Coresets cho Huấn luyện Mô hình Học máy Hiệu quả về Dữ liệu

(Dựa trên bài báo "Coresets for Data-efficient Training of Machine Learning Models" của Mirzasoleiman, Bilmes và Leskovec, 2020)

Người Trình Bày: Phạm Thanh Hiếu

Hanoi, June 2025

OUTLINE

Đặt vấn đề

- **Phát biểu bài toán gốc:** Bài toán tối ưu thường gặp trong huấn luyện mô hình học máy:

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w), \tag{1}$$

trong đó,

  - $f_i(w)$: hàm mất mát trên điểm dữ liệu thứ i,
  - $f(w)$: tổng mất mát toàn bộ.

- **Phương pháp Gradient Descent (GD)** (Cauchy, A.-L., 1847) là phương pháp cập nhật tham số w theo hướng giảm hàm mất mát

$$w_{k+1} = w_k - \alpha \nabla f(w_k).$$

Nếu tập dữ liệu V có hàng triệu điểm, việc tính $\nabla f(w)$ rất "tốn kém" về **thời gian và chi phí tính toán**.

Đặt vấn đề

- **Bối cảnh:**
  - **IG (Incremental Gradient)** (Bertsekas, D.P., 1996) là các phương pháp huấn luyện mô hình bằng cách ước lượng **gradient từng phần**, thay vì tính toàn bộ gradient một lúc. Mỗi bước lặp chỉ dùng một (hoặc một nhóm nhỏ/mini-batch) điểm dữ liệu để cập nhật mô hình $\rightarrow$ **tiết kiệm thời gian**, đặc biệt hữu ích với tập dữ liệu lớn.

Đặt vấn đề

- **Bối cảnh:**
  - Các phương pháp **Incremental Gradient – IG** như SGD và các biến thể tăng tốc của nó (bao gồm SGD với momentum, Adagrad, Adam, SAGA, và SVRG) ước lượng gradient bằng cách lặp lại trên các tập con ngẫu nhiên (mini-batch) của dữ liệu huấn luyện.

So sánh các thuật toán IG

Thuật toán	Công thức	Đặc điểm - Ý nghĩa
SGD (Stochastic Gradient Descent) (Robbins & Monro, 1951)	$w_{t+1} = w_t - \eta \nabla f_i(w_t)$	Cập nhật tham số theo gradient của một điểm hoặc mini-batch. Tiết kiệm thời gian nhưng nhiễu cao. Dễ cài đặt, dao động mạnh.
SGD with Momentum (Quian, 1999)	$v_t = \mu v_{t-1} + \eta \nabla f_i(w_t),\ w_{t+1} = w_t - v_t$	Giữ một phần gradient cũ để tăng tốc hội tụ. Giảm dao động, hội tụ nhanh hơn. Tốt cho địa hình gập ghềnh.
Adagrad (Adaptive Gradient) (Duchi et al., 2011)	$G_t = G_{t-1} + (\nabla f_i(w_t))^2,\ w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t+\epsilon}} \nabla f_i(w_t)$	Tự điều chỉnh bước học theo từng tham số. Tốt khi gradient thưa thớt. Bước học giảm dần.
Adam (Adaptive Moment Estimation) (Kingma & Ba, 2014)	m_t, v_t theo trung bình động bậc 1 và bậc 2, $w_{t+1} = w_t - \eta \frac{m_t}{\sqrt{v_t}+\epsilon}$	Kết hợp Adagrad và momentum. Hiệu quả trong DNN. Hội tụ nhanh và ổn định.
SAGA (Stochastic Average Gradient) (Defazio et al., 2014)	$w_{t+1} = w_t - \eta(\nabla f_i(w_t) - \nabla f_i(\phi_i) + \bar{g})$	Dùng lại gradient cũ để giảm phương sai. Cần lưu thêm vector gradient. Hội tụ nhanh hơn SGD.
SVRG (Stochastic Variance Reduced Gradient) (Johnson & Zhang, 2013)	$w_{t+1} = w_t - \eta(\nabla f_i(w_t) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w}))$	Khử nhiễu bằng gradient tham chiếu. Phải tính gradient toàn phần định kỳ. Hội tụ tốt hơn.

Cách làm này giúp tạo ra một **ước lượng không chệch của gradient toàn phần** ($E_i[\nabla f_i(w)] = \nabla f(w)$), tuy nhiên việc chọn ngẫu nhiên dữ liệu cũng làm xuất hiện nhiễu (phương sai) trong ước lượng gradient, tức là $Var[\nabla f_i(w)] = E[\|\nabla f_i(w) - \nabla f(w)\|^2] > 0$.

Chính vì vậy, các phương pháp gradient ngẫu nhiên thường hội tụ chậm.

Đặt vấn đề

- **Câu hỏi mở:**
  - **Vấn đề:** Có thể huấn luyện chỉ trên tập con nhỏ $S \subset V$ (toàn bộ dữ liệu huấn luyện) mà vẫn **(xấp xỉ) hội tụ đến nghiệm tối ưu toàn cục**?
  - **Lợi ích:** Tăng tốc theo tỉ lệ $|V|/|S|$ (tỉ lệ này có thể rất lớn nếu $S \ll V$) mỗi epoch nếu chọn được S hiệu quả.

