BỘ KHOA HỌC VÀ CÔNG NGHỆ

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Chu Quang Hưng

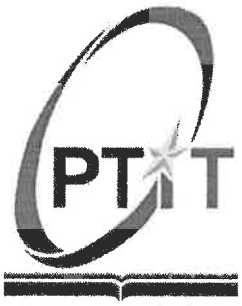
CẢI TIẾN CRAIG BẰNG CONVEX HULL ĐỂ XÂY DỰNG CORESET NÂNG CAO HIỆU SUẤT HUẤN LUYỆN MÔ HÌNH HỌC MÁY

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT
(Theo định hướng ứng dụng)

HÀ NỘI - 2026

BỘ KHOA HỌC VÀ CÔNG NGHỆ

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Chu Quang Hưng

CẢI TIẾN CRAIG BẰNG CONVEX HULL ĐỂ XÂY DỰNG CORESET NÂNG CAO HIỆU SUẤT HUẤN LUYỆN MÔ HÌNH HỌC MÁY

Chuyên ngành: Khoa học máy tính
Mã số: 8.48.01.01

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT
(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC : TS. Nguyễn Kiều Linh

HÀ NỘI - 2026

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.
Các số liệu, kết quả nêu trong đề án tốt nghiệp là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm 2026

Tác giả đề án tốt nghiệp



Chu Quang Hưng

LỜI CẢM ƠN

Trong quá trình học tập và hoàn thành đề án này, tôi đã nhận được sự giúp đỡ, hướng dẫn tận tình từ quý thầy cô, gia đình và bạn bè. Trước tiên, tôi xin gửi lời cảm ơn sâu sắc đến cô TS. Nguyễn Kiều Linh - người đã trực tiếp hướng dẫn và tận tình chỉ bảo tôi trong suốt quá trình thực hiện đề án. Những góp ý và định hướng của cô đã giúp tôi hoàn thiện nội dung nghiên cứu một cách tốt nhất.

Xin cảm ơn gia đình và bạn bè đã luôn động viên, khích lệ và hỗ trợ tôi trong suốt thời gian qua.

Mặc dù đã có nhiều cố gắng, nhưng do kiến thức và kinh nghiệm còn hạn chế nên đề án không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự góp ý của các thầy cô để em có thể hoàn thiện hơn. Tôi xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2026

Tác giả đề án tốt nghiệp



Chu Quang Hưng

Mục lục

LỜI CẢM ƠN ..... **i**

MỤC LỤC ..... **i**

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT ..... **iv**

DANH MỤC CÁC HÌNH VẼ ..... **vi**

DANH MỤC CÁC BẢNG ..... **vii**

PHẦN MỞ ĐẦU **1**

CHƯƠNG 1. CƠ SỞ LÝ LUẬN VÀ TỔNG QUAN **4**

1.1 Coreset và các phương pháp xây dựng coreset 4

1.1.1 Định nghĩa và vai trò của coreset 4

1.1.2 Phân loại các phương pháp lựa chọn coreset 6

1.1.3 Lấy mẫu theo độ nhạy (sensitivity sampling) 7

1.1.4 Lựa chọn dựa trên hàm dưới mô-đun (submodular selection) 8

1.1.5 Phương pháp CRAIG 10

1.1.6 Một số phương pháp coreset hiện đại 11

1.1.7 Chi phí lựa chọn và sự đánh đổi của coreset 13

1.2 Tối ưu hóa dựa trên gradient 13

1.2.1 Bài toán tối ưu dạng tổng và phương pháp gradient tăng dần 13

1.2.2 Thuật toán hạ gradient ngẫu nhiên (SGD) 14

1.2.3 Các biến thể cải tiến (SVRG, SAGA) 15

1.2.4 Lý thuyết hội tụ 17

1.3 Lý thuyết về bao lồi (convex hull) 17

1.3.1 Định nghĩa và tính chất hình học 17

1.3.2 Điểm cực biên, siêu phẳng tựa và thuật toán tính bao lồi 17

1.3.3 Vai trò trong biểu diễn và rút gọn dữ liệu 19

CHƯƠNG 2. CẢI TIẾN CRAIG BẰNG BAO LỒI 21

2.1 Phương pháp CRAIG 21

2.1.1 Phân tích phương pháp CRAIG 21

2.1.2 Phân tích khoảng cách d_{ij} trong các miền lồi và phi lồi 24

2.1.3 Hạn chế thực tiễn của thuật toán tham lam trong CRAIG 26

2.2 Ý tưởng cải tiến CRAIG bằng bao lồi 27

2.2.1 Cơ sở hình học 27

2.2.2 Khảo sát các kỹ thuật xấp xỉ bao lồi trong không gian nhiều chiều 29

2.2.3 Cải tiến CRAIG bằng thuật toán xấp xỉ bao lồi 29

2.3 Thuật toán CRAIG-CH (CRAIG with Convex Hull) 30

2.3.1 Bước 1: Tìm xấp xỉ bao lồi để tạo tập ứng viên H 30

2.3.2 Bước 2: Thực thi CRAIG trên tập ứng viên H 33

2.3.3 Trọng số coreset và hàm mất mát có trọng số 34

2.3.4 Ví dụ minh họa 36

2.4 Phân tích độ phức tạp và tài nguyên 37

2.5 Lựa chọn tham số và bảo đảm lý thuyết 39

2.5.1 Lựa chọn ngân sách ứng viên k_H 39

2.5.2 Bảo đảm lý thuyết được kế thừa 39

2.6 Ưu điểm kỳ vọng và sự đánh đổi 40

2.6.1 Hạn chế và phạm vi áp dụng 40

CHƯƠNG 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ 43

3.1 Môi trường và dữ liệu thực nghiệm 43

3.1.1 Môi trường thực nghiệm 43

3.1.2 Dữ liệu thực nghiệm 43

3.2 Các phương pháp so sánh 44

3.3 Các tiêu chí đánh giá 44

3.4 Kết quả thực nghiệm 45

3.4.1 Kiểm nghiệm với mô hình hồi quy logistic trên bộ dữ liệu FashionM-NIST 45

3.4.2 Kiểm nghiệm với mô hình ResNet-20 trên bộ dữ liệu CIFAR-10 49

PHẦN KẾT LUẬN 54

TÀI LIỆU THAM KHẢO 57

BẢN CAM ĐOAN TURNITIN 60

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
CHVS4	Tên thuật toán xấp xỉ bao lồi (Ding 2017) được sử dụng trong đề án
CNN: Convolutional Neural Network	Mạng nơ-ron tích chập
CRAIG-CH: CRAIG with Convex Hull	Phương pháp CRAIG cải tiến sử dụng bao lồi (phương pháp đề xuất)
CRAIG: Coresets for Accelerating Incremental Gradient descent	Phương pháp xây dựng coreset để huấn luyện hiệu quả (từ paper gốc)
DenseNet: Dense Network	Mạng nơ-ron kết nối dày đặc
DNN: Deep Neural Network	Mạng nơ-ron sâu
LR: Logistic Regression	Mô hình hồi quy logistic (cho bài toán lồi)
Random: lấy mẫu ngẫu nhiên	Phương pháp lấy mẫu ngẫu nhiên (dùng làm cơ sở so sánh)
ReLU: Rectified Linear Unit	Hàm kích hoạt (phi tuyến) phổ biến trong mạng nơ-ron
ResNet: Residual Network	Mạng nơ-ron kết nối tắt (mạng thặng dư)
SAGA	Thuật toán hạ gradient cải tiến (tương tự SVRG)
SGD: Stochastic Gradient Descent	Thuật toán hạ gradient ngẫu nhiên
SVRG: Stochastic Variance Reduced Gradient	Thuật toán hạ gradient giảm phương sai ngẫu nhiên
VGG: Visual Geometry Group	Kiến trúc mạng nơ-ron (đặt theo tên nhóm phát triển tại Oxford)
ViT: Vision Transformer	Kiến trúc mô hình Transformer cho thị giác máy tính

DANH MỤC HÌNH VẼ

1.1 Minh họa ý tưởng coreset. Từ toàn bộ điểm dữ liệu (chấm xám), phương pháp chọn một tập con nhỏ các điểm đại diện (điểm đỏ); mỗi điểm coreset mang một trọng số tương ứng với số điểm mà nó đại diện. 5
1.2 Sơ đồ phân loại các phương pháp lựa chọn coreset thành bốn nhóm. Đề án kết hợp nhóm hình học và nhóm dựa trên gradient (hai khối tô cam). 7

2.1 Minh họa ý tưởng CRAIG-CH trong không gian gradient. Các đỉnh bao lồi (vòng tròn xanh) là tập ứng viên H; bước tham lam của CRAIG chọn ra coreset (ô vuông đỏ) từ chính tập ứng viên này. 28
2.2 Quy trình hai giai đoạn của CRAIG-CH. Bước lọc bao lồi bằng CHVS4 thu gọn n_c điểm của lớp thành k_H ứng viên, rồi bước tham lam của CRAIG chọn ra coreset gồm r điểm kèm trọng số. 30
2.3 Cơ chế mở rộng đơn hình của thuật toán CHVS4. Bắt đầu từ điểm khởi tạo v_0 (đỉnh xa tâm dữ liệu nhất), mỗi bước bổ sung đỉnh có phần dư lớn nhất so với không gian con sinh bởi các đỉnh đã chọn; các đỉnh chọn ra đều nằm trên bao lồi. 31
2.4 Trọng số coreset. Coreset phân hoạch cả lớp thành r nhóm theo điểm coreset gần nhất; mỗi điểm coreset j (ô vuông đỏ) mang trọng số γ_j bằng số điểm của nhóm, và tổng các trọng số bằng số điểm của lớp. . . 35

3.1 So sánh độ chính xác các phương pháp trên FashionMNIST. 46
3.2 Thời gian bước lựa chọn coreset trên FashionMNIST. CRAIG-CH rẻ hơn CRAIG nhờ chỉ tìm trên tập ứng viên đã lọc. 47
3.3 So sánh độ chính xác các phương pháp trên CIFAR-10. 50
3.4 Thời gian bước lựa chọn coreset trên CIFAR-10. Chi phí của CRAIG tăng nhanh theo tỷ lệ nén, còn CRAIG-CH tăng chậm hơn. 51

DANH MỤC BẢNG BIỂU

2.1 Ví dụ minh họa: tám điểm gradient của một lớp, tập ứng viên H do CHVS4 lọc ra, và cách phân nhóm để gán trọng số coreset. 37
2.2 So sánh chi phí của bước lựa chọn coreset và bộ nhớ khoảng cách trên một lớp dữ liệu (cho một lần lựa chọn) 38

3.1 Độ chính xác kiểm thử trên FashionMNIST tại epoch 100 (đơn vị: %) . 45
3.2 Thời gian bước lựa chọn coreset trên FashionMNIST (đơn vị: giây/epoch) 47
3.3 Tổng thời gian thực thi trên FashionMNIST, 100 epoch (đơn vị: giây) . 47
3.4 Sai số xấp xỉ gradient đã chuẩn hóa trên FashionMNIST 48
3.5 Độ chính xác kiểm thử trên CIFAR-10 tại epoch 200 (đơn vị: %) 49
3.6 Thời gian bước lựa chọn coreset trên CIFAR-10 (đơn vị: giây/epoch) . . 51
3.7 Tổng thời gian thực thi trên CIFAR-10, 200 epoch (đơn vị: giây) 51
3.8 Sai số xấp xỉ gradient đã chuẩn hóa trên CIFAR-10 52

PHẦN MỞ ĐẦU

Trong kỷ nguyên dữ liệu lớn, việc huấn luyện các mô hình học máy, đặc biệt là các mô hình học sâu, ngày càng tốn kém về tài nguyên tính toán, thời gian và năng lượng [19]. Vì vậy, tăng hiệu suất huấn luyện luôn là một trong những mục tiêu quan trọng của học máy. Để xử lý lượng dữ liệu ngày càng lớn, các phương pháp tối ưu hóa lặp (Incremental Gradient – IG) như Stochastic Gradient Descent (SGD) [2, 18] đã trở thành chuẩn mực nhờ khả năng cập nhật tham số dựa trên một mẫu hoặc một minibatch tại mỗi bước. Bên cạnh đó, các biến thể giảm phương sai như SVRG [11] hay SAGA [5] tiếp tục cải thiện chất lượng cập nhật bằng cách giảm nhiễu gradient. Các kết quả lý thuyết cho thấy những thuật toán này có tốc độ hội tụ tốt hơn đáng kể so với việc sử dụng gradient toàn cục [8, 16]. Mặc dù vậy, ngay cả với các thuật toán IG, việc lặp lại toàn bộ dữ liệu nhiều lần trong quá trình huấn luyện vẫn tạo ra chi phí tính toán rất lớn khi kích thước dữ liệu tăng nhanh theo từng năm.

Để vượt qua giới hạn này, các nhà nghiên cứu đã đưa ra khái niệm coreset như một giải pháp theo hướng "giảm dữ liệu". Coreset là một tập con có trọng số sao cho việc huấn luyện trên tập con này tạo ra mô hình gần tương đương với mô hình được huấn luyện trên toàn bộ dữ liệu [15, 19]. Các công trình đầu tiên về coreset tập trung vào những bài toán cụ thể (ví dụ như k-means [9] hay Gaussian Mixture Models [13]) với giả định tương đối đơn giản về phân bố dữ liệu. Những phương pháp này đặt nền móng cho hướng xây dựng tập dữ liệu rút gọn nhưng vẫn mang tính đại diện cao.

Trong những năm gần đây, nhiều phương pháp coreset tổng quát hơn đã được phát triển. Một hướng tiếp cận nổi bật là CRAIG (Mirzasoleiman và cộng sự, 2020 [15]), trong đó tác giả xem gradient của hàm mất mát như "đại diện" cho đóng góp của mỗi mẫu, rồi tìm một tập con sao cho tổng gradient trên coreset gần nhất với tổng gradient trên toàn bộ dữ liệu. Bài toán này được mô hình hóa dưới dạng tối ưu một hàm dưới mô-đun (submodular) và giải bằng thuật toán tham lam (greedy) để chọn mẫu trong từng bước. CRAIG có cơ sở lý thuyết rõ ràng đối với các mô hình lồi, đồng thời cho hiệu quả thực nghiệm đáng kể: tăng tốc đến 6 lần ở bài toán hồi quy logistic và 3 lần ở mạng nơ-ron.

Tuy nhiên, CRAIG vẫn còn hạn chế khi áp dụng cho mô hình phi lồi. Để khắc phục, CREST (Yang và cộng sự, 2023 [24]) đã mở rộng hướng tiếp cận này bằng cách xấp xỉ cục bộ hàm mất mát thành các hàm bậc hai, rồi thực hiện lựa chọn coreset theo từng vòng. CREST còn sử dụng thêm cơ chế loại bỏ các mẫu "đã học" để tập trung vào những mẫu khó đối với mô hình. Qua các thử nghiệm trên CIFAR và dữ liệu ngôn ngữ, CREST giúp tăng tốc từ 1,7 đến 2,5 lần mà gần như không làm giảm hiệu năng

mô hình.

Ngoài ra, một số nghiên cứu gần đây còn tích hợp coreset với các mục tiêu chuyên biệt. RCS (Xu và cộng sự, 2023 [23]) hướng tới bài toán học tương phản đối kháng (adversarial contrastive learning), lựa chọn các mẫu sao cho sai khác biểu diễn giữa dữ liệu tự nhiên và dữ liệu bị tấn công được giữ ở mức thấp nhất, qua đó giúp mô hình duy trì tính bền vững. TAGCOS (Zhang và cộng sự, 2025 [25]) lại tập trung vào quá trình tinh chỉnh theo chỉ dẫn (instruction tuning) cho các mô hình ngôn ngữ lớn (Large Language Model – LLM). Phương pháp này sử dụng gradient làm đặc trưng để phân cụm các mẫu dữ liệu và chọn đại diện không phụ thuộc nhiệm vụ (task-agnostic); nhờ đó chỉ cần 5% dữ liệu đã đạt kết quả gần bằng khi huấn luyện trên toàn bộ dữ liệu.

Trong số các phương pháp trên, CRAIG là nền tảng trực tiếp cho đề án này nhờ khả năng áp dụng cho nhiều loại mô hình học máy. Tuy vậy, quá trình lựa chọn tham lam của CRAIG vẫn tốn kém về tính toán, vì tại mỗi bước phải đánh giá trên toàn bộ không gian dữ liệu. Đây chính là hạn chế mà đề án hướng tới khắc phục.

