^{T-1}$
```

Step2: Backpropagating DVEmb

- Complexity of computation & memory
  - Matmuls and additions in updating $\mathrm{DVEmb}^{(t)}(z)$ and $\mathbf{M}^{(t_s)}$
    $\Rightarrow \mathcal{O}(BT\tilde{p}^2)$ flops
  - If we compute $\mathrm{DVEmb}^{(t)}(z)$ and $\mathbf{M}^{(t_s)}$ per layer under assumption,
    $\Rightarrow L\mathcal{O}(BT(\tilde{p}/L)^2) = \mathcal{O}(BT\tilde{p}^2/L)$ flops
  - For memory, $\mathbf{M}^{(t_s)}$ requires $\mathcal{O}(B\tilde{p}^2/L^2)$

  - Regular model training needs $\mathcal{O}(BTp)$ flops and $\mathcal{O}(p)$ memory.

Step2: Backpropagating DVEmb

- Complexity of storage
  - Each $\mathrm{DVEmb}^{(t)}(z)$ has dimension $\mathcal{O}(\tilde{p})$, resulting in a total storage requirement of $\mathcal{O}(BT\tilde{p})$
  - It can be large, but author argue that disk storage is relatively inexpensive in modern computing environments.

Influence checkpointing

- Backpropagation algorithm (step2) has runtime complexity of $\mathcal{O}(T)$, as it sequentially computes $\mathrm{DVEmb}^{(t)}$ for $t_s = T-1, \dots, 0$
  - This can still be costly for long training periods.

  ⇒Can't we Parallelize?

- We pick K evenly spaced training steps $0 < t_1 < t_2 < \cdots < t_K = T$
- Extended data value embedding notation $\mathrm{DVEmb}^{(t_s \to t_\ell)}(z^*)$ as the data value embedding of $z^* \in \mathcal{B}_{t_s}$ for the intermediate checkpoint θ_{t_ℓ}.
  - Note. $\mathrm{DVEmb}^{(t_s)} = \mathrm{DVEmb}^{(t_s \to T)}$

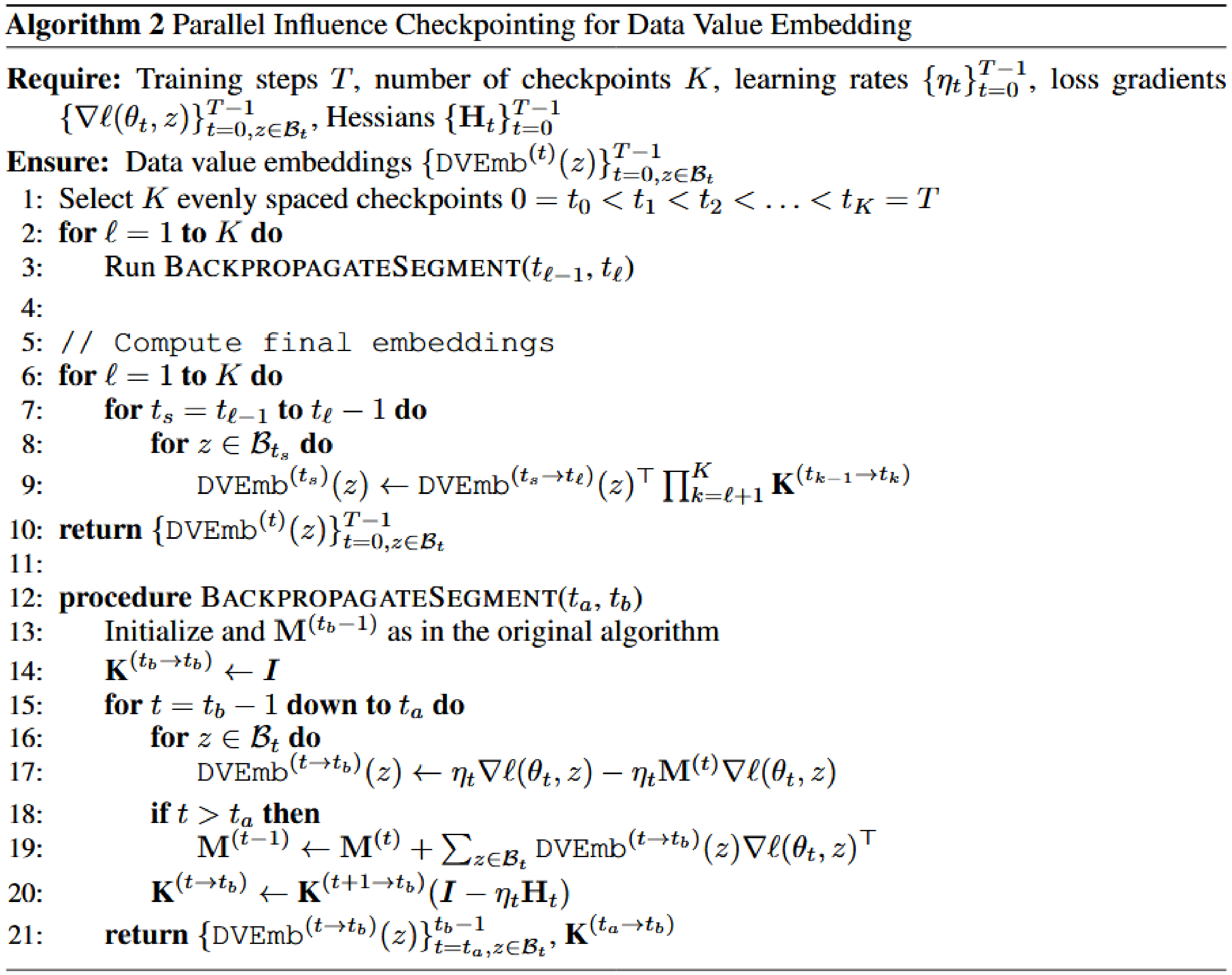
Influence checkpointing

- Letting $\mathbf{K}^{(t_a \to t_b)} := \prod_{t=t_a}^{t_b-1}(\boldsymbol{I} - \eta_t \mathbf{H}_t)$,
  $\mathrm{DVEmb}^{(t_s \to T)}$ can be computed from $\mathrm{DVEmb}^{(t_s \to t_l)}$ as follows
  $\Rightarrow \mathrm{DVEmb}^{(t_s \to T)}(z^*) = \mathrm{DVEmb}^{(t_s \to t_l)}(z^*)^\top \mathbf{K}^{(t_l \to T)}$
  $\left(\because \mathrm{DVEmb}^{(t_s \to T)} = \left[\prod_{k=t_s+1}^{T-1}(\boldsymbol{I} - \eta_k \mathbf{H}_k)\right]\eta_{t_s} \nabla \ell\big(\theta_{t_s}, z^*\big)\right)$

- And we have $\mathbf{K}^{(t_\ell \to T)} = \prod_{\ell=\ell_0+1}^{K} \mathbf{K}^{(t_{\ell-1} \to t_\ell)}$ that allows to compute $\mathrm{DVEmb}^{(t_s \to T)}$