Thách thức chính khi chọn tập con S

1. **Thiếu nguyên tắc lựa chọn rõ ràng:** Chẳng hạn, chọn các điểm gần ranh giới phân loại giúp mô hình tinh chỉnh ranh giới, trong khi chọn các điểm đa dạng giúp mô hình nắm bắt phân bố dữ liệu tốt hơn.
2. **Yêu cầu tốc độ xử lý:** Việc tìm S cần đủ nhanh để không tốn nhiều thời gian hơn cả quá trình tối ưu, nếu không sẽ không đạt được lợi ích tăng tốc.
3. **Cần xác định thêm bước nhảy gradient:** Việc chỉ chọn điểm chưa đủ; cần chọn bước nhảy phù hợp cho từng điểm trong S vì nó ảnh hưởng đến tốc độ hội tụ.
4. **Đảm bảo lý thuyết và hội tụ:** Dù phương pháp có thể hiệu quả trên một số bộ dữ liệu, cần có hiểu biết lý thuyết và đảm bảo hội tụ toán học.

Giới thiệu CRAIG

Mục tiêu chính: Phát triển phương pháp **CRAIG** – một phương pháp chọn coreset (tập con dữ liệu huấn luyện có trọng số) để tăng tốc huấn luyện cho các mô hình học máy lớn sử dụng các phương pháp Gradient gia tăng (IG). CRAIG chọn coreset $S \subset V$ sao cho

$$\sum_{j \in S} \gamma_j \nabla f_j(w) \approx \sum_{i \in V} \nabla f_i(w).$$

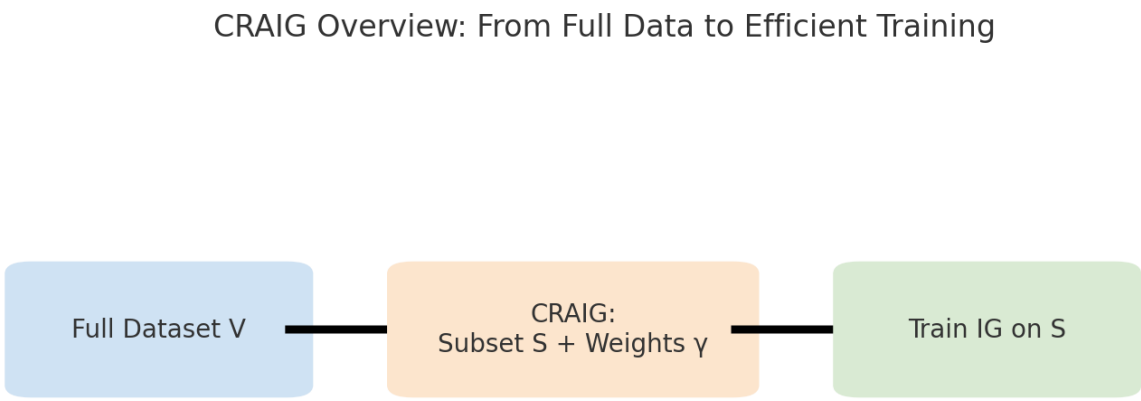
Mục tiêu: chỉ cần học từ "người đại diện", không cần "toàn dân".

Giới thiệu CRAIG

Ý tưởng chính:

- CRAIG chọn một tập con $S \subset V$ sao cho **xấp xỉ tốt nhất gradient trên toàn bộ** tập V. Cụ thể, tập S được chọn sao cho giảm được giới hạn trên của sai số ước lượng gradient toàn phần.
- CRAIG xây dựng một **hàm mục tiêu submodular** dựa trên khoảng cách gradient, và chứng minh rằng bài toán chọn S chính là bài toán **tối đa hóa một hàm submodular dạng facility location**.
- Điều này cho phép áp dụng thuật toán tham lam (greedy) hiệu quả để chọn coreset.

Sơ đồ hoạt động CRAIG



- Input: Tập dữ liệu đầy đủ V
- CRAIG: Chọn coreset $S \subset V$, tính trọng số γ_j
- Training: Huấn luyện trên S với IG
- Output: Mô hình hiệu quả, tốc độ nhanh hơn $\approx \frac{|V|}{|S|}$

CRAIG – Định nghĩa Bài toán Chọn Coreset

- **Mục tiêu của CRAIG:** Tìm một tập con có trọng số $S \subseteq V$ và các trọng số $\gamma_j \geq 0$ sao cho ước lượng gradient từ coreset xấp xỉ tốt nhất gradient của toàn bộ tập dữ liệu, *trên toàn bộ không gian tham số* $\mathcal{W}$.
- **Phát biểu bài toán:**

$$S^* = \arg \min_{S \subseteq V, \gamma_j \geq 0} |S|$$
$$\text{s.t.} \quad \max_{w \in \mathcal{W}} \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \varepsilon \tag{2}$$

(Tìm tập S nhỏ nhất sao cho sai số tối đa không vượt quá ε)

- **Thách thức với Công thức 2:**
  1. **Tính toán** $\max_w$**:** Việc tính toán gradient trên toàn bộ không gian $\mathcal{W}$ là không khả thi.
  2. **Tìm** S^* **tối ưu:** Duyệt qua tất cả các tập con là bài toán NP-hard.