Mặt khác, bao lồi (convex hull) là một khái niệm cơ bản trong hình học tính toán, được sử dụng để xác định các điểm dữ liệu cực biên. Những điểm này thường mang nhiều thông tin về sự phân bố và ranh giới của dữ liệu. Do đó, việc sử dụng bao lồi để thu hẹp không gian tìm kiếm, tập trung vào các điểm "giàu thông tin" nhất, là một hướng hợp lý để cải tiến thuật toán lựa chọn dữ liệu.

Từ những phân tích trên, đề án "Cải tiến CRAIG bằng convex hull để xây dựng coreset nâng cao hiệu suất huấn luyện mô hình học máy" được đề xuất. Mục tiêu của đề án là giảm chi phí tính toán của bước lựa chọn coreset trong CRAIG, trong khi vẫn giữ chất lượng coreset ở mức hợp lý, qua đó giúp quá trình huấn luyện mô hình tiết kiệm tài nguyên hơn.

Các đóng góp chính của đề án gồm:

- Hệ thống hóa cơ sở lý luận về bài toán lựa chọn coreset và phân tích hạn chế của CRAIG về chi phí của bước lựa chọn.
- Đề xuất và cài đặt thuật toán CRAIG-CH, kết hợp bước lọc bao lồi CHVS4 với bước tối ưu hóa tham lam của CRAIG để giảm chi phí của bước lựa chọn coreset.
- Đánh giá thực nghiệm CRAIG-CH trên FashionMNIST (hồi quy logistic) và CIFAR-10 (ResNet-20), so sánh với lấy mẫu ngẫu nhiên, CRAIG và CHVS4 về độ chính xác, thời gian và sai số xấp xỉ gradient.

Về phạm vi, đề án tập trung vào bài toán phân loại ảnh với hai bộ dữ liệu chuẩn

nêu trên và hai lớp mô hình đại diện cho bài toán lồi và phi lồi; phương pháp nghiên cứu gồm phân tích lý thuyết kết hợp kiểm chứng thực nghiệm.

Ngoài phần mở đầu và kết luận, đề án trình bày ba nội dung chính như sau:

Chương 1: CƠ SỞ LÝ LUẬN VÀ TỔNG QUAN

Chương 1 trình bày tổng quan các kiến thức nền tảng cần thiết cho đề án: coreset và các phương pháp xây dựng coreset, tối ưu hóa dựa trên gradient, và lý thuyết về bao lồi (convex hull).

Chương 2: CẢI TIẾN CRAIG BẰNG BAO LỒI

Chương 2 phân tích hạn chế của phương pháp CRAIG gốc, giới thiệu ý tưởng và phương pháp cải tiến CRAIG bằng bao lồi, trình bày thuật toán CRAIG-CH (CRAIG with Convex Hull) và phân tích các ưu điểm kỳ vọng.

Chương 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Chương 3 trình bày chi tiết thiết lập thực nghiệm và các kết quả đạt được.

CHƯƠNG 1. CƠ SỞ LÝ LUẬN VÀ TỔNG QUAN

Chương này trình bày tổng quan các kiến thức nền tảng và cơ sở lý luận cần thiết cho đề án. Nội dung chính bao gồm ba trụ cột: (1) khái niệm về *coreset* và các phương pháp xây dựng coreset, đặc biệt là các phương pháp dựa trên hàm dưới mô-đun như CRAIG; (2) các thuật toán tối ưu hóa dựa trên gradient, vốn là công cụ để huấn luyện mô hình học máy; và (3) lý thuyết về bao lồi (convex hull), cung cấp một góc nhìn hình học cho việc rút gọn và biểu diễn dữ liệu. Những kiến thức này là nền tảng để phân tích, so sánh và đề xuất các phương pháp lựa chọn coreset hiệu quả trong các chương sau.

1.1 Coreset và các phương pháp xây dựng coreset

1.1.1 Định nghĩa và vai trò của coreset

Trong bối cảnh dữ liệu lớn (*big data*), việc huấn luyện các mô hình học máy trên toàn bộ tập dữ liệu V với n điểm dữ liệu, vốn yêu cầu tối ưu hóa một hàm mục tiêu dạng tổng:

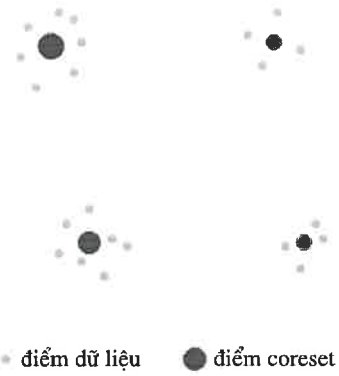
$$L(w) = \sum_{i=1}^{n} f_i(w) + r(w) \tag{1.1}$$

trở nên rất tốn kém về mặt tính toán. Ở đây, $f_i(w)$ là hàm mất mát (loss function) trên điểm dữ liệu thứ i và $r(w)$ là thành phần điều chuẩn (regularizer).

Một *coreset* (tập lõi) là một tập con nhỏ, có trọng số $S \subseteq V$ (với $|S| \ll n$), cùng với một bộ trọng số $\{\gamma_j\}_{j \in S}$, sao cho mô hình được huấn luyện trên coreset sẽ xấp xỉ tốt mô hình được huấn luyện trên toàn bộ tập V. Về mặt lý tưởng, hàm mất mát của coreset sẽ xấp xỉ hàm mất mát gốc:

$$L_S(w) = \sum_{j \in S} \gamma_j f_j(w) + r(w) \approx L(w) \tag{1.2}$$

Vai trò chính của coreset là giảm đáng kể chi phí tính toán (thời gian huấn luyện, yêu cầu bộ nhớ) trong khi vẫn duy trì được độ chính xác mong muốn của mô hình. Hình 1.1 minh họa ý tưởng này.



Hình 1.1. Minh họa ý tưởng coreset. Từ toàn bộ điểm dữ liệu (chấm xám), phương pháp chọn một tập con nhỏ các điểm đại diện (điểm đỏ); mỗi điểm coreset mang một trọng số tương ứng với số điểm mà nó đại diện.

Tùy theo phạm vi mà bảo đảm xấp xỉ có hiệu lực, người ta phân biệt hai mức độ chặt chẽ của một coreset. Một *coreset mạnh* (strong coreset) đòi hỏi hàm mất mát có trọng số $L_S(w)$ xấp xỉ tốt hàm mất mát gốc $L(w)$ tại *mọi* giá trị tham số w trong miền xét $\mathcal{W}$, tức là bảo đảm xấp xỉ đúng trên toàn bộ không gian tham số. Cụ thể, với một sai số tương đối $\epsilon > 0$ cho trước, tập con có trọng số S được gọi là một *ϵ-coreset* nếu

$$|L_S(w) - L(w)| \leq \epsilon\, L(w), \quad \forall w \in \mathcal{W}. \tag{1.3}$$

Ngược lại, một *coreset yếu* (weak coreset) chỉ yêu cầu bảo đảm xấp xỉ trong một lân cận của nghiệm tối ưu, hoặc chỉ cần nghiệm thu được khi tối ưu trên S đủ gần nghiệm thu được khi tối ưu trên toàn bộ V. Coreset mạnh cho bảo đảm tổng quát hơn nhưng thường khó xây dựng và có kích thước lớn hơn, trong khi coreset yếu dễ đạt hơn và đủ dùng khi mục tiêu chỉ là tìm nghiệm. Trong bối cảnh tăng tốc huấn luyện, điều thực sự quan trọng là coreset xấp xỉ tốt gradient dẫn dắt quá trình tối ưu dọc theo quỹ đạo huấn luyện, nên các phương pháp như CRAIG gần với tinh thần của coreset yếu hơn là đòi hỏi một xấp xỉ đều trên toàn bộ miền tham số.

Coreset được sử dụng theo hai mục tiêu khác nhau. Mục tiêu thứ nhất là *rút gọn dữ liệu* để lưu trữ và xử lý: người ta thay tập dữ liệu lớn bằng một tập con cô đọng có trọng số, rồi thực hiện một lần các phép phân tích hoặc tối ưu trên tập con đó thay vì trên toàn bộ dữ liệu. Cách dùng này phù hợp với những bài toán mà coreset chỉ cần được xây dựng một lần, chẳng hạn khi ước lượng tham số của một mô hình lồi hay khi phân cụm. Mục tiêu thứ hai, cũng là hướng mà đề án theo đuổi, là *tăng tốc huấn luyện lặp*: ở mỗi vòng của một thuật toán tối ưu dựa trên gradient, quá trình cập nhật tham số chỉ duyệt qua coreset thay vì toàn bộ tập dữ liệu, nhờ đó giảm chi phí tính toán trên mỗi vòng. Hai mục tiêu này chia sẻ cùng một ý tưởng nền tảng là thay thế

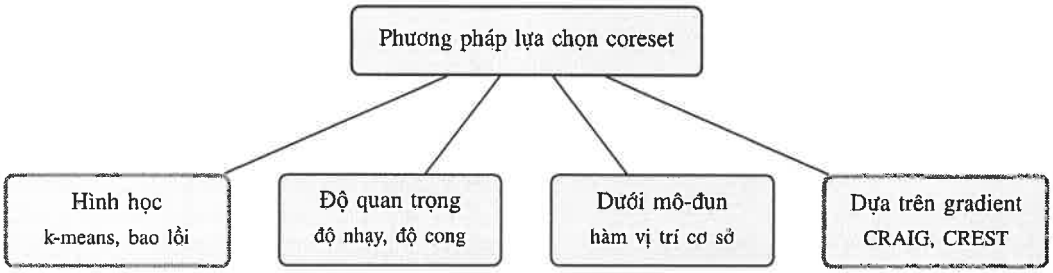
dữ liệu gốc bằng một đại diện nhỏ gọn, nhưng khác nhau ở tần suất sử dụng coreset và do đó đặt ra những yêu cầu khác nhau về chi phí của chính bước xây dựng coreset.

Khái niệm coreset có nguồn gốc từ hình học tính toán (computational geometry), nơi nó được đề xuất để giải các bài toán hình học và phân cụm trên những tập điểm rất lớn bằng cách chỉ giữ lại một tóm lược có kích thước nhỏ nhưng vẫn đủ để trả lời truy vấn với sai số kiểm soát được [9]. Một đặc điểm cốt lõi khiến ý tưởng này có giá trị là kích thước của coreset có thể độc lập, hoặc gần như độc lập, với số điểm dữ liệu n, mà phụ thuộc chủ yếu vào sai số cho phép ϵ và độ phức tạp nội tại của bài toán, chẳng hạn số chiều của dữ liệu hay số tham số của mô hình. Nói cách khác, khi n tăng lên rất lớn thì kích thước coreset không nhất thiết tăng theo cùng tốc độ, nên tỉ lệ rút gọn càng lớn và lợi ích tính toán càng rõ. Chính đặc điểm này giải thích vì sao coreset trở thành một công cụ tự nhiên cho bối cảnh dữ liệu lớn, và về sau được mang từ hình học tính toán sang bài toán rút gọn dữ liệu huấn luyện trong học máy.

Trọng số $\{\gamma_j\}_{j\in S}$ không phải là một chi tiết kỹ thuật tùy chọn mà là thành phần bắt buộc để coreset không bị chệch (unbiased). Nếu chỉ chọn tập con S rồi cộng trực tiếp $\sum_{j\in S} f_j(w)$ mà không nhân trọng số thì tổng này chỉ phản ánh khối lượng của riêng S và sẽ đánh giá thiếu một cách có hệ thống so với tổng gốc $\sum_{i\in V} f_i(w)$, đồng thời làm sai lệch tỉ lệ đóng góp giữa các vùng của tập dữ liệu. Mỗi điểm được chọn thực chất đứng thay cho một nhóm điểm quanh nó, nên phải được nhân với một trọng số phản ánh số lượng, hay khối lượng, điểm mà nó thay mặt; khi đó tổng có trọng số mới trở thành một ước lượng không chệch của tổng gốc. Tùy phương pháp, cách xác định trọng số khác nhau: với các phương pháp lấy mẫu không đều, trọng số thường được lấy bằng nghịch đảo xác suất chọn để bù lại việc các điểm được chọn với tần suất khác nhau; còn với các phương pháp dựa trên tính đại diện như CRAIG, trọng số γ_j đếm số điểm mà điểm j đại diện tốt nhất. Nếu bỏ qua trọng số, ước lượng gradient trung bình trên coreset sẽ bị chệch và có thể kéo quá trình tối ưu đi lệch hướng, làm mất chính lợi ích mà coreset hướng tới.

1.1.2 Phân loại các phương pháp lựa chọn coreset

Các phương pháp lựa chọn coreset có thể được chia thành bốn nhóm chính, dựa trên cách xác định tính đại diện của điểm dữ liệu. Hình 1.2 tóm tắt bốn nhóm này.



Hình 1.2. Sơ đồ phân loại các phương pháp lựa chọn coreset thành bốn nhóm. Đề án kết hợp nhóm hình học và nhóm dựa trên gradient (hai khối tô cam).

Nhóm thứ nhất dựa trên cấu trúc hình học của dữ liệu. Các phương pháp này khai thác phân bố không gian của các điểm để giữ lại những điểm định hình nên hình dạng của tập dữ liệu, tiêu biểu là coreset cho bài toán k-means và k-median [9], cho mô hình hỗn hợp Gauss [13], và các phương pháp dựa trên bao lồi.

Nhóm thứ hai dựa trên độ quan trọng của từng điểm. Mỗi điểm được gán một trọng số phản ánh mức ảnh hưởng của nó tới nghiệm, rồi coreset được lấy mẫu theo trọng số đó. Ngoài lấy mẫu theo độ nhạy, nhóm này còn có các phương pháp chọn điểm theo độ cong của hàm mất mát [7] hoặc theo độ khó của mẫu [21].

Nhóm thứ ba mô hình hóa tính đại diện và tính đa dạng bằng một hàm dưới mô-đun, rồi tối ưu bằng thuật toán tham lam [20]; hàm vị trí cơ sở là một ví dụ tiêu biểu của nhóm này.

Nhóm thứ tư dựa trên gradient: chọn tập con sao cho tổng gradient có trọng số của nó xấp xỉ tổng gradient của toàn bộ dữ liệu. Ngoài CRAIG [15] và CREST [24], hướng này còn được mở rộng bằng cách phân cụm gradient [3] hoặc tối thiểu hóa hàm mất mát bình phương [4].

Đề án nằm ở giao của hai nhóm cuối: dùng bao lồi trong không gian gradient để lọc ứng viên, rồi dùng bước tham lam dựa trên gradient của CRAIG để chọn coreset cuối cùng.

1.1.3 Lấy mẫu theo độ nhạy (sensitivity sampling)

Đây là một phương pháp xây dựng coreset cổ điển. Ý tưởng là gán cho mỗi điểm dữ liệu $i \in V$ một "độ nhạy" (sensitivity) σ_i, đại diện cho tầm quan trọng hoặc "đòn bẩy" (leverage) của điểm đó đối với bài toán. Các điểm có độ nhạy cao là những điểm có ảnh hưởng lớn nhất đến nghiệm cuối cùng.

Sau khi tính toán được độ nhạy, coreset được xây dựng bằng cách lấy mẫu ngẫu nhiên k điểm từ V, trong đó xác suất chọn điểm i tỷ lệ thuận với độ nhạy của nó:

$p_i \propto \sigma_i$. Mặc dù có bảo đảm lý thuyết cho một số bài toán cụ thể (như k-means), việc định nghĩa và tính toán độ nhạy cho các bài toán tối ưu lồi tổng quát hoặc cho mạng nơ-ron sâu là rất phức tạp.

1.1.4 Lựa chọn dựa trên hàm dưới mô-đun (submodular selection)

Một hướng tiếp cận khác là xem việc chọn coreset như một bài toán tối ưu hóa tổ hợp. Ta định nghĩa một hàm $F(S)$ đo lường chất lượng, sự đa dạng hoặc tính đại diện của một tập con S.

Một hàm tập hợp $F : 2^V \to \mathbb{R}$ được gọi là *dưới mô-đun* (submodular) nếu nó có tính chất "lợi ích giảm dần" (diminishing returns). Tức là, lợi ích của việc thêm một phần tử e vào một tập con nhỏ S sẽ lớn hơn hoặc bằng lợi ích của việc thêm e vào một tập con lớn hơn $T \supseteq S$:

$$F(e|S) \geq F(e|T), \quad \forall S \subseteq T \subseteq V, e \notin T \tag{1.4}$$

với $F(e|S) = F(S \cup \{e\}) - F(S)$ là lợi ích cận biên.