- Plus, using this, if the intermediate checkpoints $\theta_{t_1}, \dots, \theta_{t_{K-1}}$ was saved, we can analyze how the influence changes on different intermediate checkpoints.

Influence checkpointing



Algorithm 2 Parallel Influence Checkpointing for Data Value Embedding

```
Require: Training steps $T$, number of checkpoints $K$, learning rates $\{\eta_t\}_{t=0}^{T-1}$, loss gradients
    $\{\nabla\ell(\theta_t, z)\}_{t=0, z\in\mathcal{B}_t}^{T-1}$, Hessians $\{\mathbf{H}_t\}_{t=0}^{T-1}$
Ensure: Data value embeddings $\{\mathtt{DVEmb}^{(t)}(z)\}_{t=0, z\in\mathcal{B}_t}^{T-1}$
 1: Select $K$ evenly spaced checkpoints $0 = t_0 < t_1 < t_2 < \ldots < t_K = T$
 2: for $\ell = 1$ to $K$ do
 3:     Run BackpropagateSegment($t_{\ell-1}, t_\ell$)
 4:
 5: // Compute final embeddings
 6: for $\ell = 1$ to $K$ do
 7:     for $t_s = t_{\ell-1}$ to $t_\ell - 1$ do
 8:         for $z \in \mathcal{B}_{t_s}$ do
 9:             $\mathtt{DVEmb}^{(t_s)}(z) \leftarrow \mathtt{DVEmb}^{(t_s \to t_\ell)}(z)^\top \prod_{k=\ell+1}^{K} \mathbf{K}^{(t_{k-1} \to t_k)}$
10: return $\{\mathtt{DVEmb}^{(t)}(z)\}_{t=0, z\in\mathcal{B}_t}^{T-1}$
11:
12: procedure BackpropagateSegment($t_a, t_b$)
13:     Initialize and $\mathbf{M}^{(t_b-1)}$ as in the original algorithm
14:     $\mathbf{K}^{(t_b \to t_b)} \leftarrow \boldsymbol{I}$
15:     for $t = t_b - 1$ down to $t_a$ do
16:         for $z \in \mathcal{B}_t$ do
17:             $\mathtt{DVEmb}^{(t \to t_b)}(z) \leftarrow \eta_t \nabla\ell(\theta_t, z) - \eta_t \mathbf{M}^{(t)} \nabla\ell(\theta_t, z)$
18:         if $t > t_a$ then
19:             $\mathbf{M}^{(t-1)} \leftarrow \mathbf{M}^{(t)} + \sum_{z\in\mathcal{B}_t} \mathtt{DVEmb}^{(t \to t_b)}(z) \nabla\ell(\theta_t, z)^\top$
20:         $\mathbf{K}^{(t \to t_b)} \leftarrow \mathbf{K}^{(t+1 \to t_b)}(\boldsymbol{I} - \eta_t \mathbf{H}_t)$
21:     return $\{\mathtt{DVEmb}^{(t \to t_b)}(z)\}_{t=t_a, z\in\mathcal{B}_t}^{t_b-1}$, $\mathbf{K}^{(t_a \to t_b)}$
```

Content

Fidelty evaluation

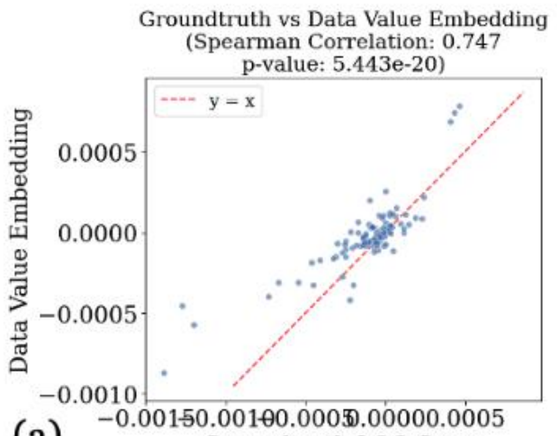
- Does it really accurate? How accurate is it?