Giải quyết Thách thức 1 – Giới hạn trên cho Sai số Ước lượng

- **Ý tưởng:** Thay vì tính $\max_w$, CRAIG tìm một *giới hạn trên* cho sai số.
- **Gán điểm dữ liệu:** Với coreset S và một w cụ thể, mỗi điểm $i \in V$ được gán cho điểm $j = s_w(i) \in S$ có gradient gần nhất tại w.
- **Trọng số:** γ_j là số lượng điểm $i \in V$ được gán cho $j \in S$.
- **Giới hạn trên sai số:**

$$\left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \sum_{i \in V} \left\| \nabla f_i(w) - \nabla f_{s_w(i)}(w) \right\| \tag{3}$$

- **Chuyển hóa mục tiêu:**

$$\min_S \sum_{i \in V} \min_{j \in S} \|\nabla f_i(w) - \nabla f_j(w)\| \tag{4}$$

Giải quyết Thách thức 1 – Giới hạn trên cho Sai số Ước lượng

- **Định nghĩa khoảng cách $d_{\{ij\}}$:** Để loại bỏ sự phụ thuộc vào w (đặc biệt trong bài toán lồi), định nghĩa khoảng cách gradient "worst-case":

$$d_{\{ij\}} = \max_{w \in \mathcal{W}} \|\nabla f_i(w) - \nabla f_j(w)\| \tag{5}$$

- **Bài toán cực tiểu hóa:**

$$S^* = \arg\min_{S \subseteq V} |S| \text{ s.t. } L(S) = \sum_{i \in V} \min_{j \in S} d_{\{ij\}} \leq \varepsilon. \tag{6}$$

- **Ý nghĩa:** Tìm tập con nhỏ nhất sao cho khoảng cách gradient "worst-case" từ mỗi điểm trong V đến coreset không vượt quá ngưỡng ε.

Giải quyết Thách thức 1 – Tính toán khoảng cách $d_{\{ij\}}$ (Convex Problems)

- **Bài toán lồi (Convex Problems – ví dụ: Logistic Regression, SVM):**
  - $d_{\{ij\}}$ có thể được giới hạn trên hiệu quả bởi một đại lượng chỉ phụ thuộc vào dữ liệu x_i, x_j:

$$d_{\{ij\}} \leq \text{const} \cdot \|x_i - x_j\| \tag{7}$$

    nếu các điểm thuộc cùng lớp, $|y_i - y_j| \approx 0$.
  - Nếu hai điểm không cùng nhãn ($y_i \neq y_j$), ta có thể ước lượng:
    1. **Phương án "nhãn hóa" dữ liệu**: $x_i' = y_i x_i, x_j' = y_j x_j$, rồi dùng $d_{\{i,j\}} = \|x_i' - x_j'\|(1 + L_\sigma \|x_j'\| B)$ với $L_\sigma = \sup_z |\sigma'(z)| = 1/4, \sigma(z) = 1/(1 + e^{-z})$ là hàm sigmoid và $\|w\| \leq B$; hoặc đơn giản hơn chỉ để $d_{\{i,j\}} \leq C_2 \|x_i' - x_j'\|$.
    2. **Phương án Lipschitz tổng quát:** Nhiều hàm mất mát lồi (như logistic, SVM với hinge loss, v.v.) đều là L-smooth (có gradient Lipschitz) trên miền $\mathcal{W}$: $\|\nabla f_i(w) - \nabla f_j(w)\| \leq L_{max} \|(x_i, y_i) - (x_j, y_j)\|$, với $L_{max} = \max_{i,j} L_{i,j}$. Ta có thể giới hạn

$$d_{\{i,j\}} \leq L_{max} \|(x_i, y_i) - (x_j, y_j)\| = L_{max} \sqrt{\|x_i - x_j\|^2 + (y_i - y_j)^2}.$$

Giải quyết Thách thức 1 – Tính toán khoảng cách $d_{\{ij\}}$ (Non-convex DNNs)

- **Bài toán không lồi (Non-convex Problems – ví dụ: Deep Neural Networks):**
  - Giới hạn trên của $d_{\{ij\}}$ phụ thuộc vào tham số w hiện tại.
- **Giải pháp của CRAIG:**
  - Xấp xỉ $\|\nabla f_i(w) - \nabla f_j(w)\|$ bằng cách tập trung vào sự khác biệt của gradient hàm mất mát tại **đầu vào của lớp cuối cùng**.
  - Định nghĩa khoảng cách:

$$d_{\{ij\}}(w) \approx c_1 \cdot \left\| G_i^{\nabla \ell(L)}(w) - G_j^{\nabla \ell(L)}(w) \right\| + c_2 \qquad (8)$$

  - Coreset S cần được **chọn lại định kỳ** trong quá trình huấn luyện DNN.
  - Chi phí chọn lại nhỏ so với chi phí lan truyền ngược.