Nhiều hàm đo lường sự đa dạng, như các hàm vị trí cơ sở (facility location), là các hàm dưới mô-đun. Bài toán tìm tập con S kích thước k để tối đa hóa $F(S)$ (vốn là NP-hard) có thể được xấp xỉ rất hiệu quả bằng một thuật toán tham lam đơn giản: bắt đầu với $S = \emptyset$, và ở mỗi bước, thêm phần tử e có lợi ích cận biên $F(e|S)$ lớn nhất.

Chất lượng của thuật toán tham lam được phát biểu chính thức như sau.

Định lý 1.1. *Với một hàm dưới mô-đun đơn điệu không âm F, tập con S do thuật toán tham lam chọn thỏa mãn $F(S) \geq (1 - 1/e)\, F(S^*)$, trong đó S^* là nghiệm tối ưu và $1 - 1/e \approx 0{,}63$ [17].*

Nhờ bảo đảm này, thuật toán tham lam trở thành công cụ tiêu chuẩn cho các bài toán lựa chọn tập con dạng dưới mô-đun. Để giảm số lần đánh giá lợi ích cận biên, có thể dùng biến thể tham lam trì hoãn (lazy greedy): biến thể này khai thác tính lợi ích giảm dần để bỏ qua các phần tử chắc chắn không đạt lợi ích lớn nhất [14]. Tuy vậy, chi phí của bước lựa chọn vẫn tăng theo kích thước dữ liệu, và đây là điểm mà đề án tập trung cải thiện.

Để làm rõ hơn cơ sở của bảo đảm vừa nêu và cách hiện thực nó một cách hiệu quả, phần dưới đây phân tích kỹ ba khía cạnh: các điều kiện mà hàm mục tiêu cần thỏa mãn, trực giác của tỷ lệ $(1 - 1/e)$, và cơ chế tăng tốc bước tham lam.

Bên cạnh tính dưới mô-đun, bảo đảm $(1 - 1/e)$ ở Định lý 1.1 còn đòi hỏi hàm mục tiêu có thêm hai tính chất. Thứ nhất là tính *chuẩn hóa*: $F(\emptyset) = 0$, nghĩa là tập

rỗng không mang lại giá trị nào và mọi giá trị của hàm được đo tương đối so với điểm xuất phát này. Thứ hai là tính *đơn điệu* (monotone): việc mở rộng tập con không bao giờ làm giảm giá trị hàm,

$$F(S) \leq F(T), \quad \forall S \subseteq T \subseteq V. \tag{1.5}$$

Một cách tương đương, tính đơn điệu nói rằng mọi lợi ích cận biên đều không âm, $F(e \mid S) \geq 0$ với mọi e và mọi S. Hai tính chất này bảo đảm giá trị $F(S^*)$ ở vế phải bất đẳng thức là một đại lượng không âm và không giảm theo kích thước tập chọn, nhờ đó tỷ lệ xấp xỉ $(1 - 1/e)$ mới có ý nghĩa và bài toán tối đa hóa mới được đặt đúng [17].

Hàm *vị trí cơ sở* là ví dụ tiêu biểu thỏa mãn đồng thời cả ba yêu cầu trên (dưới mô-đun, đơn điệu và chuẩn hóa). Với mỗi cặp điểm $i, j \in V$, gọi $s_{ij} \geq 0$ là độ tương đồng giữa chúng, hàm có dạng

$$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}. \tag{1.6}$$

Về mặt trực giác, mỗi điểm i trong toàn bộ tập dữ liệu được "đại diện" bởi phần tử giống nó nhất trong tập chọn S, và số hạng $\max_{j \in S} s_{ij}$ chính là mức độ đại diện tốt nhất mà S dành cho i. Tổng trên toàn bộ V do đó đo mức độ bao phủ của S đối với dữ liệu: $F(S)$ nhận giá trị lớn khi mọi điểm đều tìm được một đại diện gần gũi trong S, và đó là lý do hàm này ưu tiên những tập con vừa đại diện tốt vừa trải rộng khắp dữ liệu. Với quy ước lấy $\max$ trên tập rỗng bằng 0, ta có ngay $F(\emptyset) = 0$, tức hàm được chuẩn hóa. Hàm cũng đơn điệu vì khi thêm một phần tử vào S, mỗi số hạng $\max_{j \in S} s_{ij}$ chỉ có thể giữ nguyên hoặc tăng lên, không bao giờ giảm. Cuối cùng, tính dưới mô-đun đến từ chỗ một phần tử mới chỉ tạo ra lợi ích ở những điểm i mà nó đại diện tốt hơn đại diện sẵn có; khi S đã lớn, số điểm như vậy ít dần đi, nên lợi ích cận biên giảm theo đúng đặc trưng "lợi ích giảm dần" [20].

Có thể hình dung trực giác đằng sau tỷ lệ $(1 - 1/e)$ qua diễn tiến của bước tham lam. Ký hiệu S_t là tập chọn sau t bước và S^* là nghiệm tối ưu kích thước k. Tính dưới mô-đun cùng tính đơn điệu bảo đảm rằng ở mỗi bước, phần tử tốt nhất luôn thu hẹp được ít nhất một phần tỷ lệ với khoảng cách còn lại tới tối ưu:

$$F(S_{t+1}) - F(S_t) \geq \frac{1}{k}(F(S^*) - F(S_t)). \tag{1.7}$$

Nói cách khác, mỗi lần thêm một phần tử, thuật toán "lấp" được ít nhất $1/k$ phần khoảng trống giữa giá trị hiện tại và giá trị tối ưu. Đặt $\delta_t = F(S^*) - F(S_t)$ là khoảng

trống đó, bất đẳng thức trên tương đương với $\delta_{t+1} \leq (1 - 1/k)\,\delta_t$, tức khoảng trống co lại theo cấp số nhân sau mỗi bước. Vì $\delta_0 = F(S^*)$ (do $F(\emptyset) = 0$), sau k bước ta được $\delta_k \leq (1 - 1/k)^k\, F(S^*) \leq e^{-1} F(S^*)$, dẫn tới $F(S_k) \geq (1 - 1/e)\, F(S^*)$. Chính cơ chế thu hẹp khoảng trống đều đặn này giải thích vì sao một chiến lược cục bộ, ở mỗi bước chỉ nhìn vào lợi ích cận biên trước mắt, lại đạt được một bảo đảm mang tính toàn cục [17].

Chi phí chủ yếu của bước tham lam nằm ở chỗ, tại mỗi vòng, về nguyên tắc phải tính lại lợi ích cận biên $F(e \mid S)$ cho mọi ứng viên e chưa được chọn, một khối lượng công việc lớn khi tập ứng viên có kích thước lớn. Biến thể *tham lam trì hoãn* (lazy greedy) khai thác chính tính lợi ích giảm dần để tránh phần lớn các lần tính lại này [14]. Do tính dưới mô-đun, lợi ích cận biên của một phần tử chỉ có thể giảm khi tập chọn lớn dần; vì thế giá trị $F(e \mid S)$ đã tính ở một vòng trước luôn là một chặn trên cho lợi ích cận biên thực của e ở các vòng sau. Thuật toán vì vậy duy trì một hàng đợi ưu tiên lưu các phần tử cùng chặn trên lợi ích của chúng, sắp xếp theo thứ tự giảm dần. Ở mỗi vòng, nó chỉ lấy phần tử ở đỉnh hàng đợi và tính lại lợi ích cận biên hiện thời của riêng phần tử này; nếu giá trị vừa tính vẫn không nhỏ hơn chặn trên (đã lưu) của phần tử đứng kế tiếp trong hàng đợi, thì theo tính chất chặn trên, phần tử ở đỉnh chắc chắn là phần tử có lợi ích cận biên lớn nhất và được chọn ngay mà không cần chạm tới các ứng viên còn lại. Ngược lại, phần tử này được cập nhật lại giá trị và đưa trở về đúng vị trí trong hàng đợi, rồi lặp lại quá trình. Nhờ vậy, mỗi vòng thường chỉ cần một vài lần đánh giá hàm thay vì phải quét toàn bộ tập ứng viên, giúp giảm đáng kể chi phí thực tế trong khi vẫn cho kết quả trùng khớp với bước tham lam thông thường [20].

1.1.5 Phương pháp CRAIG

Phương pháp CRAIG (Coresets for Accelerating Incremental Gradient descent) là một ví dụ tiêu biểu của việc áp dụng hàm dưới mô-đun vào lựa chọn coreset để huấn luyện hiệu quả.

Ý tưởng của CRAIG là chọn một tập con S có trọng số $\{\gamma_j\}$ sao cho gradient của coreset xấp xỉ tốt nhất gradient của toàn bộ tập dữ liệu. Tức là, CRAIG tìm cách cực tiểu hóa sai số xấp xỉ gradient trên toàn bộ không gian tham số w:

$$\min_{S \subseteq V, \gamma_j \geq 0} \max_{w \in \mathcal{W}} \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \tag{1.8}$$

Bài báo CRAIG chỉ ra rằng việc cực tiểu hóa một chặn trên (upper bound) của biểu

thức này tương đương với việc tối đa hóa một hàm vị trí cơ sở (vốn là một hàm dưới mô-đun):

$$F(S) = \sum_{i \in V} \max_{j \in S} s_{ij} \tag{1.9}$$

trong đó "sự tương đồng" s_{ij} được suy ra từ khoảng cách gradient d_{ij}, chẳng hạn $s_{ij} = C_0 - d_{ij}$ với C_0 là hằng số đủ lớn để $s_{ij} \geq 0$.

Do đó, CRAIG sử dụng thuật toán tham lam để xây dựng coreset: ở mỗi bước, chọn điểm dữ liệu mới j sao cho nó đại diện tốt nhất (giống nhất) cho nhiều điểm i chưa được bao phủ tốt trong V.

Đối với các mô hình mạng nơ-ron sâu, việc tính toán gradient đầy đủ là không khả thi. CRAIG đề xuất một phương pháp xấp xỉ hiệu quả: chỉ sử dụng gradient của hàm mất mát đối với *logits* (đầu vào của lớp tuyến tính cuối cùng), tức là $p - y_{\text{one-hot}}$, để đo khoảng cách.

1.1.6 Một số phương pháp coreset hiện đại

Trên nền tảng của CRAIG, nhiều phương pháp lựa chọn coreset gần đây đã mở rộng phạm vi áp dụng và cải thiện chất lượng coreset được chọn.

CREST [24] mở rộng hướng tiếp cận của CRAIG sang các mô hình phi lồi. Thay vì giả định hàm mất mát là lồi, CREST xấp xỉ cục bộ hàm mất mát bằng các hàm bậc hai và lựa chọn coreset theo từng vòng. Phương pháp này còn loại bỏ dần các điểm dữ liệu đã được mô hình học tốt để tập trung vào những điểm khó, nhờ đó giảm chi phí huấn luyện mà gần như không làm giảm độ chính xác.

RCS [23] áp dụng coreset cho bài toán học tương phản đối kháng (adversarial contrastive learning). Phương pháp này chọn các điểm dữ liệu sao cho sai khác biểu diễn giữa dữ liệu gốc và dữ liệu bị tấn công là nhỏ nhất, qua đó giúp mô hình giữ được tính bền vững trước nhiễu đối kháng.

TAGCOS [25] áp dụng coreset cho quá trình tinh chỉnh theo chỉ dẫn (instruction tuning) của các mô hình ngôn ngữ lớn (Large Language Model – LLM). Phương pháp này dùng gradient làm đặc trưng để phân cụm dữ liệu, rồi chọn các điểm dữ liệu đại diện không phụ thuộc nhiệm vụ (task-agnostic).

Điểm chung của các phương pháp trên là dùng gradient làm đặc trưng để đánh giá tầm quan trọng của từng điểm dữ liệu. Tuy nhiên, giống như CRAIG, chúng vẫn dựa vào quá trình lựa chọn tốn kém trên toàn bộ tập dữ liệu. Đây là động lực để đề án thu hẹp không gian tìm kiếm bằng góc nhìn hình học của bao lồi.

Bên cạnh ba phương pháp tiêu biểu trên, một số hướng gần đây đánh giá điểm dữ liệu theo những tín hiệu khác nhau. Nhánh thứ nhất vẫn dùng gradient làm đặc trưng nhưng thay đổi cách khai thác. Chai và cộng sự [3] phân cụm các gradient rồi chọn đại diện cho từng cụm, dựa trên nhận xét rằng những điểm có gradient gần nhau đóng góp gần như nhau vào bước cập nhật, nên chỉ cần giữ lại một số ít đại diện của mỗi cụm là đủ để tái tạo gradient trung bình của toàn cụm. Cách làm này cho coreset đa dạng và ổn định hơn so với việc chọn tham lam thuần túy, đồng thời dễ song song hóa; đổi lại, chất lượng coreset phụ thuộc nhiều vào bước phân cụm trong không gian gradient nhiều chiều, và bản thân việc phân cụm cũng phát sinh chi phí đáng kể. Theo một hướng khác, Chen và cộng sự [4] không khớp gradient mà đặt việc chọn coreset thành bài toán tối thiểu hóa sai khác bình phương giữa hàm mất mát trên coreset và hàm mất mát trên toàn bộ dữ liệu. Ưu điểm là tiêu chí lựa chọn gắn thẳng với đại lượng ta thực sự quan tâm là giá trị hàm mất mát, thay vì thông qua một chặn trên trung gian; hạn chế là bài toán bình phương này nhạy với các điểm ngoại lai và thường phải giải lại mỗi khi tham số mô hình thay đổi.

Nhánh thứ hai không chấm điểm theo mức đóng góp vào gradient trung bình, mà theo mức độ khó hoặc lượng thông tin mà mỗi mẫu mang lại. Garg và cộng sự [7] dựa vào độ cong của hàm mất mát quanh mỗi điểm: những điểm nằm ở vùng có độ cong lớn được xem là ảnh hưởng mạnh tới nghiệm và được ưu tiên giữ lại. Xia và cộng sự [21] đo độ khó của mẫu, chẳng hạn qua giá trị hàm mất mát hoặc mức độ mô hình còn dự đoán sai, rồi loại bỏ dần những mẫu dễ đã được học tốt để dành ngân sách coreset cho các mẫu khó và giàu thông tin hơn. Điểm mạnh của nhánh này là giữ lại đúng những mẫu quyết định ranh giới phân lớp, nhờ đó một coreset nhỏ vẫn có sức phân biệt cao; điểm yếu là các mẫu khó thường trùng với mẫu nhiễu hoặc bị gán nhãn sai, nên nếu không lọc cẩn thận, coreset có thể khuếch đại nhiễu và khiến mô hình kém ổn định.

Như vậy, bên cạnh gradient vốn là đặc trưng chung của các phương pháp kể trên, các nhánh vừa trình bày cho thấy tín hiệu dùng để đánh giá tầm quan trọng của điểm dữ liệu có thể đa dạng hơn: lấy trực tiếp từ hàm mất mát, hoặc từ độ khó và độ cong của mẫu. Tuy vậy, dù dựa trên tín hiệu nào, các phương pháp này đều chia sẻ một đặc điểm chung: để xếp hạng và chọn ra tập con, trước tiên chúng phải tính đặc trưng và chấm điểm cho gần như mọi điểm trên toàn bộ tập dữ liệu, nên phần lớn chi phí của bước lựa chọn coreset đến từ chính lần quét toàn bộ dữ liệu này. Đây cũng là lý do cần nhìn nhận chi phí của bước lựa chọn như một yếu tố tách bạch với chất lượng xấp xỉ, điều được phân tích ở tiểu mục tiếp theo.

1.1.7 *Chi phí lựa chọn và sự đánh đổi của coreset*

Chất lượng của một coreset thường được đánh giá qua mức xấp xỉ: coreset càng làm cho tổng gradient có trọng số gần với tổng gradient của toàn bộ dữ liệu thì càng tốt. Tuy nhiên, mức xấp xỉ không phải là yếu tố duy nhất, vì bản thân bước lựa chọn coreset cũng tốn chi phí tính toán, và chi phí này quyết định lợi ích thực tế của phương pháp.