- Computing ground truth of LOO requires retraining the model multiple times.

  ⇒Use MNIST dataset using a small MLP trained with standard SGD

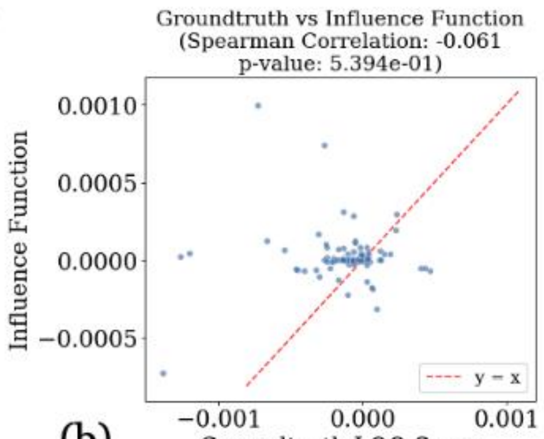
- Two settings
  1. Single epoch removal
     - Data point is excluded from training for single epoch.
     - Here, data point removed from the last epoch.
  2. All epoch removal
     - Data point is excluded in all epochs.

Fidelty evaluation

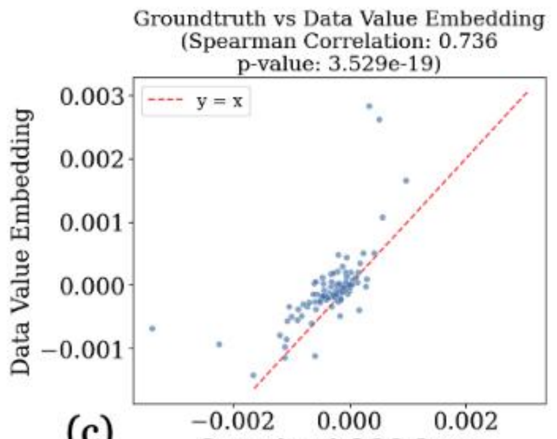
- Spearman correlation between ground-truth LOO when the MLP is trained for 3 epochs
  - (a) the data value embedding, (b) the influence function for single epoch removal.
  - (c), (d) all epoch removal.
- Shows that data value embedding has a high Spearman correlation with ground truth LOO.