Giải quyết Thách thức 2 – Thuật toán CRAIG – Tối ưu hoá Submodular

- **Bài toán:** Tìm tập S nhỏ nhất sao cho tổng khoảng cách đến điểm gán gần nhất trong S nhỏ hơn ε. Đây là một dạng của bài toán Set Cover.
- **Chuyển đổi thành bài toán Tối đa hóa Submodular:**
  - Thay vì tìm S nhỏ nhất để đạt $L(S) = \sum_{i \in V} \min_{j \in S} d_{i,j} \leq \varepsilon$, ta cố định kích thước S và tối đa hóa mức độ "phủ sóng" (giảm thiểu $L(S)$).
  - Định nghĩa hàm lợi ích (Facility Location):

$$F(S) = L(\emptyset) - L(S) = \sum_{i \in V} \left(d_{\{i,0\}} - \min_{j \in S} d_{\{ij\}} \right) \tag{9}$$

    (trong đó $d_{\{i,0\}}$ là một hằng số hoặc khoảng cách đến một "điểm ảo").
  - Hàm $F(S)$ là một hàm đơn điệu và submodular.
- **Hàm Submodular – "Lợi ích Giảm dần":**
  - Thêm một điểm dữ liệu mới vào một coreset nhỏ sẽ mang lại *lợi ích lớn hơn* (giảm $L(S)$) so với khi thêm điểm đó vào coreset lớn hơn.

Giải quyết Thách thức 2 – Thuật toán CRAIG – Greedy - Bảo đảm chất lượng

Thuật toán Tham lam (Greedy) cho CRAIG:

1. Bắt đầu với $S = \emptyset$.
2. Lặp lại cho đến khi $L(S) \leq \varepsilon$ (hoặc đạt kích thước mong muốn r):
   - Chọn điểm

$$j^* = \arg\max_{j \in V \setminus S} [F(S \cup \{j\}) - F(S)]$$

     (điểm mang lại lợi ích biên lớn nhất)
   - Thêm j^* vào S.
3. Tính trọng số γ_j cho mỗi $j \in S$ (số điểm $i \in V$ gán gần nhất theo $d_{\{ij\}}$).

Đảm bảo: Thuật toán tham lam cung cấp giải pháp xấp xỉ tốt cho bài toán submodular.

Phân tích Tốc độ Hội tụ của CRAIG

- **Thiết lập:** Sử dụng một phương pháp Incremental Gradient (IG) tổng quát trên coreset S (có trọng số γ_j).
- **Giả định:** Coreset S ước lượng gradient đầy đủ với sai số tối đa là:

$$\max_w \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \varepsilon \tag{10}$$

- **Định lý 1 (Hàm lồi mạnh – Strongly Convex):** IG trên coreset S với bước học $\alpha_k = \mathcal{O}(1/k^\tau)$ (với $\tau \in (0, 1]$) hội tụ đến một vùng lân cận của điểm tối ưu w^*.
- **Tốc độ hội tụ:**
  - Nếu $\tau = 1$:

$$\|w_k - w^*\|^2 = \mathcal{O}(1/k) \quad \text{(sau một số epoch nhất định)} \tag{11}$$

  - Nếu $\tau < 1$: $\|w_k - w^*\|^2 \leq \cdots + \dfrac{2\varepsilon R}{\mu^2}$ (khi $k \to \infty$)

Phân tích Tốc độ Hội tụ của CRAIG

- **Quan trọng:** Tốc độ hội tụ *tương tự* như IG trên toàn bộ dữ liệu V, nhưng sai số hội tụ có thêm một phần:

$$\frac{2\varepsilon R}{\mu^2}$$

do sai số xấp xỉ gradient từ coreset.

- **Tăng tốc độ:** CRAIG đạt được tốc độ lý thuyết theo tỉ lệ:

$$\frac{|V|}{|S|}$$

trong mỗi epoch.

- **Định lý 2 (Hàm lồi mạnh và trơn – Smooth):**
  - Tốc độ hội tụ nhanh hơn: $\|w_k - w^*\|^2 = \mathcal{O}(1/k^\tau)$ nếu $\tau > 0$, sau một số epoch nhất định.
  - Cũng hội tụ đến vùng lân cận $\frac{2\varepsilon R}{\mu}$.

Tổng kết tốc độ hội tụ

Giả thiết	**Tốc độ hội tụ**	**Vùng hội tụ**
Lồi mạnh	$\mathcal{O}(1/k)$	$\frac{2R\varepsilon}{\mu^2}$
Lồi mạnh + gradient trơn	$\mathcal{O}(1/k^\tau)$	$\frac{2\varepsilon}{\mu}$

Table: So sánh các tốc độ hội tụ của CRAIG

Thực nghiệm

Câu hỏi Nghiên cứu Thực nghiệm:

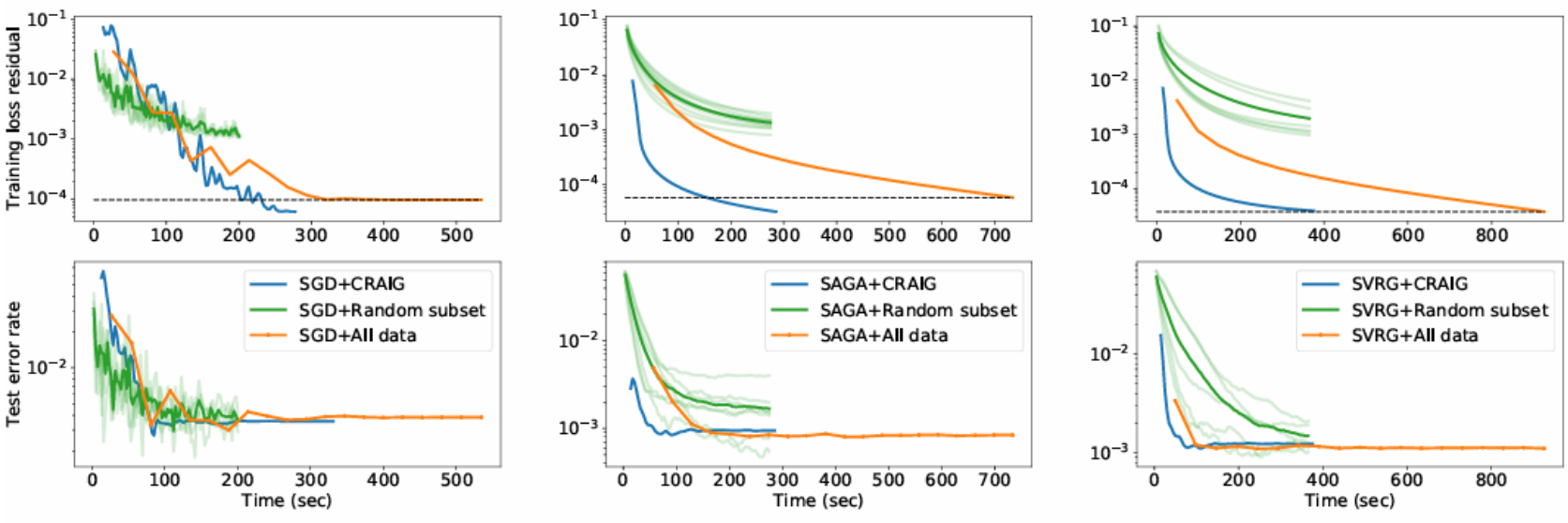
1. Mất mát và độ chính xác của IG trên coreset CRAIG so với trên toàn bộ dữ liệu như thế nào?
2. Coreset có thể nhỏ đến mức nào mà vẫn duy trì hiệu suất tốt?
3. CRAIG có mở rộng tốt cho dữ liệu lớn và bài toán không lồi không?

Thiết lập:

- **Bài toán Convex:** Logistic Regression (SGD, SAGA, SVRG) trên Covtype, ljcnn1.
- **Bài toán Non-Convex (DNNs):** Mạng 2 lớp trên MNIST (SGD), ResNet-20 trên CIFAR10 (SGD).
- **So sánh với:** Huấn luyện trên toàn bộ dữ liệu, và trên tập con ngẫu nhiên cùng kích thước.
- **Đo lường:** Thời gian thực tế (wall-clock time) bao gồm cả thời gian chọn coreset.

Thực nghiệm trên Covtype – Tăng tốc và Gradient

- **Figure 1 – Loss residual & error rate:**
  - CRAIG (10%) đạt kết quả tương đương toàn bộ dữ liệu với tốc độ nhanh hơn 3x so với SGD, SVRG, SAGA.
  - So sánh: CRAIG (xanh dương), random 10% (xanh lá), toàn bộ dữ liệu (cam).

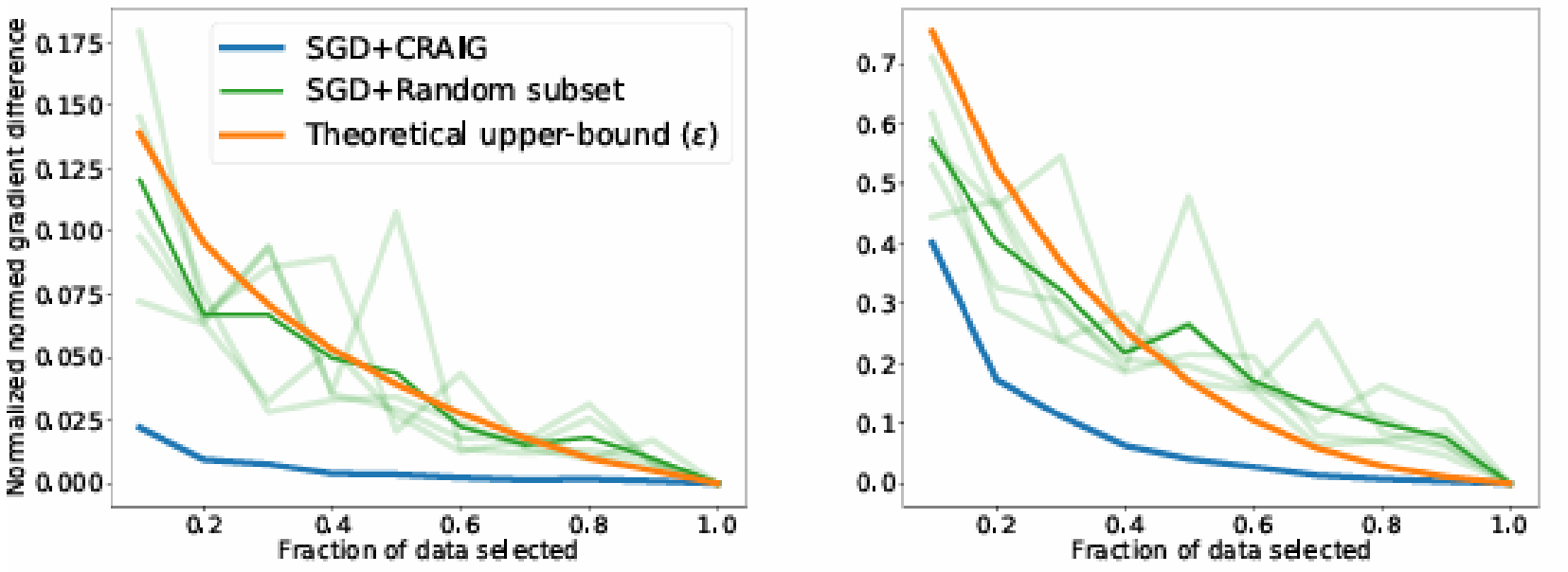


(a) SGD (b) SAGA (c) SVRG

Figure 1: Loss & Error Rate (Covtype)