Giữa hai yếu tố này tồn tại một sự đánh đổi. Một phương pháp tìm kiếm càng kỹ thì coreset thu được càng đại diện, nhưng chi phí lựa chọn càng lớn. Với mô hình lồi [1], coreset chỉ cần chọn một lần nên chi phí này ít quan trọng. Nhưng với mô hình phi lồi, gradient thay đổi theo tham số nên coreset phải được chọn lại định kỳ trong quá trình huấn luyện, khiến chi phí lựa chọn tích lũy và có thể lấn át phần thời gian tiết kiệm được. Vì vậy, một phương pháp tốt cần cân bằng giữa mức xấp xỉ và chi phí lựa chọn. Đây chính là hướng tiếp cận của đề án: giữ mức xấp xỉ ở mức hợp lý trong khi giảm chi phí của bước lựa chọn. Các đại lượng dùng để đánh giá cụ thể sự đánh đổi này được trình bày chi tiết ở Chương 3.

1.2 Tối ưu hóa dựa trên gradient

1.2.1 *Bài toán tối ưu dạng tổng và phương pháp gradient tăng dần*

Hầu hết các mô hình học máy được huấn luyện bằng cách cực tiểu hóa một hàm mục tiêu có dạng tổng hữu hạn trên toàn bộ tập dữ liệu:

$$L(w) = \sum_{i=1}^{n} f_i(w), \tag{1.10}$$

trong đó mỗi $f_i(w)$ là phần đóng góp của điểm dữ liệu thứ i vào hàm mất mát (ở đây tạm bỏ qua thành phần điều chuẩn để làm rõ cấu trúc tổng). Do gradient cũng là một tổng, $\nabla L(w) = \sum_{i=1}^{n} \nabla f_i(w)$, mỗi bước cập nhật theo gradient đầy đủ đòi hỏi duyệt qua toàn bộ n điểm và tốn chi phí $O(n)$; khi n lớn, đây trở thành nút thắt tính toán. *Phương pháp gradient tăng dần* (incremental gradient – IG) khắc phục điều này bằng cách chia nhỏ bước cập nhật: thay vì tính tổng gradient trên tất cả n điểm, mỗi bước chỉ dùng gradient của một (hoặc một số ít) điểm dữ liệu và lần lượt đi qua từng thành phần f_i của tổng. Nhờ đó tham số được cập nhật thường xuyên hơn với chi phí mỗi bước độc lập với n.

Một đặc điểm quyết định tới hành vi của phương pháp gradient tăng dần là thứ tự duyệt các thành phần f_i. Với cách duyệt *tuần hoàn* (cyclic), thuật toán đi qua các

điểm dữ liệu theo một trật tự cố định và lặp lại trật tự đó qua từng vòng; cách này đơn giản nhưng tốc độ hội tụ nhạy cảm với trật tự sắp xếp và trong trường hợp bất lợi có thể chậm [2]. Với cách duyệt *ngẫu nhiên*, thứ tự các điểm được xáo trộn lại ở mỗi vòng, giúp làm giảm ảnh hưởng của một trật tự cố định và thường cho hội tụ ổn định hơn trong thực tế. Các phân tích hội tụ cho phương pháp (dưới) gradient tăng dần chỉ ra rằng tốc độ hội tụ phụ thuộc chặt chẽ vào cách giảm bước học theo thời gian và vào thứ tự duyệt này [16, 2].

Một hướng cải tiến quan trọng là *phương pháp gradient tăng dần gộp* (incremental aggregated gradient – IAG). Thay vì chỉ dùng gradient của điểm hiện tại, IAG duy trì và cộng dồn các gradient đã tính gần nhất của mọi thành phần, rồi cập nhật tham số theo tổng gộp đó; mỗi bước chỉ làm mới gradient của một điểm nhưng vẫn tận dụng thông tin của toàn bộ tổng. Nhờ cơ chế này, với hàm mục tiêu lồi mạnh và bước học đủ nhỏ, IAG đạt tốc độ hội tụ tuyến tính, nhanh hơn hẳn so với gradient tăng dần cơ bản vốn chỉ hội tụ dưới tuyến tính [8]. Có thể thấy các phương pháp giảm phương sai được trình bày ở phần sau cũng chia sẻ chung tinh thần này: khai thác lại gradient đã lưu để bù đắp cho việc mỗi bước chỉ nhìn thấy một phần của tổng.

Chính họ phương pháp gradient tăng dần nêu trên là đối tượng mà CRAIG hướng tới tăng tốc. Tên đầy đủ của phương pháp, "Coresets for Accelerating Incremental Gradient descent", phản ánh trực tiếp ý tưởng đó: thay vì để thuật toán tăng dần duyệt qua toàn bộ n thành phần của tổng trong (1.10), CRAIG chọn ra một coreset S nhỏ cùng bộ trọng số $\{\gamma_j\}$ sao cho tổng gradient có trọng số trên S xấp xỉ tổng gradient trên toàn bộ dữ liệu [15]. Khi đó mỗi vòng huấn luyện chỉ cần duyệt qua $|S| \ll n$ điểm của coreset thay cho n điểm, nhờ đó giảm số lần tính gradient trên mỗi vòng mà vẫn giữ được hướng cập nhật gần như không đổi. Đây là cầu nối giữa bài toán tối ưu dạng tổng ở mục này và cơ chế lựa chọn coreset dựa trên gradient, mà cơ sở tối ưu và phân tích chi tiết của nó sẽ được trình bày ở chương tiếp theo.

1.2.2 Thuật toán hạ gradient ngẫu nhiên (SGD)

Thuật toán hạ gradient ngẫu nhiên (Stochastic Gradient Descent – SGD) là phương pháp lặp phổ biến nhất để tối ưu hóa các hàm mục tiêu dạng tổng. Thay vì tính toán gradient đầy đủ $\nabla L(w)$ (tốn $O(n)$ chi phí), ở mỗi bước t, SGD chỉ lấy một mẫu ngẫu nhiên (hoặc một mini-batch) i_t và cập nhật tham số theo hướng ngược lại của gradient ngẫu nhiên:

$$w_{t+1} = w_t - \eta_t \nabla f_{i_t}(w_t) \tag{1.11}$$

với η_t là tốc độ học (learning rate).

Ưu điểm của SGD là chi phí mỗi vòng lặp rất thấp ($O(1)$), cho phép nó thực hiện nhiều cập nhật nhanh chóng. Tuy nhiên, nhược điểm là gradient ngẫu nhiên có phương sai (variance) cao, khiến quá trình hội tụ bị dao động mạnh và thường chậm hơn ở giai đoạn cuối.

1.2.3 Các biến thể cải tiến (SVRG, SAGA)

Để giải quyết vấn đề phương sai cao của SGD, các phương pháp giảm phương sai đã được đề xuất.

- SVRG (Stochastic Variance Reduced Gradient): SVRG giảm phương sai bằng cách sử dụng một "snapshot" (ảnh chụp nhanh) của tham số $\tilde{w}$ và gradient đầy đủ $\mu_{\tilde{w}} = \frac{1}{n}\sum_i \nabla f_i(\tilde{w})$ tại điểm đó. Công thức cập nhật trong mỗi vòng lặp nội (inner loop) là:

$$w_{t+1} = w_t - \eta_t \left(\nabla f_{i_t}(w_t) - \nabla f_{i_t}(\tilde{w}) + \mu_{\tilde{w}}\right) \tag{1.12}$$

  Biểu thức này là một ước lượng không chệch (unbiased) của gradient đầy đủ, nhưng có phương sai thấp hơn nhiều khi w_t tiến gần đến $\tilde{w}$. SVRG yêu cầu tính toán gradient đầy đủ định kỳ, thường là sau mỗi epoch nội.

- SAGA: SAGA cũng là một phương pháp giảm phương sai nhưng tránh việc phải tính gradient đầy đủ. Thay vào đó, nó duy trì một bảng $G = [g_1, \ldots, g_n]$ lưu trữ gradient *gần nhất* đã được tính cho *mỗi* điểm dữ liệu. Công thức cập nhật là:

$$w_{t+1} = w_t - \eta_t \left(\nabla f_{i_t}(w_t) - g_{i_t} + \frac{1}{n}\sum_{k=1}^{n} g_k\right) \tag{1.13}$$

  Sau đó, g_{i_t} trong bảng được cập nhật thành $\nabla f_{i_t}(w_t)$. SAGA hội tụ nhanh nhưng yêu cầu bộ nhớ rất lớn ($O(nd)$) để lưu trữ toàn bộ bảng gradient.

Ngoài SGD lấy một điểm và gradient đầy đủ lấy toàn bộ n điểm, trong thực tế người ta thường dùng *mini-batch SGD* như một lựa chọn trung hòa giữa hai thái cực này. Ở mỗi bước, thay vì một điểm i_t, thuật toán lấy một lô ngẫu nhiên $B_t \subseteq V$ gồm $b = |B_t|$ điểm và cập nhật theo gradient trung bình trên lô:

$$w_{t+1} = w_t - \eta_t \frac{1}{|B_t|}\sum_{i \in B_t} \nabla f_i(w_t) \tag{1.14}$$

Vì gradient trên lô là trung bình của b ước lượng độc lập, phương sai của nó giảm xấp xỉ theo tỉ lệ $1/b$ so với SGD một điểm, nhờ đó quỹ đạo cập nhật ổn định hơn và

cho phép dùng tốc độ học lớn hơn. Ngoài ra, các gradient trong cùng một lô có thể được tính đồng thời, nên mini-batch SGD tận dụng tốt khả năng tính toán song song của phần cứng hiện đại như GPU. Kích thước lô b do đó là một tham số đánh đổi: lô lớn làm giảm phương sai và tăng hiệu suất phần cứng, nhưng mỗi bước tốn nhiều tính toán hơn và số lần cập nhật trên một epoch giảm đi.

Song song với việc giảm phương sai, một hướng cải tiến khác tác động trực tiếp lên hướng và độ dài của bước cập nhật. Phương pháp *quán tính* (momentum) tích lũy các gradient trong quá khứ thành một hướng di chuyển có đà, giúp dập bớt dao động theo những phương đổi dấu liên tục và tăng tốc theo những phương gradient nhất quán. Các phương pháp thích nghi tốc độ học, tiêu biểu là Adam, còn điều chỉnh riêng bước cập nhật cho từng thành phần tham số dựa trên thống kê gradient trong quá khứ, nhờ đó giảm sự nhạy cảm với việc chọn tốc độ học và xử lý tốt các gradient có độ lớn không đồng đều. Những kỹ thuật này được dùng rất phổ biến khi huấn luyện các mô hình phi lồi như mạng nơ-ron sâu; chúng độc lập với việc chọn tập dữ liệu nên có thể kết hợp trực tiếp với coreset.

Nhóm phương pháp giảm phương sai kể trên còn có một tiền thân đáng chú ý là SAG (stochastic average gradient) [18]. Giống SAGA, SAG duy trì một bảng lưu gradient gần nhất của mỗi điểm và dùng trung bình của bảng để ổn định hướng cập nhật, nhưng ước lượng gradient của SAG là chệch (biased); SAGA [5] về sau chỉnh lại thành ước lượng không chệch, đổi lại vẫn giữ chi phí bộ nhớ $O(nd)$ của bảng gradient. Trong khi đó, SVRG [11] chọn cách đánh đổi ngược lại: không lưu bảng mà tính lại gradient đầy đủ theo định kỳ, nhờ vậy tiết kiệm bộ nhớ nhưng phải quét toàn bộ dữ liệu sau mỗi chu kỳ. Như vậy, giữa các biến thể tồn tại một phổ đánh đổi giữa bộ nhớ và chi phí tính toán.

Điểm chung của mọi biến thể trên là chúng chỉ giảm phương sai của gradient ngẫu nhiên trên một tập dữ liệu cố định, chứ không làm giảm số điểm phải xử lý: mỗi epoch vẫn phải duyệt qua toàn bộ n điểm, dù để lấp đầy và cập nhật bảng gradient hay để tính lại gradient đầy đủ. Do đó chi phí mỗi epoch vẫn tỉ lệ với n và trở thành nút thắt khi n rất lớn. Đây chính là chỗ mà coreset bổ trợ cho các phương pháp này theo một hướng khác: thay vì cải thiện cách sử dụng gradient, coreset giảm trực tiếp số điểm phải xử lý ở mỗi vòng bằng cách chỉ giữ lại một coreset có trọng số đại diện cho toàn bộ dữ liệu. Nhờ tính bổ trợ đó, coreset có thể dùng chung với mini-batch SGD, quán tính hay các phương pháp giảm phương sai để cùng rút ngắn thời gian huấn luyện.

1.2.4 *Lý thuyết hội tụ*

Lý thuyết hội tụ cho các thuật toán tối ưu này phụ thuộc vào tính chất của hàm f.

- Hàm lồi (convex): SGD hội tụ với tốc độ $O(1/\sqrt{T})$ (cận tối ưu).
- Hàm lồi mạnh (strongly convex): SGD hội tụ *tuyến tính* ($O(\log(1/\epsilon))$) nhưng chỉ hội tụ đến một "vùng lân cận" (noise ball) quanh điểm tối ưu, do phương sai cố hữu. Các phương pháp SVRG và SAGA có thể đạt được hội tụ tuyến tính thực sự đến điểm tối ưu.
- Hàm phi lồi (non-convex, ví dụ mạng nơ-ron sâu): Các thuật toán này được đảm bảo sẽ hội tụ đến một điểm dừng (stationary point), tại đó norm của gradient gần bằng 0 ($\|\nabla L(w)\|^2 \leq \epsilon$).

Các phương pháp coreset như CRAIG được chứng minh là có thể duy trì các tốc độ hội tụ này, trong khi giảm đáng kể chi phí tính toán trên mỗi epoch.

1.3 Lý thuyết về bao lồi (convex hull)

1.3.1 *Định nghĩa và tính chất hình học*

Bao lồi (convex hull) của một tập hợp các điểm $P = \{p_1, \ldots, p_n\}$ trong không gian $\mathbb{R}^d$, ký hiệu là $\text{Conv}(P)$, được định nghĩa là tập hợp lồi nhỏ nhất chứa tất cả các điểm trong P.

Về mặt toán học, nó là tập hợp của tất cả các *tổ hợp lồi* (convex combinations) của các điểm trong P:

$$\text{Conv}(P) = \left\{ \sum_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \forall i, \text{ và } \sum_{i=1}^{n} \alpha_i = 1 \right\} \tag{1.15}$$

Về mặt hình học, ta có thể tưởng tượng bao lồi như một sợi dây thun được kéo căng để bao bọc xung quanh toàn bộ các điểm P. Các điểm $p_i \in P$ nằm trên ranh giới của bao lồi, tạo thành các góc của khối đa diện, được gọi là các *đỉnh* (vertices) của bao lồi.

1.3.2 *Điểm cực biên, siêu phẳng tựa và thuật toán tính bao lồi*

Trong số các điểm nằm trên biên của bao lồi, có một lớp điểm giữ vai trò cốt lõi là các *điểm cực biên* (extreme point). Một điểm $p \in P$ được gọi là điểm cực biên của $\text{Conv}(P)$ nếu nó không thể viết được thành tổ hợp lồi của những điểm khác trong tập, tức là không tồn tại hai điểm phân biệt $q, r \in \text{Conv}(P)$ và số $\lambda \in (0, 1)$ sao cho

$p = \lambda q + (1 - \lambda)r$. Nói cách khác, điểm cực biên nằm ở "góc" của khối lồi và không nằm bên trong đoạn thẳng nối bất kỳ hai điểm nào khác của khối. Đối với một tập điểm hữu hạn, tập các điểm cực biên trùng đúng với tập các đỉnh của bao lồi đã nêu ở tiểu mục trước.

Một cách phát biểu tương đương và trực quan hơn dựa trên khái niệm *siêu phẳng tựa* (supporting hyperplane). Điểm $p \in P$ là điểm cực biên khi và chỉ khi tồn tại một siêu phẳng đi qua p sao cho toàn bộ tập P nằm gọn về một phía của nó. Cụ thể, tồn tại vector pháp tuyến a và số b sao cho

$$\langle a, x \rangle \leq b \quad \forall x \in P, \qquad \text{và} \qquad \langle a, p \rangle = b. \tag{1.16}$$

Siêu phẳng $\{x : \langle a, x \rangle = b\}$ chỉ "chạm" vào tập tại điểm p mà không cắt qua phần trong của khối lồi. Chính vì mỗi đỉnh đều có một hướng a riêng để nó trở thành điểm xa nhất của tập, các điểm cực biên là những điểm "nhô ra" nhất theo một hướng nào đó, trong khi mọi điểm không phải cực biên đều bị kẹp giữa các đỉnh và không có siêu phẳng tựa của riêng mình.