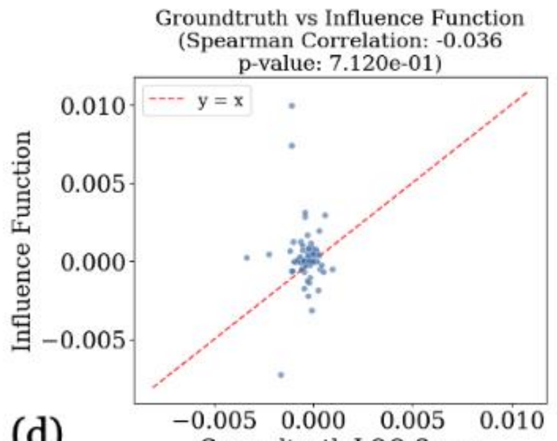
(a)



(b)



(c)



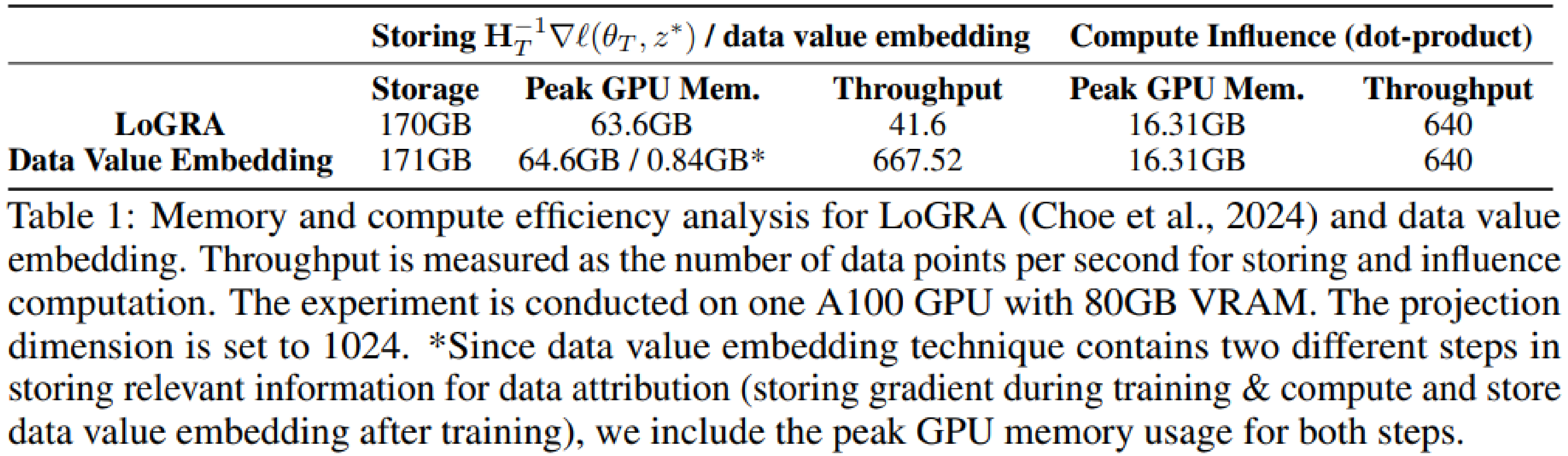
(d)

Computational efficiency

- Storage, memory, computational efficiency
  - Data value embedding vs LoGRA
    (author saids that the LoGRA is most efficient influence function)
  - LoGRA also uses random projection and stores the projected Hessian adjusted gradient $H_T^{-1}\nabla\ell(\theta_T, z^*)$, influence function can be computed via dot-product with test data gradient.
  - Computing data influence for Pytha-410M trained on 1% of the Pile dataset.

Computational efficiency

- Storage, memory, computational efficiency



	Storing $H_T^{-1}\nabla\ell(\theta_T, z^*)$ / data value embedding			Compute Influence (dot-product)	
	Storage	**Peak GPU Mem.**	**Throughput**	**Peak GPU Mem.**	**Throughput**
LoGRA	170GB	63.6GB	41.6	16.31GB	640
Data Value Embedding	171GB	64.6GB / 0.84GB*	667.52	16.31GB	640

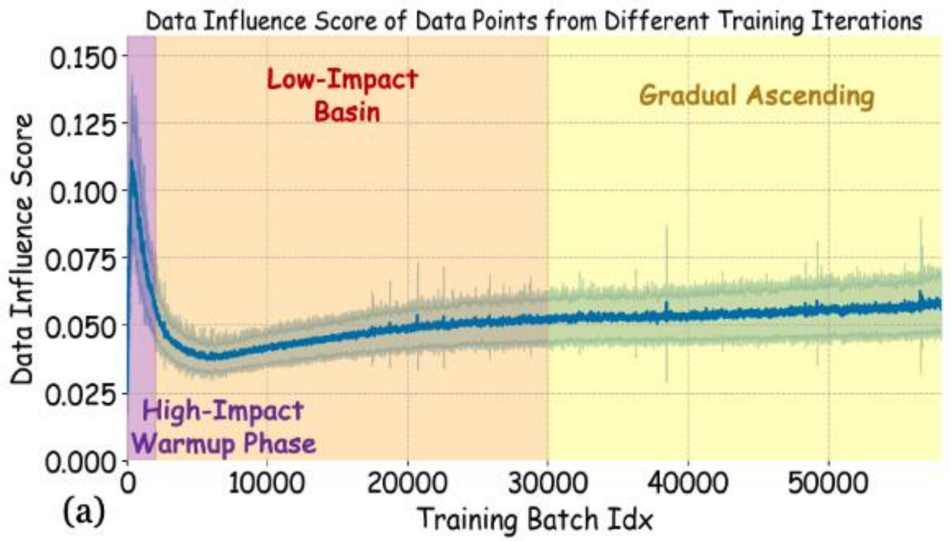
Table 1: Memory and compute efficiency analysis for LoGRA (Choe et al., 2024) and data value embedding. Throughput is measured as the number of data points per second for storing and influence computation. The experiment is conducted on one A100 GPU with 80GB VRAM. The projection dimension is set to 1024. *Since data value embedding technique contains two different steps in storing relevant information for data attribution (storing gradient during training & compute and store data value embedding after training), we include the peak GPU memory usage for both steps.

- LoGRA requires recomputing gradients for all training data on the final model θ_T, which is computationally same as one epoch of model training.