Thực nghiệm trên Covtype – Tăng tốc và Gradient

- **Figure 2 – Gradient xấp xỉ:**
  - So sánh sai khác gradient giữa CRAIG vs. random subset và giới hạn lý thuyết (Eq. 8).
  - CRAIG chính xác hơn hẳn tập chọn ngẫu nhiên.



(a) Covtype (b) Ijcnn1

Figure 2: Gradient Approximation

Ảnh hưởng của Kích thước Coreset – Ijcnn1

- CRAIG được áp dụng cho các coreset kích thước 10%, 20%, ..., 90%.
- So sánh với các tập chọn ngẫu nhiên cùng kích thước.
- CRAIG với coreset 30% đạt tăng tốc **5.6x** so với huấn luyện toàn bộ dữ liệu.

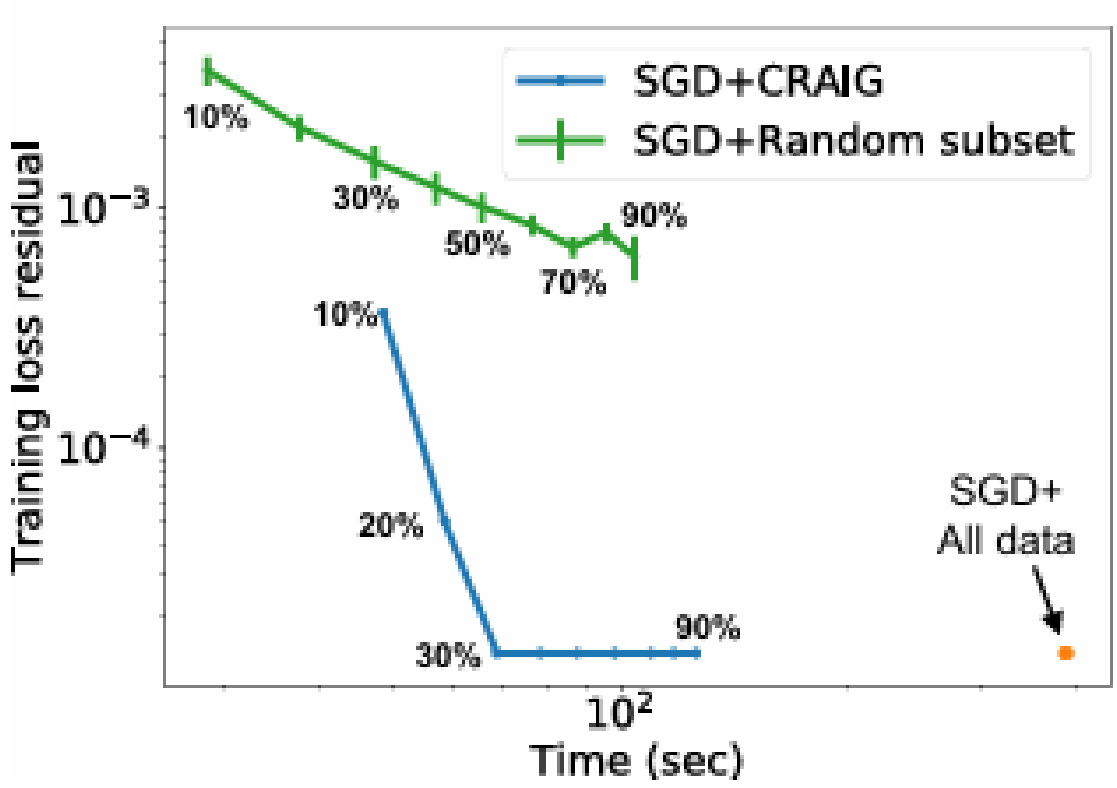


Figure 3: Loss Residual vs. Subset Size (Ijcnn1)

Thực nghiệm trên MNIST – 2-layer NN

- So sánh test accuracy và training loss khi huấn luyện bằng CRAIG vs. random trên tập con.
- CRAIG đạt **tăng tốc 2x – 3x** và **hiệu quả khái quát hóa tốt hơn**.