Việc xác định tập điểm cực biên của một tập điểm cho trước là bài toán cơ bản của hình học tính toán, và đã có nhiều thuật toán chính xác hiệu quả trong không gian số chiều thấp. Trong mặt phẳng ($d = 2$), thuật toán quét Graham (Graham scan) sắp xếp các điểm theo góc cực rồi duyệt một lượt để loại dần những điểm không phải đỉnh, đạt độ phức tạp $O(n \log n)$; thuật toán gói quà (gift wrapping, còn gọi là Jarvis march) lần lượt "bọc" quanh tập điểm giống như gói một món quà, với chi phí $O(nh)$ trong đó h là số đỉnh của bao lồi; thuật toán Quickhull theo chiến lược chia để trị đạt $O(n \log n)$ trong trường hợp trung bình [10]. Khi h nhỏ so với n, các thuật toán này cho phép rút gọn tập dữ liệu xuống một số ít đỉnh với chi phí gần tuyến tính, nên bao lồi là một công cụ rút gọn rất hiệu quả trong không gian hai hoặc ba chiều.

Tuy nhiên, sức mạnh rút gọn của bao lồi phụ thuộc chặt chẽ vào số chiều. Trong mặt phẳng, các đỉnh thường chỉ chiếm một phần nhỏ của tập điểm, tạo thành một đường biên mảnh bao quanh dữ liệu, nên tập đỉnh nhỏ hơn hẳn tập gốc. Khi số chiều tăng, tỷ lệ điểm nằm trên biên của bao lồi lại có xu hướng tăng nhanh: ngày càng nhiều điểm trở thành điểm cực biên, khiến tập đỉnh không còn nhỏ so với tập gốc và làm mất dần lợi thế cô đọng. Kết quả là ở số chiều cao, việc tính bao lồi chính xác không những khó khả thi mà còn kém hữu ích, nên buộc phải chuyển sang các phương pháp xấp xỉ; phần phân tích chi tiết về chi phí tính toán, lời nguyền số chiều và các kỹ thuật xấp xỉ tương ứng được trình bày ở Chương 2.

1.3.3 Vai trò trong biểu diễn và rút gọn dữ liệu

Lý thuyết bao lồi cung cấp một cách rút gọn dữ liệu dựa trên cấu trúc hình học. Một tính chất cơ bản là *mọi* điểm p bên trong bao lồi $\text{Conv}(P)$ đều có thể được biểu diễn dưới dạng một tổ hợp lồi của các *đỉnh* của nó.

Áp dụng điều này vào học máy, nếu chúng ta xem tập hợp gradient của tất cả các điểm dữ liệu trong một lớp ($\{\nabla f_i(w)\}_{i \in \text{lớp } c}$) là tập P, thì các đỉnh của $\text{Conv}(P)$ đại diện cho các gradient cực biên (extreme) nhất của lớp đó.

Tổng gradient của toàn bộ lớp (hoặc gradient trung bình) là một điểm nằm *bên trong* bao lồi này. Do đó, thay vì sử dụng tất cả các điểm, chúng ta chỉ cần xác định các đỉnh của bao lồi. Tập hợp các đỉnh này tạo thành một coreset hình học, có khả năng biểu diễn toàn bộ tập dữ liệu gốc một cách rất cô đọng. Đây chính là cơ sở lý thuyết cho việc sử dụng bao lồi trong lựa chọn coreset: tập các đỉnh của bao lồi tạo thành một tập ứng viên giàu thông tin, có thể dùng để thu hẹp không gian tìm kiếm cho các thuật toán dựa trên hàm dưới mô-đun như CRAIG. Góc nhìn về điểm cực biên còn giúp giải thích vì sao, trong học máy, những điểm nằm ở rìa của tập gradient lại đáng được chú ý. Theo đúng định nghĩa nêu trên, một gradient cực biên là gradient không tái tạo được bằng tổ hợp lồi của các gradient còn lại; nó mang một hướng cập nhật mà không mẫu nào khác cung cấp được. Ngược lại, gradient của một điểm nằm sâu bên trong là "bản pha trộn" của các hướng đã có, nên phần lớn thông tin nó mang theo là dư thừa. Các gradient cực biên thường ứng với những mẫu không điển hình hoặc khó phân loại, những mẫu định hình phần rìa của phân phối và do đó có ảnh hưởng lớn tới ranh giới quyết định. Vì mỗi điểm cực biên đóng góp một hướng cập nhật riêng, một tập nhỏ gồm các điểm cực biên có thể bảo toàn được sự đa dạng của các hướng gradient trong khi loại bỏ phần lớn các mẫu dư thừa ở vùng trung tâm.

Đây là nền tảng cho phương pháp cải tiến CRAIG-CH được trình bày trong chương tiếp theo.

Kết luận chương

Chương này đã trình bày ba khối kiến thức cơ bản: (1) các phương pháp xây dựng coreset, tập trung vào CRAIG (dựa trên hàm dưới mô-đun và xấp xỉ gradient); (2) các thuật toán tối ưu cơ bản (SGD) và nâng cao (SVRG, SAGA) dùng để huấn luyện; và (3) lý thuyết bao lồi, cung cấp một nền tảng hình học để xác định các đỉnh mang tính đại diện của tập dữ liệu.

Cơ sở lý luận này dẫn tới hướng nghiên cứu trung tâm của đề án: kết hợp phương pháp lựa chọn coreset dựa trên tối ưu hóa (CRAIG) với góc nhìn hình học

của bao lồi nhằm giảm chi phí tính toán của CRAIG mà vẫn giữ được chất lượng của coreset. Các chương tiếp theo sẽ trình bày phương pháp cải tiến này (CRAIG-CH) và thiết kế các thí nghiệm để đánh giá nó về tốc độ, độ chính xác và sai số xấp xỉ gradient khi huấn luyện các mô hình học máy.

CHƯƠNG 2. CẢI TIẾN CRAIG BẰNG BAO LỒI

Từ cơ sở lý luận được trình bày trong Chương 1, có thể thấy CRAIG là một phương pháp lựa chọn coreset dựa trên việc tối ưu hóa hàm dưới mô-đun để xấp xỉ gradient toàn cục. Tuy nhiên, phương pháp này vẫn còn hạn chế, chủ yếu về chi phí tính toán khi áp dụng trên các tập dữ liệu lớn.

Chương này trình bày một cải tiến gọi là CRAIG-CH (CRAIG with Convex Hull), nhằm khắc phục hạn chế nói trên. Ý tưởng cốt lõi là kết hợp khả năng tối ưu hóa độ bao phủ của CRAIG với khả năng chọn điểm cực biên của lý thuyết bao lồi. Thay vì để CRAIG tìm kiếm trên toàn bộ tập dữ liệu, đề án dùng một xấp xỉ của bao lồi để lọc ra một tập ứng viên giàu thông tin, nhờ đó thuật toán tham lam của CRAIG hoạt động hiệu quả và nhanh hơn.

2.1 Phương pháp CRAIG

2.1.1 Phân tích phương pháp CRAIG

CRAIG (Coresets for Accelerating Incremental Gradient descent) là một phương pháp xây dựng coreset tổng quát, được thiết kế để tăng tốc các thuật toán IG (Incremental Gradient). Ý tưởng cốt lõi của CRAIG là tìm một tập con có trọng số S sao cho tổng gradient có trọng số của nó xấp xỉ tốt nhất tổng gradient của toàn bộ tập dữ liệu V. Mục tiêu này được phát biểu bằng bài toán tối ưu sau:

$$S^* = \arg \min_{\substack{S \subseteq V,\ \gamma_j \geq 0 \\ \forall j}} |S|, \quad \text{s.t.} \quad \max_{w \in \mathcal{W}} \left\| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \leq \epsilon. \tag{2.1}$$

trong đó w là vector tham số của mô hình, $\mathcal{W}$ là không gian tham số, $f_i(w)$ là hàm mất mát trên mẫu dữ liệu i, γ_j là trọng số của mẫu j trong coreset, và ϵ là sai số xấp xỉ cho phép.

Việc giải trực tiếp bài toán này là bất khả thi, vì nó đòi hỏi tính gradient trên toàn bộ không gian tham số $\mathcal{W}$, một công việc còn tốn kém hơn cả việc huấn luyện mô hình. Hơn nữa, việc tìm tập con tối ưu S^* là một bài toán NP-khó.

CRAIG vượt qua thách thức này bằng một bước chuyển đổi lý thuyết. Thay vì tối thiểu hóa trực tiếp sai số gradient, thuật toán tối thiểu hóa một cận trên của sai số

này. Bằng bất đẳng thức tam giác, sai số được chặn trên bởi:

$$\left\| \sum_{i \in V} \nabla f_i(w) \; - \; \sum_{j \in S} \gamma_j \nabla f_j(w) \right\| \; \leq \; \sum_{i \in V} \min_{j \in S} \|\nabla f_i(w) - \nabla f_j(w)\|. \tag{2.2}$$

với γ_j được định nghĩa là số lượng điểm dữ liệu trong V có gradient gần với gradient của điểm $j \in S$ nhất. Vế phải của bất đẳng thức là tổng các khoảng cách tối thiểu từ mỗi điểm trong tập dữ liệu đến một điểm trong coreset, xét trong không gian gradient.

Để loại bỏ sự phụ thuộc vào w, CRAIG định nghĩa khoảng cách d_{ij} giữa hai điểm dữ liệu i và j là chuẩn lớn nhất của hiệu gradient của chúng trên toàn bộ không gian tham số:

$$d_{ij} \triangleq \max_{w \in \mathcal{W}} \|\nabla f_i(w) - \nabla f_j(w)\| \, . \tag{2.3}$$

Bài toán tối ưu ban đầu được chuyển thành việc tối thiểu hóa hàm mục tiêu

$$L(S) = \sum_{i \in V} \min_{j \in S} d_{ij}. \tag{2.4}$$

Đây là bài toán tương đương với bài toán medoid (tìm phần tử đại diện) trong không gian gradient.

Điểm mấu chốt của CRAIG là chỉ ra rằng việc tối thiểu hóa $L(S)$ tương đương với việc tối đa hóa một hàm dưới mô-đun có tên là hàm vị trí cơ sở (facility location), được định nghĩa là

$$F(S) = L(\{s_0\}) - L(S \cup \{s_0\}), \tag{2.5}$$

trong đó s_0 là một phần tử phụ trợ và $L(\{s_0\})$ là một hằng số. Cách phát biểu này tương đương với dạng độ tương đồng $F(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$ đã nêu ở Chương 1, trong đó độ tương đồng s_{ij} được suy ra từ khoảng cách gradient d_{ij}: khoảng cách càng nhỏ thì độ tương đồng càng lớn. Hàm dưới mô-đun có tính chất lợi ích cận biên giảm dần, cho phép dùng một thuật toán tham lam đơn giản để tìm lời giải gần tối ưu một cách hiệu quả. Thuật toán này bắt đầu với một tập rỗng, và ở mỗi bước chọn phần tử mang lại lợi ích cận biên lớn nhất cho hàm mục tiêu $F(S)$.

Cận trên bằng bất đẳng thức tam giác nêu trên là hệ quả trực tiếp của cách

chọn trọng số γ_j, chứ không phải một phép chặn tùy tiện; phần dưới đây trình bày phép chứng minh ngắn gọn. Với mỗi điểm $i \in V$, ký hiệu $\zeta(i)$ là điểm coreset có gradient gần gradient của i nhất, và trọng số γ_j chính là số điểm được gán về đại diện j:

$$\zeta(i) = \arg\min_{j \in S} \|\nabla f_i(w) - \nabla f_j(w)\|, \qquad \gamma_j = |\{\, i \in V : \zeta(i) = j \,\}|. \tag{2.6}$$

Khi trọng số được định nghĩa như vậy, tổng có trọng số trên coreset khai triển ngược thành một tổng trên toàn bộ tập dữ liệu:

$$\sum_{j \in S} \gamma_j \, \nabla f_j(w) = \sum_{j \in S} \sum_{i \,:\, \zeta(i) = j} \nabla f_j(w) = \sum_{i \in V} \nabla f_{\zeta(i)}(w), \tag{2.7}$$

vì mỗi điểm i đóng góp đúng một lần gradient của đại diện gần nó nhất. Nói cách khác, việc nhân gradient của j với số điểm mà nó đại diện tương đương với việc thay gradient của từng điểm i bằng gradient của đại diện $\zeta(i)$ của nó. Nhờ đó, sai số xấp xỉ gradient trở thành tổng của các sai lệch theo từng điểm:

$$\sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \, \nabla f_j(w) = \sum_{i \in V} (\nabla f_i(w) - \nabla f_{\zeta(i)}(w)). \tag{2.8}$$

Áp dụng bất đẳng thức tam giác cho tổng các vector ở vế phải, rồi dùng chính định nghĩa của $\zeta(i)$, ta thu lại đúng cận trên đã phát biểu:

$$\left\| \sum_{i \in V} (\nabla f_i(w) - \nabla f_{\zeta(i)}(w)) \right\| \;\le\; \sum_{i \in V} \|\nabla f_i(w) - \nabla f_{\zeta(i)}(w)\| \;=\; \sum_{i \in V} \min_{j \in S} \|\nabla f_i(w) - \nabla f_j(w)\|. \tag{2.9}$$

Bước cuối dùng đẳng thức $\|\nabla f_i(w) - \nabla f_{\zeta(i)}(w)\| = \min_{j \in S} \|\nabla f_i(w) - \nabla f_j(w)\|$, vốn đúng do $\zeta(i)$ được chọn là đại diện gần nhất. Như vậy, chính cách gán mỗi điểm về đại diện gần nhất và đặt trọng số bằng số điểm được gán đã biến sai số tổng thể của cả tập dữ liệu thành một tổng các khoảng cách cục bộ, mỗi khoảng cách có thể tối thiểu hóa được một cách độc lập bằng việc chọn tập S cho tốt.

Bài toán medoid vừa nêu, tức tối thiểu hóa $L(S)$ với ràng buộc $|S| \le r$, có một đặc điểm cần nhấn mạnh: các đại diện bị buộc phải là những điểm dữ liệu có thật trong V (các *medoid*), chứ không phải các tâm cụm trung bình đặt ở vị trí bất kỳ như trong bài toán k-means. Ràng buộc này là bắt buộc đối với coreset: mỗi điểm coreset phải ứng với một mẫu có thật để có thể lấy gradient thật của nó trong quá trình huấn luyện, trong khi một tâm cụm trung bình không tương ứng với mẫu nào nên không dùng được. Do đó, tối thiểu hóa $L(S)$ đồng nghĩa với việc chọn r mẫu đại diện sao

cho mọi điểm trong lớp đều có một đại diện nằm gần nó trong không gian gradient.

Có thể thấy rõ nguồn gốc của tính dưới mô-đun của hàm vị trí cơ sở $F(S)$ đã nêu từ cấu trúc min trong $L(S)$. Khi thêm một đại diện j vào tập S, mỗi điểm i chỉ được lợi đúng phần khoảng cách rút ngắn được nếu j gần i hơn mọi đại diện đã có, nên lợi ích cận biên của j là

$$F(j \mid S) = \sum_{i \in V} \max\{\, 0,\ \min_{s \in S} d_{is} - d_{ij} \,\}. \tag{2.10}$$

Khi S càng lớn, khoảng cách $\min_{s \in S} d_{is}$ từ mỗi điểm tới đại diện gần nhất của nó đã càng nhỏ, nên lượng rút ngắn mà một đại diện mới còn có thể mang lại càng ít. Đây chính là biểu hiện cụ thể của tính lợi ích cận biên giảm dần đã nhắc tới, và là lý do bảo đảm $(1 - 1/e)$ của thuật toán tham lam áp dụng được cho hàm vị trí cơ sở.

2.1.2 *Phân tích khoảng cách d_{ij} trong các miền lồi và phi lồi*

Khả năng áp dụng của CRAIG phụ thuộc nhiều vào việc tính hoặc xấp xỉ khoảng cách d_{ij}. Cách tiếp cận này khác nhau đáng kể giữa mô hình lồi và mô hình phi lồi.