- Data value embedding operates only on projected vectors.

  ⇒63.6GB vs 0.84GB

Analyzing training dynamics

- Conducted with Pytha-410M trained on 1% of the Pile dataset.
- Can categorize in 3 distinct regimes
    1. High impact warmup phase
        - High data influence score
    2. Low impact basin
        - Low data influence score
    3. Gradual ascending
        - The later a data point participates in the training, the higher its influence score becomes.

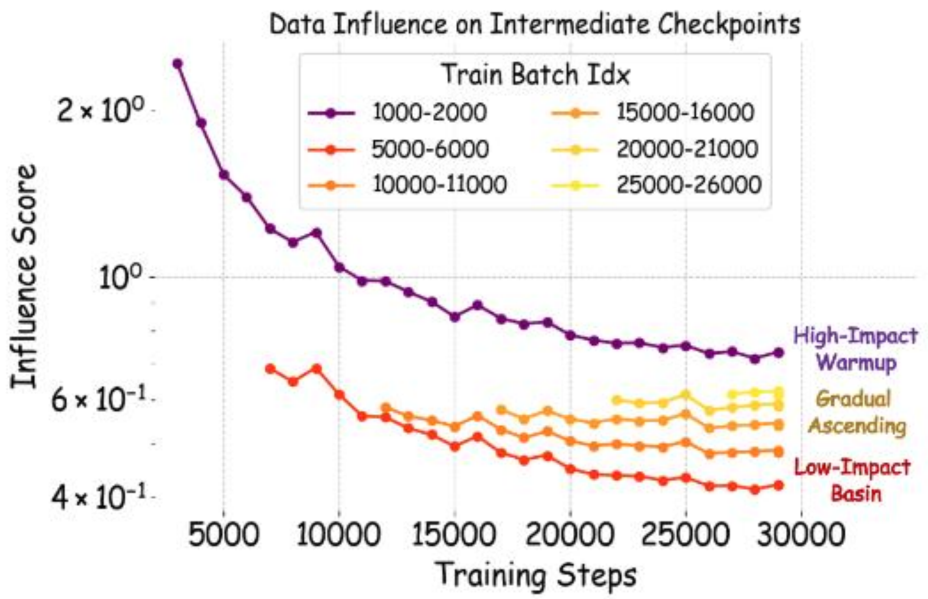


(a)

- Can find some intuition.

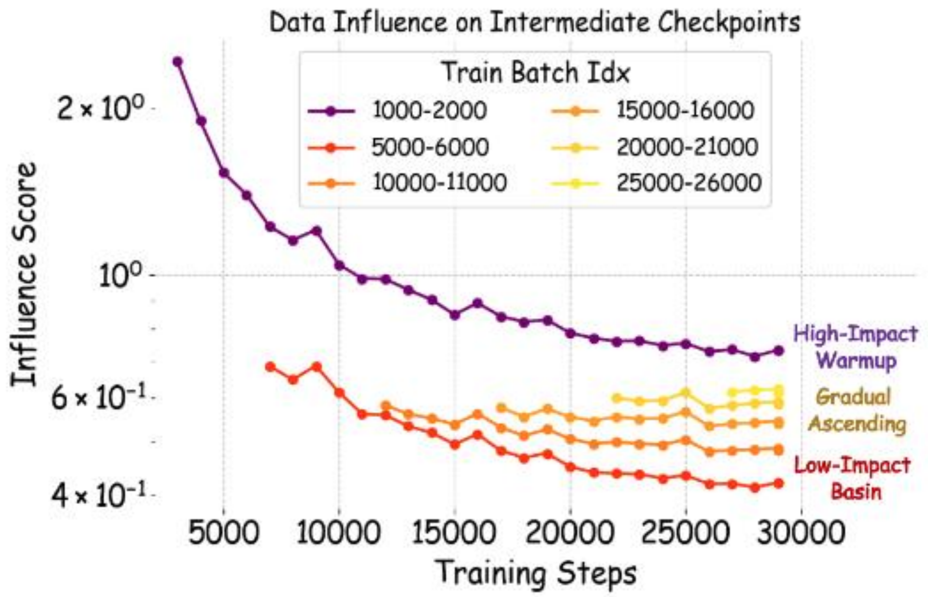
Analyzing training dynamics

1. Parameter initialization and warmup training are important for final model performance.
   - In very early stage of training, the gradient norms are large.
     - Significant parameter updates
   - Data points from the High-impact Warmup Phase are maintaining substantial influence throughout the training process, even as their immediate impact diminishes over time.