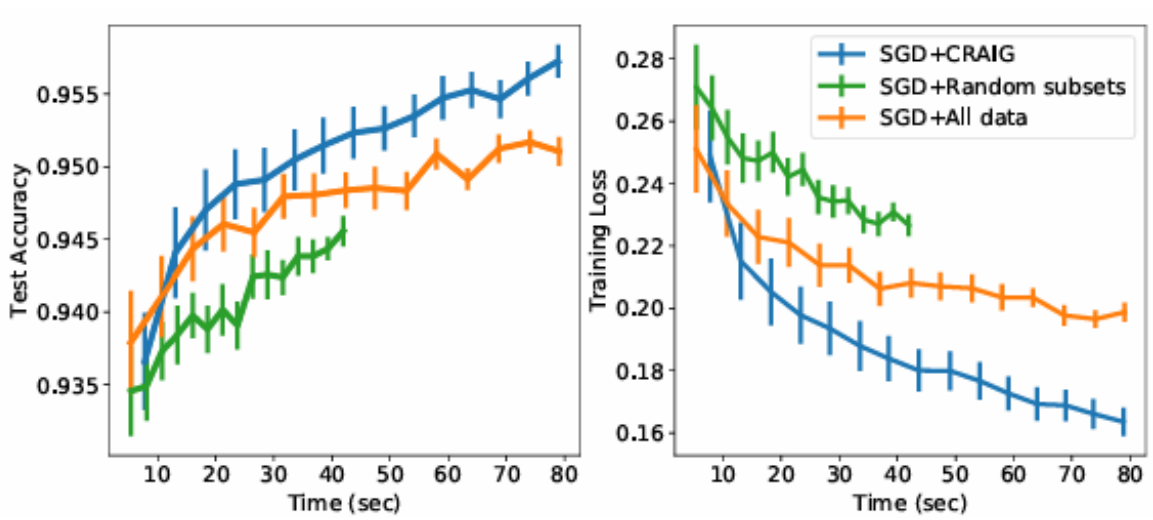


Figure 4: CRAIG vs. Random – MNIST 2-layer NN

CRAIG: Hiệu quả dữ liệu khi huấn luyện ResNet-20 trên CIFAR10

- CRAIG chọn tập con nhỏ (1–20%) tại mỗi epoch hoặc mỗi 5 epoch để huấn luyện ResNet-20.
- CRAIG đạt test accuracy tốt hơn so với random subset có cùng số điểm backprop.
- CRAIG sử dụng ít điểm dữ liệu phân biệt hơn $\rightarrow$ **hiệu quả dữ liệu cao hơn**.

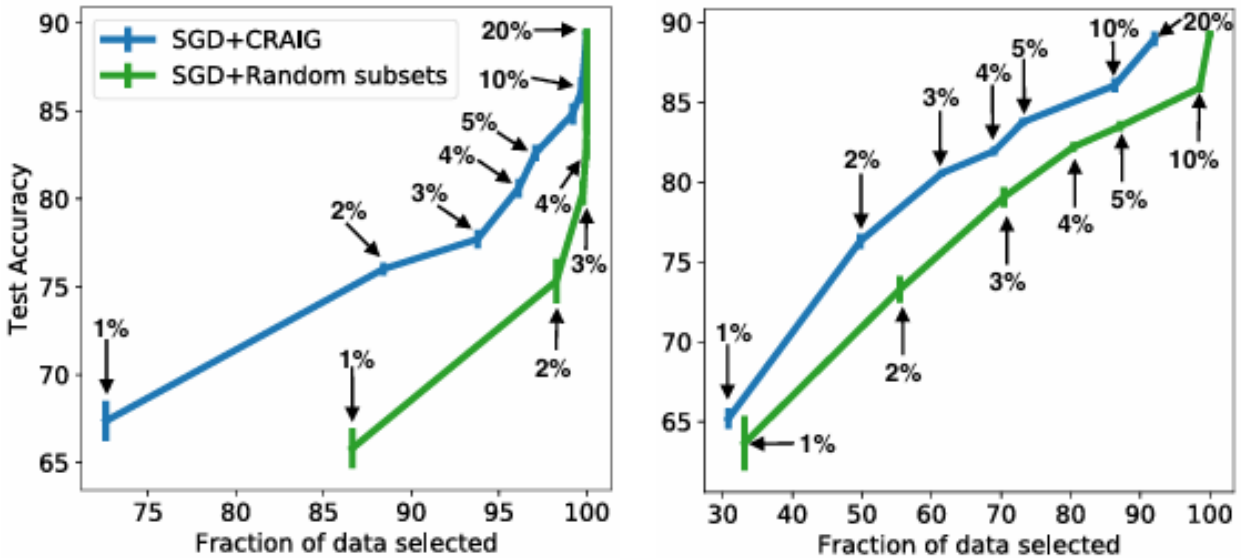
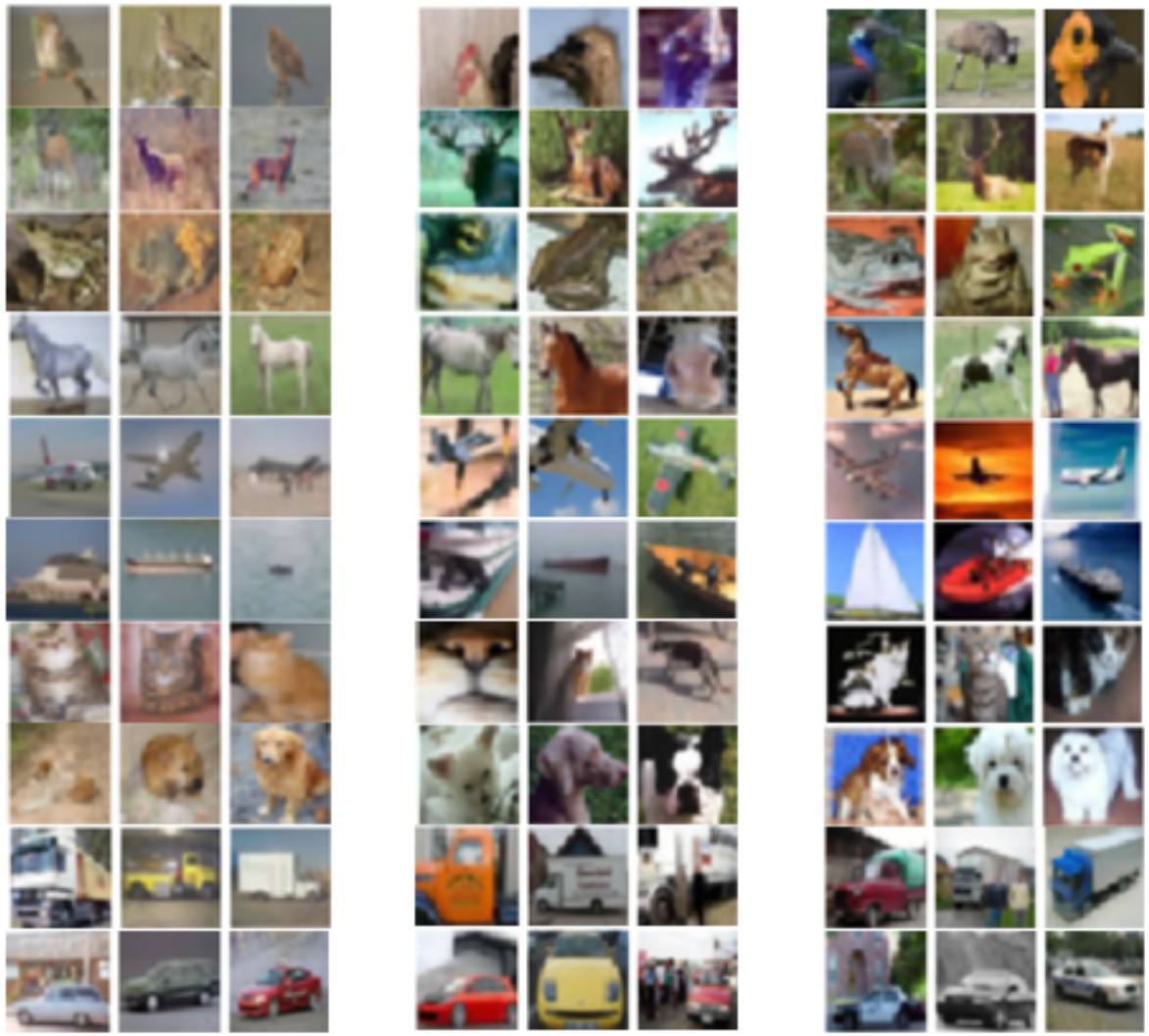