- Mô hình lồi: Đối với một lớp các bài toán lồi như hồi quy tuyến tính, hồi quy ridge, hồi quy logistic và máy vector hỗ trợ (SVM), hiệu của chuẩn gradient có thể được xấp xỉ và chặn trên bởi:

$$\forall w, i, j \quad \|\nabla f_i(w) - \nabla f_j(w)\| \leq d_{ij} \leq \text{const} \cdot \|x_i - x_j\| \tag{2.11}$$

  với điều kiện $\|w\|$ bị chặn. Đây là một kết quả quan trọng, vì nó cho thấy khoảng cách trong không gian gradient có thể được xấp xỉ bằng khoảng cách Euclidean trong không gian đặc trưng đầu vào. Nhờ vậy, ma trận khoảng cách d_{ij} có thể được tính trước *một lần duy nhất* trước khi huấn luyện. Do đó, đối với mô hình lồi, việc lựa chọn coreset trở thành một bước tiền xử lý tĩnh, độc lập với quá trình tối ưu hóa.

- Mô hình phi lồi (mạng nơ-ron sâu): Vấn đề phức tạp hơn nhiều đối với mạng nơ-ron sâu, vì không thể tìm một cận trên độc lập với w như trong trường hợp lồi. Tuy nhiên, các nghiên cứu cho thấy sự biến thiên của chuẩn gradient chủ yếu được quyết định bởi gradient của hàm mất mát đối với đầu vào của lớp cuối cùng.

Dựa trên nhận xét này, CRAIG đề xuất phép xấp xỉ:

$$\|\nabla f_i(w) - \nabla f_j(w)\| \;\leq\; c_1 \left\| \Sigma'_L(z_i^{(L)})\, \nabla f_i^{(L)}(w) \;-\; \Sigma'_L(z_j^{(L)})\, \nabla f_j^{(L)}(w) \right\| \;+\; c_2. \tag{2.12}$$

trong đó $\Sigma'_L(z_i^{(L)})\nabla f_i^{(L)}(w)$ là gradient của hàm mất mát đối với đầu vào của lớp cuối cùng (trước hàm kích hoạt) cho điểm dữ liệu i, còn c_1, c_2 là các hằng số.

Có thể giải thích vì sao gradient theo logits của lớp cuối chiếm phần lớn biến thiên của chuẩn gradient bằng cách xét chuỗi đạo hàm của phép lan truyền ngược. Với hàm mất mát entropy chéo kết hợp softmax, gradient của hàm mất mát theo logits (đầu vào của lớp tuyến tính cuối) có dạng đặc biệt gọn là $p - y$, trong đó p là vector xác suất dự đoán và y là nhãn ở dạng one-hot. Đại lượng $p - y$ đóng vai trò tín hiệu lỗi ở đỉnh mạng, và toàn bộ gradient ở các lớp phía trước đều được sinh ra bằng cách lan truyền ngược chính tín hiệu này.

Cụ thể, gradient của hàm mất mát theo tham số của một lớp ℓ có dạng tích của tín hiệu lỗi $\delta^{(\ell)}$ lan truyền về lớp đó với đầu vào của lớp; còn tín hiệu lỗi ở mỗi lớp lại được tính từ tín hiệu lỗi của lớp kế tiếp qua công thức truy hồi

$$\delta^{(\ell)} = \left((W^{(\ell+1)})^\top \delta^{(\ell+1)} \right) \odot \sigma'(z^{(\ell)}), \qquad \delta^{(L)} = p - y, \tag{2.13}$$

trong đó $W^{(\ell)}$ là ma trận trọng số của lớp ℓ, $z^{(\ell)}$ là đầu vào trước kích hoạt của lớp, σ' là đạo hàm của hàm kích hoạt, và $\odot$ là phép nhân theo từng phần tử. Điểm khởi đầu của truy hồi là $\delta^{(L)} = p - y$. Như vậy, tín hiệu lỗi ở mọi lớp đều là ảnh của $p - y$ qua một dãy phép biến đổi tuyến tính xác định bởi các ma trận trọng số $W^{(\ell)}$. Tại một giá trị tham số w cho trước, các ma trận trọng số này là chung cho mọi điểm dữ liệu; do đó sự khác nhau về gradient giữa hai mẫu, xét cho cùng, bắt nguồn từ sự khác nhau của $p - y$ ở đỉnh mạng được các lớp phía sau chia sẻ qua lan truyền ngược. Hai mẫu có $p - y$ gần nhau sẽ có tín hiệu lỗi gần nhau ở mọi lớp, kéo theo gradient đầy đủ cũng gần nhau.

Nhận xét này cho thấy sự khác biệt giữa gradient của hai điểm, tức đại lượng quyết định khoảng cách d_{ij}, được chi phối chủ yếu bởi sự khác biệt của thành phần $p - y$ giữa chúng. Thành phần $p - y$ vừa đặt ra độ lớn chung cho toàn bộ gradient, vừa là nguồn tín hiệu lỗi duy nhất được các lớp phía sau chia sẻ qua lan truyền ngược, nên phần biến thiên còn lại do các kích hoạt trung gian đóng góp là thứ yếu so với nó. Đây là cơ sở để CRAIG dùng chuẩn hiệu của vector $p - y$, một vector chỉ có số chiều bằng số lớp C, thay cho chuẩn hiệu gradient đầy đủ khi đo khoảng cách d_{ij} trong không gian gradient.

Phép xấp xỉ này phụ thuộc vào tham số w, vốn thay đổi liên tục trong quá trình huấn luyện. Điều này có nghĩa là khoảng cách d_{ij}, và do đó coreset tối ưu, cũng thay đổi theo thời gian. Vì vậy, đối với mô hình học sâu, CRAIG phải được thực thi định kỳ để cập nhật coreset. Sự chuyển từ lựa chọn coreset tĩnh (mô hình lồi) sang lựa chọn động (mô hình phi lồi) có ý nghĩa lớn về hiệu suất: chi phí tính toán của chính thuật toán lựa chọn coreset trở thành một yếu tố quan trọng, ảnh hưởng trực tiếp đến tổng thời gian huấn luyện.

2.1.3 Hạn chế thực tiễn của thuật toán tham lam trong CRAIG

CRAIG cung cấp các bảo đảm hội tụ chặt chẽ về lý thuyết. Các định lý trong bài báo cho thấy một thuật toán IG khi áp dụng trên coreset do CRAIG tạo ra sẽ hội tụ với cùng tốc độ ($O(1/\sqrt{k})$ hoặc $O(1/k^\tau)$) như khi áp dụng trên toàn bộ tập dữ liệu. Tuy nhiên, nó chỉ hội tụ đến một vùng lân cận của điểm tối ưu, và kích thước vùng lân cận này phụ thuộc vào sai số xấp xỉ gradient ϵ.

Xét một lớp dữ liệu c với tập chỉ số mẫu V_c và số điểm $n_c = |V_c|$; coreset và bao lồi trong đề án đều được xây dựng trên từng lớp. Hạn chế đáng kể nhất của CRAIG nằm ở chi phí của bước lựa chọn tham lam: tại mỗi bước, để tìm phần tử mang lại lợi ích cận biên lớn nhất, thuật toán phải duyệt qua tất cả các điểm còn lại của lớp, nên một lần lựa chọn coreset cần $O(n_c \cdot r)$ lần đánh giá lợi ích cận biên, với $r = |S|$ là kích thước coreset (chi phí đầy đủ, kể cả chi phí của mỗi lần đánh giá, được phân tích ở Mục 2.4). Đối với mô hình lồi, chi phí này chỉ phải trả một lần. Nhưng đối với mô hình học sâu, khoảng cách gradient thay đổi theo tham số nên coreset phải được chọn lại định kỳ; nếu chọn lại E lần trong quá trình huấn luyện thì số lần đánh giá lên tới $O(E \cdot n_c \cdot r)$, tăng theo số điểm n_c.

Khi triển khai thực tế, CRAIG bộc lộ hai hạn chế:

- Chi phí của bước tham lam: ở mỗi bước phải giải

$$j^* = \arg\max_{j \in V_c \setminus S_k} F(j \mid S_k) \tag{2.14}$$

  bằng cách duyệt qua tất cả các điểm còn lại của lớp; chi phí này còn nhân lên theo số lần chọn lại coreset trong huấn luyện học sâu, như đã phân tích ở trên.

- Nguy cơ chọn điểm dư thừa trong các cụm đặc: Hàm vị trí cơ sở của CRAIG ưu tiên bao phủ các điểm dữ liệu. Nếu có một cụm rất đặc, gồm nhiều điểm nằm gần nhau nhưng ở *bên trong* đám mây gradient (tức không phải điểm cực biên), thuật toán tham lam có thể chọn nhiều điểm từ cụm này để tối đa hóa lợi ích cận biên.

Điều này có thể dẫn đến một coreset thiếu đa dạng về hình học, bỏ lỡ các điểm cực biên quan trọng nhưng thưa thớt, vốn là các điểm định hình nên bao lồi.

Đây là động lực chính của đề án: thu hẹp không gian tìm kiếm từ n_c điểm xuống một tập ứng viên nhỏ hơn nhiều, qua đó vừa giảm chi phí của bước lựa chọn, vừa hạn chế nguy cơ chọn điểm dư thừa.

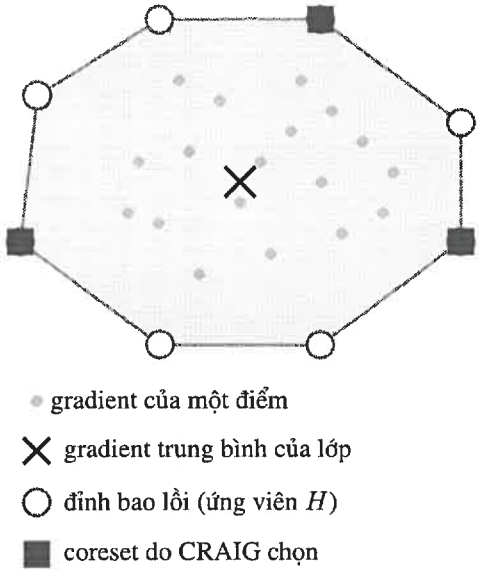
2.2 Ý tưởng cải tiến CRAIG bằng bao lồi

2.2.1 Cơ sở hình học

Phần này xây dựng cơ sở lý luận cho việc sử dụng bao lồi, đồng thời đối mặt trực tiếp với thách thức kỹ thuật lớn nhất là tính bất khả thi của việc tìm bao lồi chính xác trong không gian nhiều chiều.

Trước hết, để thấy vì sao chỉ một số ít điểm cực biên đã đủ định hình cả tập dữ liệu, ta xét một ví dụ quen thuộc trong học máy là máy vector hỗ trợ (SVM). SVM là một thuật toán phân loại tìm siêu phẳng có lề tối đa để tách hai lớp dữ liệu. Về mặt hình học, bài toán này tương đương với việc tìm khoảng cách ngắn nhất giữa hai bao lồi của các điểm dữ liệu thuộc mỗi lớp. Các điểm dữ liệu quan trọng nhất, gọi là các vector hỗ trợ, là những điểm nằm trên lề của siêu phẳng phân tách; về mặt hình học, chúng chính là các đỉnh nằm trên biên của các bao lồi tương ứng. Như vậy, chỉ một tập con rất nhỏ các điểm cực biên xác định hình dạng của bao lồi là thực sự cần thiết để định nghĩa toàn bộ ranh giới quyết định, còn các điểm nằm sâu bên trong bao lồi không đóng góp vào việc xác định siêu phẳng tối ưu.

Một cơ sở chặt chẽ hơn đến từ định lý Carathéodory: trong không gian $\mathbb{R}^d$, mọi điểm nằm trong bao lồi của một tập điểm đều biểu diễn được dưới dạng tổ hợp lồi của không quá $d+1$ đỉnh của bao lồi đó. Như đã nêu ở Chương 1, gradient trung bình của một lớp $\bar{g}_c = \frac{1}{n_c}\sum_{i \in V_c} g_i$ nằm bên trong bao lồi của tập gradient lớp đó, nên tái tạo được từ các đỉnh. Do đó, việc ưu tiên chọn các đỉnh bao lồi làm ứng viên cho coreset là một lựa chọn có cơ sở hình học, chứ không phải một phỏng đoán cảm tính. Hình 2.1 minh họa ý tưởng này trong không gian gradient.



Hình 2.1. Minh họa ý tưởng CRAIG-CH trong không gian gradient. Các đỉnh bao lồi (vòng tròn xanh) là tập ứng viên H; bước tham lam của CRAIG chọn ra coreset (ô vuông đỏ) từ chính tập ứng viên này.

Tuy có cơ sở lý thuyết, việc áp dụng trực tiếp khái niệm bao lồi [10] vào các bài toán học máy hiện đại vấp phải một rào cản gần như không thể vượt qua: lời nguyền số chiều (curse of dimensionality). Các thuật toán tính bao lồi chính xác hoạt động hiệu quả trong không gian 2 hoặc 3 chiều, với độ phức tạp thời gian thường là $O(n \log n)$ hoặc $O(n \log h)$, trong đó n là tổng số điểm và h là số điểm trên bao lồi. Tuy nhiên, khi số chiều d tăng, độ phức tạp tính toán tăng theo cấp số nhân; các thuật toán như Quickhull hay Clarkson–Shor có độ phức tạp trong trường hợp xấu nhất là $O(n^{\lfloor d/2 \rfloor})$.

Trong đề án, bao lồi được xét trong không gian gradient. Như đã trình bày ở Chương 1, đối với mô hình học sâu, CRAIG xấp xỉ gradient của mỗi mẫu bằng gradient của hàm mất mát theo logits (đầu vào của lớp tuyến tính cuối), tức một vector có số chiều bằng số lớp C của bài toán phân loại. Với FashionMNIST và CIFAR-10 ($C = 10$), độ phức tạp tính bao lồi chính xác đã lên tới $O(n^5)$; với các bài toán nhiều lớp hơn như CIFAR-100 ($C = 100$), con số này là $O(n^{50})$. Khi n lên tới hàng chục nghìn mẫu mỗi lớp, việc tính bao lồi chính xác là bất khả thi. Đây là một ràng buộc không thể bỏ qua, và nó định hình lại toàn bộ hướng tiếp cận: bất kỳ phương pháp nào dựa trên bao lồi trong không gian nhiều chiều đều phải từ bỏ ý định tính toán chính xác và chuyển sang các phương pháp xấp xỉ.

2.2.2 *Khảo sát các kỹ thuật xấp xỉ bao lồi trong không gian nhiều chiều*

Vì việc tính chính xác là không khả thi, trọng tâm chuyển sang các phương pháp xấp xỉ bao lồi. Có hai nhóm phương pháp chính:

- Loại 1 – Giảm chiều rồi tính bao lồi chính xác: Ý tưởng là dùng một kỹ thuật giảm chiều, chẳng hạn phân tích thành phần chính (Principal Component Analysis – PCA), để chiếu dữ liệu từ không gian d chiều ban đầu xuống một không gian d' chiều thấp hơn nhiều (d' từ 4 đến 10), rồi áp dụng thuật toán tính bao lồi chính xác như Quickhull trên tập đã chiếu. Các đỉnh tìm được trong không gian chiều thấp được ánh xạ ngược lại để xác định điểm ứng viên trong không gian ban đầu. Cách này khả thi về tính toán, nhưng phép chiếu có thể làm mất thông tin về cấu trúc cực biên ở không gian gốc.
- Loại 2 – Xấp xỉ trực tiếp trong không gian nhiều chiều: Nhóm phương pháp này tìm trực tiếp một tập con các đỉnh xấp xỉ bao lồi mà không cần giảm chiều, chấp nhận một sai số xấp xỉ nhất định (chẳng hạn theo khoảng cách Hausdorff). Thuật toán CHVS4 [6] thuộc nhóm này; ngoài ra còn có các hướng dựa trên mạng nơ-ron như DeepHull, trong đó một mạng nơ-ron lồi được huấn luyện để xấp xỉ ranh giới đa diện của bao lồi.

Đề án sử dụng phương pháp loại 2, cụ thể là thuật toán CHVS4. Lý do là CHVS4 tìm trực tiếp các đỉnh bao lồi trong không gian gradient, giữ nguyên cấu trúc hình học mà không qua bước giảm chiều, đồng thời có chi phí thấp và dễ tích hợp vào quy trình lựa chọn coreset. Cơ chế cụ thể của CHVS4 được trình bày ở Mục 2.3.