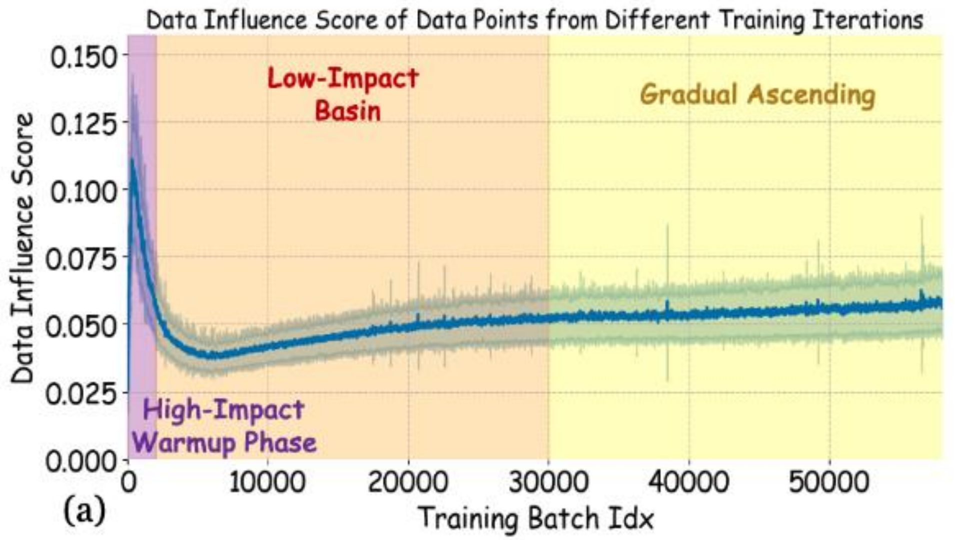
Analyzing training dynamics

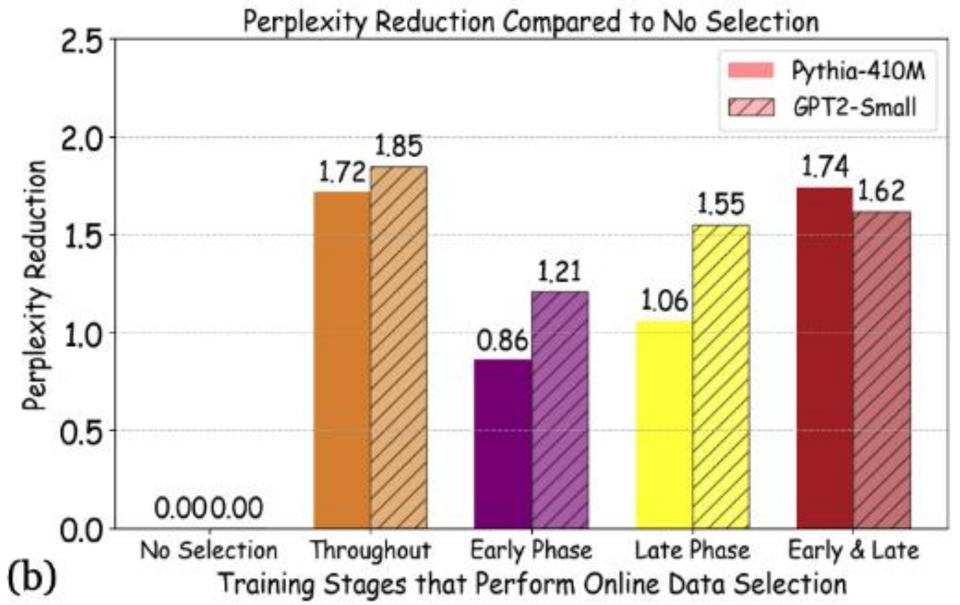
2. Influence saturation from future data
   - Training progress into a smoother loss regime, the gradient norms become relatively stable and decrease slowly.
   - Note that a data point's influence score decreases the most when future data points are similar to it $\left(\mathrm{DVEmb}^{(t_s)}(z^*) = \eta_{t_s} \nabla \ell(\theta_{t_s}, z^*) - \eta_{t_s} \sum_{t=t_s+1}^{T-1} \left(\sum_{z \in \mathcal{B}_t} \left(\nabla \ell(\theta_t, z)^\top \nabla \ell(\theta_{t_s}, z^*)\right) \mathrm{DVEmb}^{(t)}(z)\right)\right)$



Analyzing training dynamics

- How does it related to data selection strategies?
  - Following observation, data selection is most critical during the very early and later stages of training.
- Train Pythia-410M on Pile with different online data selection strategies.
- Figure shows that performing data selection only in first 2000 iteration and after 20000 iterations closely matches the performance when data selection is performed in all iterations.