Figure 5: CRAIG vs. Random – Accuracy vs. Used Data (CIFAR10)

CRAIG chọn ảnh tiêu biểu hơn theo thời gian (CIFAR10)

- Hình ảnh trong coreset được chọn bởi CRAIG tại:
  - (a) Bắt đầu huấn luyện (epoch 1)
  - (b) Giữa quá trình huấn luyện (epoch 100)
  - (c) Kết thúc huấn luyện (epoch 200)
- Quan sát:
  - Về sau, CRAIG loại bỏ được nhiều ảnh dư thừa và chọn được ảnh đại diện cho các mẫu khó học hơn.
  - Coreset phản ánh rõ hơn sự đa dạng ngữ nghĩa giữa các lớp ảnh.

CRAIG chọn ảnh tiêu biểu hơn theo thời gian (CIFAR10)



(a) First (b) Middle (c) Last

Figure 6: Ảnh được CRAIG chọn tại các thời điểm khác nhau (CIFAR10)

Kết luận

- **Vấn đề đặt ra:** Huấn luyện mô hình học máy trên tập dữ liệu lớn gây tốn chi phí tính toán, đặc biệt với các phương pháp IG.
- **Đóng góp chính của CRAIG:**
  - Xây dựng một phương pháp chọn tập con (coreset) có trọng số γ sao cho gradient xấp xỉ toàn bộ tập dữ liệu.
  - Chứng minh hội tụ lý thuyết cho bất kỳ phương pháp IG nào áp dụng trên coreset.
  - Đạt được tăng tốc đáng kể (3–6x) trên các mô hình học máy và deep learning mà không làm giảm hiệu suất.
- **Ý nghĩa thực tiễn:**
  - CRAIG là một phương pháp đơn giản, hiệu quả, dễ triển khai trên các pipeline huấn luyện hiện có.
  - Giảm chi phí huấn luyện, tăng hiệu quả dữ liệu, đặc biệt phù hợp với các hệ thống lớn hoặc giới hạn tài nguyên.

Kết luận

- **Hướng nghiên cứu mở:**
  - Mở rộng CRAIG cho các bài toán không lồi, dữ liệu phân tán (federated learning).
  - Tích hợp CRAIG với tối ưu hóa bi-level để học trọng số gamma một cách chủ động.
  - Xây dựng CRAIG online cho dữ liệu streaming hoặc học liên tục (continual learning).