2.2.3 *Cải tiến CRAIG bằng thuật toán xấp xỉ bao lồi*

Mục tiêu của CRAIG là tìm một tập con nhỏ có gradient đại diện cho toàn bộ tập dữ liệu. Các điểm nằm trên bao lồi là những điểm cực biên trong không gian gradient, nên gradient của chúng thường mang những hướng riêng biệt mà các điểm ở trung tâm không có. Vì vậy, các điểm này có khả năng tạo ra lợi ích cận biên cao trong hàm vị trí cơ sở của CRAIG.

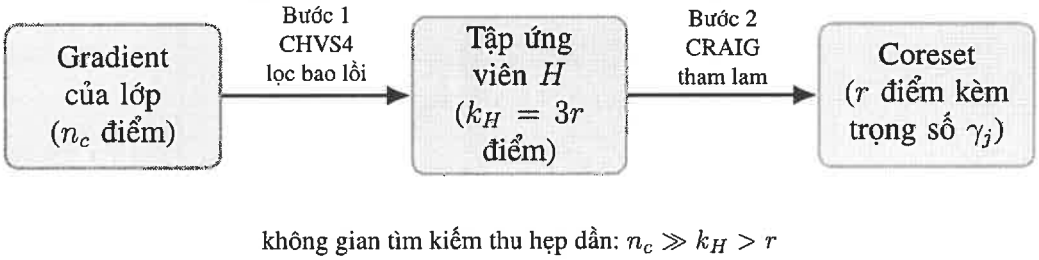
Do đó, tập các điểm xác định ranh giới hình học của dữ liệu là một tập ứng viên phù hợp cho tập các điểm xác định ranh giới gradient. Đây chính là mối liên kết giữa hai ý tưởng. Vì không thể tính bao lồi chính xác, đề án sử dụng tập đỉnh của một *bao lồi xấp xỉ* làm tập ứng viên để lựa chọn coreset. Thuật toán CRAIG-CH được xây dựng dựa trên phân tích đó, với quy trình gồm hai giai đoạn:

- Giai đoạn 1 – Lọc hình học: Thay vì làm việc trên toàn bộ tập dữ liệu V, đề án dùng thuật toán CHVS4 để tìm xấp xỉ tập đỉnh bao lồi trong không gian gradient, tạo ra một tập ứng viên thu nhỏ H chứa các điểm giàu thông tin nhất.
- Giai đoạn 2 – Lựa chọn dựa trên gradient: Sau khi có tập ứng viên H, thuật toán tham lam của CRAIG được thực thi trên tập này. Thay vì phải tìm phần tử tối ưu trên toàn bộ không gian V rộng lớn, thuật toán chỉ cần duyệt trong không gian thu hẹp H ($|H| \ll |V|$), giúp giảm đáng kể chi phí tính toán mà vẫn duy trì chất lượng coreset.

Bằng cách buộc CRAIG chỉ được chọn các điểm từ tập H (các điểm nằm trên hoặc gần bao lồi), đề án kỳ vọng giải quyết đồng thời hai hạn chế nêu trên. Các ưu điểm cụ thể của cách làm này được phân tích ở phần cuối chương.

2.3 Thuật toán CRAIG-CH (CRAIG with Convex Hull)

Đề án đề xuất thuật toán lai CRAIG-CH, kết hợp thuật toán tìm đỉnh bao lồi CHVS4 và thuật toán tham lam của CRAIG. Thuật toán hoạt động theo các bước sau, được thực hiện trên từng lớp dữ liệu. Hình 2.2 tóm tắt quy trình.



Hình 2.2. Quy trình hai giai đoạn của CRAIG-CH. Bước lọc bao lồi bằng CHVS4 thu gọn n_c điểm của lớp thành k_H ứng viên, rồi bước tham lam của CRAIG chọn ra coreset gồm r điểm kèm trọng số.

2.3.1 Bước 1: Tìm xấp xỉ bao lồi để tạo tập ứng viên H

Như đã phân tích ở Mục 2.2.1, việc tìm bao lồi chính xác trong không gian gradient là không khả thi do lời nguyền số chiều. Do đó, đề án sử dụng thuật toán xấp xỉ bao lồi CHVS4 [6]. Đây là một thuật toán được đề xuất cho bài toán phân loại trực tuyến, với mục tiêu chọn nhanh một tập nhỏ các đỉnh bao lồi đại diện cho một lớp dữ liệu mà không cần dựng bao lồi đầy đủ.

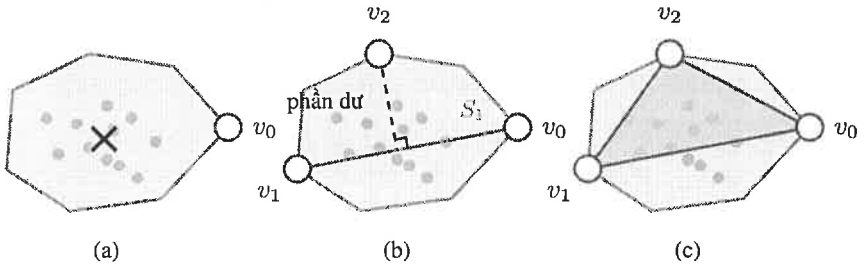
CHVS4 xây dựng dần tập đỉnh theo nguyên lý mở rộng đơn hình (simplicial expansion). Thuật toán bắt đầu từ một điểm khởi tạo, chẳng hạn điểm nằm xa tâm dữ liệu nhất. Tại mỗi bước tiếp theo, nó chọn thêm điểm nằm xa nhất so với không gian con (đơn hình) sinh bởi các đỉnh đã chọn, vì điểm xa nhất theo nghĩa này có khả năng

cao nằm trên biên của bao lồi. Quá trình lặp lại cho đến khi thu được đủ số đỉnh mong muốn.

Có thể hiểu trực giác về cơ chế này như sau: một điểm nằm sâu bên trong bao lồi luôn biểu diễn được dưới dạng tổ hợp lồi của các đỉnh, nên nó nằm gần không gian con sinh bởi các đỉnh đã chọn và có phần dư nhỏ sau khi chiếu; ngược lại, một điểm cực biên chưa được các đỉnh hiện có bao phủ sẽ để lại phần dư lớn. Vì vậy, việc luôn chọn điểm có phần dư lớn nhất sẽ ưu tiên bổ sung đúng các đỉnh mở rộng dần hình dạng của tập ứng viên theo những hướng còn thiếu, tương tự nguyên lý lấy mẫu điểm xa nhất (farthest-point sampling). Số đỉnh cần lấy được khống chế bằng một ngân sách k_H định trước, nên chi phí của thuật toán có thể kiểm soát được. Cách làm này không cần dựng toàn bộ các mặt của bao lồi nên tránh được chi phí $O(n^{\lfloor d/2 \rfloor})$; thay vào đó, chi phí chỉ ở mức khoảng $O(n_c \cdot k_H \cdot d)$ và có thể vector hóa hiệu quả. Cụ thể, gọi v_0 là điểm khởi tạo và B_t là ma trận các vector cơ sở của không gian con sinh bởi các đỉnh đã chọn ở bước t; khoảng cách từ một điểm x tới không gian con này là độ dài phần dư sau khi chiếu trực giao:

$$\mathrm{dist}(x, S_t) = \|(x - v_0) - B_t(B_t^\top B_t)^{-1} B_t^\top (x - v_0)\|_2, \tag{2.15}$$

và ở mỗi bước CHVS4 chọn điểm có $\mathrm{dist}(x, S_t)$ lớn nhất, cho đến khi thu đủ k_H đỉnh. Hình 2.3 minh họa cơ chế này qua ba bước.



Mở rộng đơn hình: mỗi bước bổ sung đỉnh có phần dư lớn nhất so với không gian con hiện tại. Dấu nhân đánh dấu tâm dữ liệu.

Hình 2.3. Cơ chế mở rộng đơn hình của thuật toán CHVS4. Bắt đầu từ điểm khởi tạo v_0 (đỉnh xa tâm dữ liệu nhất), mỗi bước bổ sung đỉnh có phần dư lớn nhất so với không gian con sinh bởi các đỉnh đã chọn; các đỉnh chọn ra đều nằm trên bao lồi.

Toàn bộ cơ chế mở rộng đơn hình mô tả ở trên được trình bày dưới dạng giả mã trong Thuật toán 1, dựa theo CHVS4 [6]. Điều làm cho thuật toán chạy nhanh nằm ở cách cập nhật phần dư: thay vì mỗi bước dựng lại phép chiếu trực giao đầy đủ theo công thức khoảng cách phần dư đã nêu, CHVS4 duy trì một *cơ sở trực chuẩn* của không gian con sinh bởi các đỉnh đã chọn và mở rộng cơ sở này tăng dần theo kiểu Gram–Schmidt. Nhờ phần dư u_i của mỗi điểm luôn được giữ ở dạng đã trừ hết thành

phần nằm trong không gian con hiện có, vector cực biên vừa được chọn tự động trực giao với cơ sở cũ; do đó chỉ cần chuẩn hóa nó thành một vector cơ sở mới rồi trừ thành phần theo hướng đó khỏi phần dư của các điểm còn lại, thay vì tính lại toàn bộ phép chiếu.

Thuật toán 1 CHVS4: tìm xấp xỉ đỉnh bao lồi

Require:
Tập gradient của lớp $G_c = \{g_i \mid i \in V_c\}$ (kích thước n_c, số chiều C)
Ngân sách ứng viên k_H
Ensure:
Tập chỉ số các đỉnh xấp xỉ bao lồi H (với $|H| \leq k_H$)
$\bar{g}_c \leftarrow \frac{1}{n_c} \sum_{i \in V_c} g_i$ ▷ tâm dữ liệu của lớp
$v_0 \leftarrow \arg\max_{i \in V_c} \|g_i - \bar{g}_c\|_2$ ▷ đỉnh xa tâm nhất làm điểm khởi tạo
$H \leftarrow \{v_0\}$
$B \leftarrow [\,]$ ▷ cơ sở trực chuẩn của không gian con, ban đầu rỗng
for $i \in V_c$ **do**
    $u_i \leftarrow g_i - g_{v_0}$ ▷ dời gốc tọa độ về đỉnh v_0
    $\rho_i \leftarrow \|u_i\|_2$ ▷ phần dư ban đầu (khoảng cách tới v_0)
end for
for $t \leftarrow 1$ to $k_H - 1$ **do**
    $v \leftarrow \arg\max_{i \in V_c \setminus H} \rho_i$ ▷ điểm có phần dư lớn nhất
    if $\rho_v = 0$ **then**
        break ▷ mọi điểm còn lại đã nằm trong không gian con hiện tại
    end if
    $H \leftarrow H \cup \{v\}$
    $b \leftarrow u_v / \rho_v$ ▷ vector cơ sở mới, đã trực giao với cơ sở hiện có
    $B \leftarrow [\,B \;\; b\,]$ ▷ mở rộng cơ sở trực chuẩn tăng dần
    for $i \in V_c \setminus H$ **do**
        $u_i \leftarrow u_i - (b^\top u_i)\, b$ ▷ trừ thành phần theo hướng b
        $\rho_i \leftarrow \|u_i\|_2$ ▷ cập nhật phần dư
    end for
end for
return H

Điều kiện dừng phụ $\rho_v = 0$ phản ánh một thực tế hình học: khi mọi điểm còn lại đều nằm trong không gian con sinh bởi các đỉnh đã chọn, chúng đều biểu diễn được dưới dạng tổ hợp của các đỉnh đó và không còn điểm cực biên mới để bổ sung; đây cũng là hệ quả trực tiếp của định lý Carathéodory đã nêu ở Mục 2.2.1. Trong điều kiện bình thường, khi ngân sách ứng viên k_H nhỏ hơn số chiều nội tại của đám mây gradient, thuật toán dừng đúng lúc thu đủ k_H đỉnh.

Chi phí của CHVS4 được quyết định bởi bước cập nhật phần dư. Giai đoạn khởi tạo tính tâm dữ liệu $\bar{g}_c$, chọn đỉnh xa tâm nhất và lập phần dư ban đầu cho n_c điểm; cả ba đều là các phép duyệt tuyến tính trên vector số chiều C, nên tốn $O(n_c \cdot C)$. Ở mỗi vòng lặp tiếp theo, thao tác tìm điểm có phần dư lớn nhất chỉ tốn $O(n_c)$; việc chuẩn hóa vector cơ sở mới tốn $O(C)$; còn việc cập nhật phần dư cho các điểm còn lại — với mỗi điểm là một tích vô hướng $b^\top u_i$, một phép trừ vector và một phép lấy chuẩn, đều trên số chiều C — tốn $O(n_c \cdot C)$. Như vậy mỗi vòng lặp có chi phí $O(n_c \cdot C)$,

và vì thuật toán lặp không quá $k_H - 1$ lần để thu đủ k_H đỉnh, tổng chi phí của CHVS4 là $O(n_c \cdot k_H \cdot C)$, tức tuyến tính theo số điểm của lớp.

Điểm mấu chốt để đạt được cận này chính là việc duy trì cơ sở trực chuẩn tăng dần. Nếu ở mỗi bước dựng lại phép chiếu trực tiếp từ dạng tổng quát $B_t(B_t^\top B_t)^{-1}B_t^\top$, thì riêng việc chiếu n_c điểm lên không gian con t chiều đã tốn $O(n_c \cdot t \cdot C)$, chưa kể chi phí nghịch đảo ma trận $B_t^\top B_t$; cộng dồn qua k_H bước sẽ cho $O(n_c \cdot k_H^2 \cdot C)$, đắt hơn một bậc k_H. Cách cập nhật tăng dần theo kiểu Gram–Schmidt loại bỏ đúng thừa số k_H dư thừa này, vì mỗi bước chỉ trừ đi thành phần theo một hướng cơ sở mới. Về bộ nhớ, thuật toán chỉ cần lưu các vector phần dư hiện thời (cỡ $n_c \times C$) và ma trận cơ sở (cỡ $k_H \times C$, với $k_H \ll n_c$), nên nhu cầu bộ nhớ ở mức $O(n_c \cdot C)$. So với chi phí $O(n^{\lfloor d/2 \rfloor})$ của việc dựng bao lồi chính xác đã phân tích ở Mục 2.2.1, chi phí tuyến tính này giải thích vì sao CHVS4 khả thi trong không gian gradient nhiều chiều. Chi phí của riêng bước lọc bao lồi này về sau được gộp cùng chi phí của bước lựa chọn coreset để cho ra tổng chi phí của CRAIG-CH, phân tích đầy đủ ở Mục 2.4.

Ta định nghĩa một ngân sách k_H cho tập ứng viên, ví dụ $k_H = \alpha \cdot r$ (với r là kích thước coreset cuối cùng và $\alpha > 1$, ví dụ $\alpha = 3$). Sau đó, ta chạy CHVS4 để tìm k_H đỉnh đầu tiên:

$$H = \text{CHVS4}(G_c, \text{budget} = k_H), \tag{2.16}$$

trong đó $G_c = \{\nabla f_i(w) \mid i \in \text{lớp } c\}$ là tập các xấp xỉ gradient của các điểm trong lớp c. Tập H (kích thước $k_H \ll n_c$) chính là tập ứng viên.

2.3.2 *Bước 2: Thực thi CRAIG trên tập ứng viên H*

Đây là bước cải tiến chính. Mô hình thực thi thuật toán tham lam của CRAIG để chọn ra r điểm coreset cuối cùng ($r < k_H$), nhưng vòng lặp tìm kiếm $\arg\max$ chỉ chạy trên H thay vì trên toàn bộ điểm của lớp. Sau khi chọn xong, mỗi điểm j được gán trọng số γ_j bằng số điểm trong lớp có gradient gần j nhất; trọng số này bảo đảm tổng gradient có trọng số của coreset xấp xỉ tổng gradient của cả lớp, đúng theo CRAIG.

Thuật toán được mô tả trong Thuật toán 2.