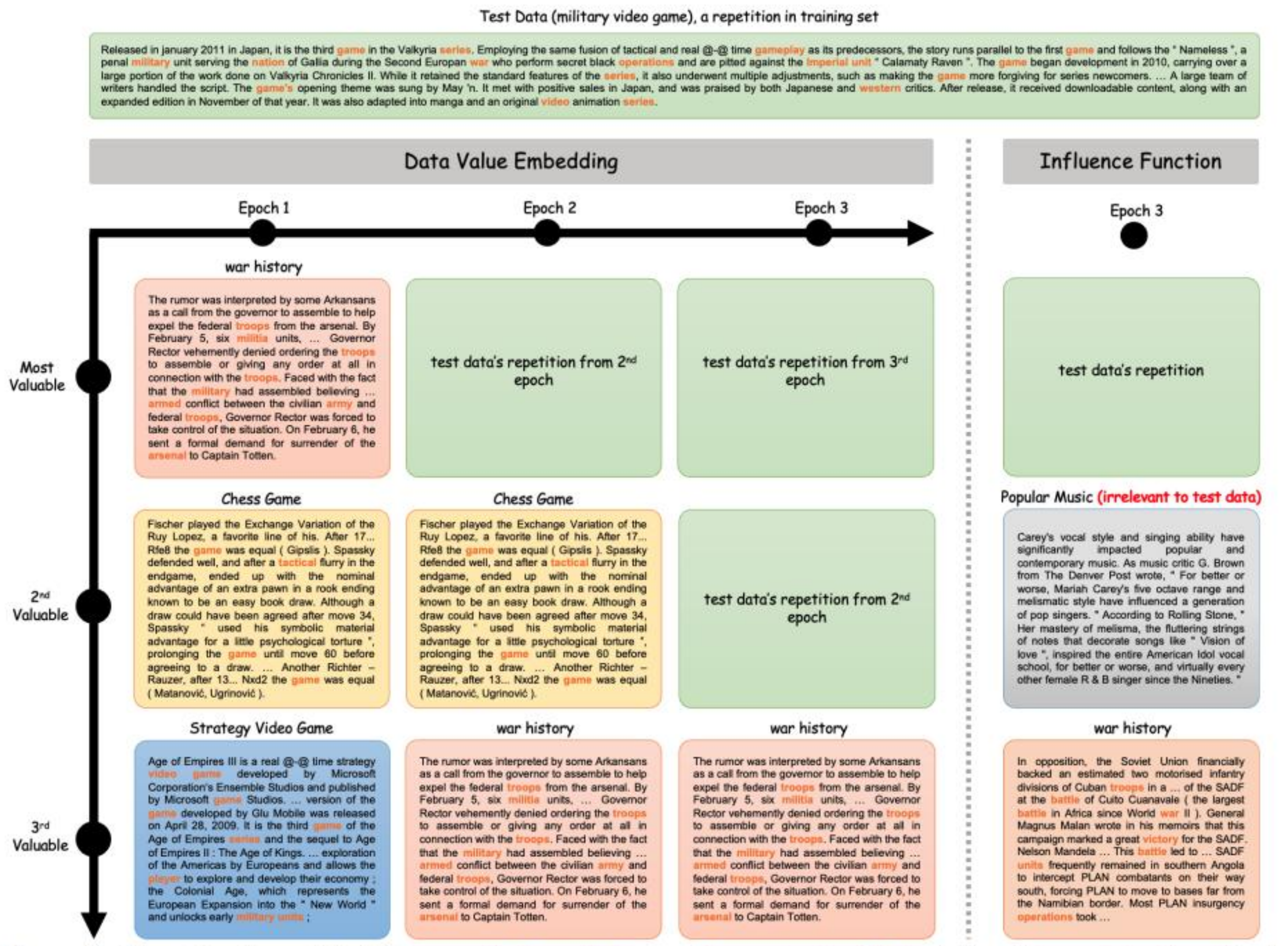
Qualitative evaluation

- What is the most valuable data points with a test data point $z^{(val)}$ identified by data value embedding?
  - Setting $z^{(val)}$ as identical to one of the training data points.

    ⇒Making most similar data point "self-influence"
    (should be highest among all training points)

- Training GPT-2 on Wikitext-103 over three epochs, where the test data is about "military video game"

Qualitative evaluation


Test Data (military video game), a repetition in training set
Data Value Embedding
Influence Function
Epoch 1
Epoch 2
Epoch 3
Epoch 3
Most Valuable
2nd Valuable
3rd Valuable
war history
test data's repetition from 2nd epoch
test data's repetition from 3rd epoch
test data's repetition
Chess Game
Chess Game
test data's repetition from 2nd epoch
Popular Music (irrelevant to test data)
Strategy Video Game
war history
war history
war history

Released in january 2011 in Japan, it is the third game in the Valkyria series. Employing the same fusion of tactical and real @-@ time gameplay as its predecessors, the story runs parallel to the first game and follows the " Nameless ", a penal military unit serving the nation of Gallia during the Second Europan war who perform secret black operations and are pitted against the Imperial unit " Calamaty Raven ". The game began development in 2010, carrying over a large portion of the work done on Valkyria Chronicles II. While it retained the standard features of the series, it also underwent multiple adjustments, such as making the game more forgiving for series newcomers. ... A large team of writers handled the script. The game's opening theme was sung by May 'n. It met with positive sales in Japan, and was praised by both Japanese and western critics. After release, it received downloadable content, along with an expanded edition in November of that year. It was also adapted into manga and an original video animation series.

The rumor was interpreted by some Arkansans as a call from the governor to assemble to help expel the federal troops from the arsenal. By February 5, six militia units, ... Governor Rector vehemently denied ordering the troops to assemble or giving any order at all in connection with the troops. Faced with the fact that the military had assembled believing ... armed conflict between the civilian army and federal troops, Governor Rector was forced to take control of the situation. On February 6, he sent a formal demand for surrender of the arsenal to Captain Totten.

Fischer played the Exchange Variation of the Ruy Lopez, a favorite line of his. After 17... Rfe8 the game was equal (Gipslis). Spassky defended well, and after a tactical flurry in the endgame, ended up with the nominal advantage of an extra pawn in a rook ending known to be an easy book draw. Although a draw could have been agreed after move 34, Spassky " used his symbolic material advantage for a little psychological torture ", prolonging the game until move 60 before agreeing to a draw. ... Another Richter – Rauzer, after 13... Nxd2 the game was equal (Matanović, Ugrinović).

Fischer played the Exchange Variation of the Ruy Lopez, a favorite line of his. After 17... Rfe8 the game was equal (Gipslis). Spassky defended well, and after a tactical flurry in the endgame, ended up with the nominal advantage of an extra pawn in a rook ending known to be an easy book draw. Although a draw could have been agreed after move 34, Spassky " used his symbolic material advantage for a little psychological torture ", prolonging the game until move 60 before agreeing to a draw. ... Another Richter – Rauzer, after 13... Nxd2 the game was equal (Matanović, Ugrinović).

Carey's vocal style and singing ability have significantly impacted popular and contemporary music. As music critic G. Brown from The Denver Post wrote, " For better or worse, Mariah Carey's five octave range and melismatic style have influenced a generation of pop singers. " According to Rolling Stone, " Her mastery of melisma, the fluttering strings of notes that decorate songs like " Vision of love ", inspired the entire American Idol vocal school, for better or worse, and virtually every other female R & B singer since the Nineties. "

Age of Empires III is a real @-@ time strategy video game developed by Microsoft Corporation's Ensemble Studios and published by Microsoft game Studios. ... version of the game developed by Glu Mobile was released on April 28, 2009. It is the third game of the Age of Empires series and the sequel to Age of Empires II : The Age of Kings. ... exploration of the Americas by Europeans and allows the player to explore and develop their economy ; the Colonial Age, which represents the European Expansion into the " New World " and unlocks early military units ;

The rumor was interpreted by some Arkansans as a call from the governor to assemble to help expel the federal troops from the arsenal. By February 5, six militia units, ... Governor Rector vehemently denied ordering the troops to assemble or giving any order at all in connection with the troops. Faced with the fact that the military had assembled believing ... armed conflict between the civilian army and federal troops, Governor Rector was forced to take control of the situation. On February 6, he sent a formal demand for surrender of the arsenal to Captain Totten.

The rumor was interpreted by some Arkansans as a call from the governor to assemble to help expel the federal troops from the arsenal. By February 5, six militia units, ... Governor Rector vehemently denied ordering the troops to assemble or giving any order at all in connection with the troops. Faced with the fact that the military had assembled believing ... armed conflict between the civilian army and federal troops, Governor Rector was forced to take control of the situation. On February 6, he sent a formal demand for surrender of the arsenal to Captain Totten.

In opposition, the Soviet Union financially backed an estimated two motorised infantry divisions of Cuban troops in a ... of the SADF at the battle of Cuito Cuanavale (the largest battle in Africa since World war II). General Magnus Malan wrote in his memoirs that this campaign marked a great victory for the SADF. Nelson Mandela ... This battle led to ... SADF units frequently remained in southern Angola to intercept PLAN combatants on their way south, forcing PLAN to move to bases far from the Namibian border. Most PLAN insurgency operations took ...

Figure 5: Visualization of (left) the evolution of the top-3 most valuable training data points identified by data value embedding throughout 3 training epochs and (right) the top-3 most valuable training data points identified by influence function. We use GPT-2 trained on Wikitext-103, with the test point being a repetition of a training data point related to a military video game. The common words between the test and training data are highlighted in orange.

Content

Conclusion and Limitations

- Introduced data value embedding
  - No need to retrain the model.
  - Efficient & Accurate than influence function using approximation.
  - Capturing temporal dynamics of training.

- Limitations
  - Tailored for SGD.
  - Tailored for cross entropy loss.
  - Doesn’t think about I/O complexity.

Thank you

hyunwoojung@postech.ac.kr

Taylor expansion

- Approximate a function to polinomial near specific point with function value and derivatives.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

- First order Taylor expansion
  - $f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a)$

Spearman correlation

- Evaluating monotonic relationship.
  - If one variable increase then other variable increases.
  - If one variable increase then other variable decreases.

- Use rank to calculate correlation.
  - $\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$
  - Where $d_i = rank(x_i) - rank(y_i)$