Thuật toán 2 CRAIG-CH (cho một lớp)

Require:
Tập gradient của lớp $G_c = \{g_i \mid i \in V_c\}$ (kích thước n_c)
Kích thước coreset r; ngân sách ứng viên $k_H > r$ (ví dụ $k_H = 3r$)
Khoảng cách Euclidean $d(g_i, g_j) = \|g_i - g_j\|_2$
Ensure:
Tập chỉ số coreset S (kích thước r) và bộ trọng số $\{\gamma_j\}_{j \in S}$

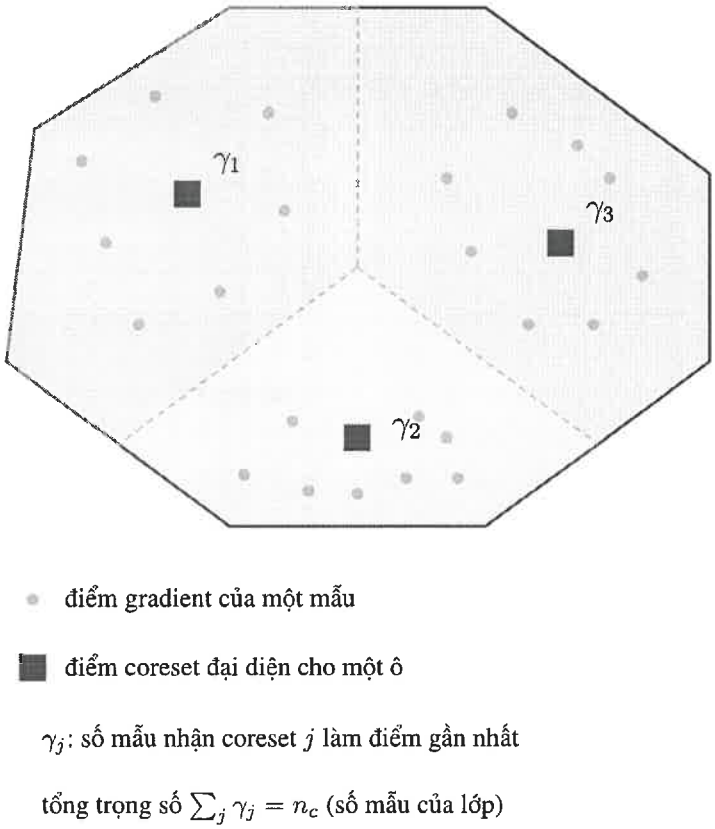
▷ Bước 1: Tìm tập ứng viên H bằng CHVS4
$H \leftarrow \text{CHVS4}(G_c, \text{budget} = k_H)$ ▷ $|H| = k_H$
▷ Bước 2: Chạy CRAIG tham lam trên H
$S \leftarrow \emptyset$
$d_{\max} \leftarrow \max_{i,j \in V_c} d(g_i, g_j)$
$L(S) \leftarrow n_c \cdot d_{\max}$ ▷ khởi tạo qua phần tử phụ trợ s_0
for $k \leftarrow 1$ to r **do**
    $j^* \leftarrow$ **nil**
    $\Delta_{\max} \leftarrow -\infty$
    for $j \in H \setminus S$ **do** ▷ duyệt tìm phần tử tốt nhất trong H
        $L(S \cup \{j\}) \leftarrow \sum_{i \in V_c} \min_{s \in S \cup \{j\}} d(g_i, g_s)$
        $\Delta_j \leftarrow L(S) - L(S \cup \{j\})$
        if $\Delta_j > \Delta_{\max}$ **then**
            $\Delta_{\max} \leftarrow \Delta_j$
            $j^* \leftarrow j$
        end if
    end for
    $S \leftarrow S \cup \{j^*\}$
    $L(S) \leftarrow L(S \cup \{j^*\})$
end for
▷ Gán trọng số theo đỉnh coreset gần nhất
for $j \in S$ **do**
    $\gamma_j \leftarrow |\{\, i \in V_c : j = \arg\min_{s \in S} d(g_i, g_s) \,\}|$
end for
return S, $\{\gamma_j\}_{j \in S}$

2.3.3 *Trọng số coreset và hàm mất mát có trọng số*

Việc gán trọng số là một phần không thể tách rời của phương pháp, giúp coreset thay thế được toàn bộ tập dữ liệu trong quá trình huấn luyện. Sau khi bước tham lam chọn xong tập chỉ số S gồm r điểm, mỗi điểm coreset $j \in S$ được gán một trọng số γ_j bằng số điểm trong lớp nhận j làm điểm coreset gần nhất trong không gian gradient:

$$\gamma_j = |\{\, i \in V_c : j = \arg\min_{s \in S} d(g_i, g_s) \,\}|. \tag{2.17}$$

Như vậy, tập S phân hoạch cả lớp thành r nhóm, mỗi điểm coreset đại diện cho nhóm điểm gần nó nhất, và tổng các trọng số đúng bằng số điểm của lớp: $\sum_{j \in S} \gamma_j = n_c$. Hình 2.4 minh họa cách phân hoạch và gán trọng số.



Hình 2.4. Trọng số coreset. Coreset phân hoạch cả lớp thành r nhóm theo điểm coreset gần nhất; mỗi điểm coreset j (ô vuông đỏ) mang trọng số γ_j bằng số điểm của nhóm, và tổng các trọng số bằng số điểm của lớp.

Trong quá trình huấn luyện, mô hình không dùng hàm mất mát trung bình thông thường trên coreset mà dùng hàm mất mát có trọng số:

$$L_S(w) = \sum_{j \in S} \gamma_j \, f_j(w), \tag{2.18}$$

nghĩa là gradient của mỗi điểm coreset được nhân với trọng số γ_j của nó. Đây chính là điều kiện để tổng gradient có trọng số của coreset xấp xỉ tổng gradient của cả lớp, đúng theo bài toán tối ưu ban đầu: $\sum_{j \in S} \gamma_j \nabla f_j(w) \approx \sum_{i \in V_c} \nabla f_i(w)$. Nếu bỏ qua trọng số và coi mọi điểm coreset như nhau, một điểm đại diện cho một nhóm đông sẽ bị tính thiếu, khiến ước lượng gradient bị lệch và làm sai lệch hướng cập nhật tham số. Vì vậy, trọng số γ_j giữ cho gradient ước lượng từ coreset không thiên lệch về độ lớn, và là yếu tố quyết định để một coreset nhỏ vẫn dẫn hướng huấn luyện đúng như toàn bộ tập dữ liệu.

2.3.4 *Ví dụ minh họa*

Để thấy rõ ba bước của CRAIG-CH phối hợp với nhau ra sao, ta xét một ví dụ số nhỏ. Giả sử bài toán chỉ có hai lớp, khi đó vector gradient theo logits của mỗi mẫu có số chiều $C = 2$ và nằm trong mặt phẳng, thuận tiện cho việc hình dung. Xét một lớp gồm $n_c = 8$ điểm gradient $g_1, \ldots, g_8$ với tọa độ cho trong Bảng 2.1; bốn điểm g_1, g_2, g_3, g_4 nằm ở biên và tạo thành bao lồi của lớp, còn bốn điểm $g_5, \ldots, g_8$ nằm bên trong đám mây gradient. Ta lấy kích thước coreset $r = 2$ và ngân sách ứng viên $k_H = 4$ (ở đây chọn nhỏ hơn giá trị mặc định $\alpha = 3$ để ví dụ được gọn, nhưng vẫn giữ $k_H > r$). Mọi khoảng cách dưới đây là khoảng cách Euclid trong không gian gradient, làm tròn đến hai chữ số thập phân.

- Bước 1 – lọc bao lồi bằng CHVS4: thuật toán xây dựng dần tập ứng viên theo nguyên lý mở rộng đơn hình. Bước khởi tạo chọn điểm xa tâm dữ liệu nhất; tâm của lớp nằm ở (2.63, 1.63) và điểm xa nhất là g_2 (khoảng cách 3.75), nên $v_0 = g_2$. Kế tiếp, điểm xa g_2 nhất là g_4 (khoảng cách 6.71), nên $v_1 = g_4$. Ở bước thứ ba, phần dư so với đường thẳng nối g_2 và g_4 đạt lớn nhất tại g_3 (bằng 3.13), nên $v_2 = g_3$. Đến đây ba đỉnh đã sinh ra toàn bộ mặt phẳng, nên phần dư theo không gian con không còn phân biệt được các điểm; theo đúng tinh thần ưu tiên điểm chưa được bao phủ, thuật toán chọn điểm còn nằm xa bao lồi hiện có nhất (xa tam giác $g_2g_4g_3$ nhất), đó là g_1 (khoảng cách 2.68). Kết quả, tập ứng viên là $H = \{g_1, g_2, g_3, g_4\}$, đúng bằng bốn đỉnh bao lồi của lớp, còn bốn điểm bên trong bị loại khỏi không gian tìm kiếm.

- Bước 2 – lựa chọn coreset bằng bước tham lam trên H: thuật toán tham lam của CRAIG chỉ duyệt trên H thay vì cả tám điểm. Ở vòng thứ nhất, nó chọn điểm làm cho tổng khoảng cách gán $L(S) = \sum_{i \in V_c} \min_{s \in S} d(g_i, g_s)$ nhỏ nhất; tính trên cả tám điểm, tổng này bằng 27.24 với g_1, 32.19 với g_2, 29.91 với g_3 và 26.64 với g_4, nên vòng thứ nhất chọn g_4 vì cho lợi ích cận biên lớn nhất. Ở vòng thứ hai, khi g_4 đã có trong coreset, việc thêm g_2 làm L giảm còn 18.51, thêm g_3 còn 18.62, thêm g_1 còn 21.90; do đó vòng thứ hai chọn g_2 (vừa nhỉnh hơn g_3). Coreset cuối cùng là $S = \{g_2, g_4\}$, gồm hai đỉnh bao lồi nằm cách xa nhau nên cùng bao quát tốt độ trải rộng của lớp.

- Bước 3 – gán trọng số: mỗi điểm của lớp được quy về điểm coreset gần nó nhất, qua đó phân hoạch lớp thành hai nhóm. Các điểm g_2, g_3, g_6 gần g_2 hơn, còn g_1, g_4, g_5, g_7, g_8 gần g_4 hơn (xem cột cuối của Bảng 2.1). Trọng số của mỗi điểm coreset bằng kích thước nhóm mà nó đại diện: $\gamma_{g_2} = 3$ và $\gamma_{g_4} = 5$. Tổng trọng số $\gamma_{g_2} + \gamma_{g_4} = 3 + 5 = 8 = n_c$, đúng bằng số điểm của lớp, nhờ đó gradient có

trọng số của coreset thay được cho tổng gradient của cả lớp. Ở ví dụ này g_4 đại diện cho một nhóm đông hơn nên mang trọng số lớn hơn; nếu bỏ qua trọng số và coi hai điểm coreset như nhau, đóng góp của nhóm quanh g_4 sẽ bị tính thiếu và hướng cập nhật tham số bị lệch.

Bảng 2.1. Ví dụ minh họa: tám điểm gradient của một lớp, tập ứng viên H do CHVS4 lọc ra, và cách phân nhóm để gán trọng số coreset.

Điểm	Tọa độ	Thuộc H	Vai trò	Coreset gần nhất
g_1	(0, 0)	có	đỉnh	g_4
g_2	(6, 0)	có	coreset	g_2
g_3	(5, 4)	có	đỉnh	g_2
g_4	(0, 3)	có	coreset	g_4
g_5	(2, 1)	không	điểm trong	g_4
g_6	(3, 1)	không	điểm trong	g_2
g_7	(2, 2)	không	điểm trong	g_4
g_8	(3, 2)	không	điểm trong	g_4

Ghi chú: coreset thu được là $S = \{g_2, g_4\}$; trọng số $\gamma_{g_2} = 3$ (các điểm g_2, g_3, g_6) và $\gamma_{g_4} = 5$ (các điểm g_1, g_4, g_5, g_7, g_8), thỏa mãn $\gamma_{g_2} + \gamma_{g_4} = n_c = 8$.

Ví dụ trên cho thấy cơ chế phối hợp ba bước: CHVS4 thu tám điểm về bốn ứng viên cực biên, bước lựa chọn coreset chọn ra hai điểm đại diện tốt trong số các ứng viên đó, và trọng số bảo toàn đúng số điểm của lớp, tương ứng với những gì Hình 2.3 và Hình 2.4 mô tả bằng hình vẽ. Các con số ở đây chỉ nhằm làm rõ cơ chế trên một cấu hình nhỏ, không phải kết quả thực nghiệm; kết quả trên dữ liệu thật được trình bày ở Chương 3.

2.4 Phân tích độ phức tạp và tài nguyên

Mục 2.1.3 đã ước lượng số lần đánh giá lợi ích cận biên của bước tham lam; phần này phân tích đầy đủ chi phí của CRAIG-CH cho một lần lựa chọn coreset trên một lớp dữ liệu có n_c điểm, trong đó C là số chiều của vector gradient (bằng số lớp của bài toán phân loại), k_H là ngân sách ứng viên và r là kích thước coreset. Một lần lựa chọn gồm bốn công đoạn:

- Tính đặc trưng gradient: mỗi điểm cần một vector gradient theo logits, thu được từ một lượt lan truyền tiến, với chi phí $O(n_c \cdot C)$. Chi phí này nhỏ so với một vòng huấn luyện đầy đủ.
- Lọc bao lồi bằng CHVS4 (Bước 1): thuật toán lặp k_H lần; mỗi lần chiếu n_c điểm lên không gian con sinh bởi các đỉnh đã chọn rồi lấy điểm có phần dư lớn nhất, tốn $O(n_c \cdot C)$ mỗi lần nhờ cập nhật cơ sở trực giao tăng dần. Tổng chi phí là $O(n_c \cdot k_H \cdot C)$.
- Tính khoảng cách điểm–ứng viên: để chạy bước tham lam, cần khoảng cách

gradient từ mỗi trong n_c điểm tới mỗi trong k_H ứng viên, với chi phí $O(n_c \cdot k_H \cdot C)$.

- Lựa chọn tham lam trên H (Bước 2): thực hiện r vòng; mỗi vòng đánh giá lợi ích cận biên của tối đa k_H ứng viên, mỗi lần đánh giá cập nhật khoảng cách nhỏ nhất trên n_c điểm, nên tốn $O(n_c \cdot k_H)$ mỗi vòng. Tổng chi phí là $O(n_c \cdot k_H \cdot r)$; thuật toán tham lam trễ (lazy greedy) có thể giảm số lần đánh giá thực tế nhưng không đổi cận trên này.

Gộp lại, chi phí của bước lựa chọn CRAIG-CH là $O(n_c \cdot k_H \cdot (C+r))$. Với ngân sách $k_H = \alpha r$, chi phí này trở thành $O(\alpha \cdot n_c \cdot r \cdot (C+r))$, tức tuyến tính theo số điểm n_c của lớp.

So sánh với CRAIG gốc: khi không có bước lọc, bước tham lam phải xét toàn bộ n_c điểm làm ứng viên ở mỗi vòng, nên chi phí là $O(n_c^2 \cdot (C+r))$ — bậc hai theo n_c. Ngược lại, CHVS4 dùng đơn lẻ (dừng ngay khi đủ r đỉnh) chỉ tốn $O(n_c \cdot r \cdot C)$, rẻ nhất nhưng đánh đổi bằng chất lượng coreset thấp hơn do không có bước tối ưu độ bao phủ. CRAIG-CH nằm giữa hai thái cực này. Bảng 2.2 tổng hợp so sánh.

Bảng 2.2. So sánh chi phí của bước lựa chọn coreset và bộ nhớ khoảng cách trên một lớp dữ liệu (cho một lần lựa chọn)

Phương pháp	Thời gian bước lựa chọn	Bộ nhớ khoảng cách
CHVS4 (dừng ở r)	$O(n_c \cdot r \cdot C)$	$O(n_c \cdot r)$
CRAIG	$O(n_c^2 \cdot (C+r))$	$O(n_c^2)$
CRAIG-CH	$O(n_c \cdot k_H \cdot (C+r))$	$O(n_c \cdot k_H)$

Ghi chú: n_c là số điểm của lớp, C là số chiều gradient, r là kích thước coreset, $k_H = \alpha r$ là ngân sách ứng viên với $k_H \ll n_c$.

Về bộ nhớ, điểm khác biệt quan trọng nằm ở ma trận khoảng cách. Nếu lưu toàn bộ khoảng cách để tăng tốc bước tham lam, CRAIG cần ma trận $n_c \times n_c$, tức $O(n_c^2)$; với một lớp cỡ vài nghìn đến vài chục nghìn điểm, con số này lên tới hàng chục đến hàng trăm triệu phần tử, gây áp lực lớn lên bộ nhớ. CRAIG-CH chỉ cần ma trận khoảng cách điểm–ứng viên cỡ $n_c \times k_H$, tức $O(n_c \cdot k_H)$, nhỏ hơn nhiều lần. Ưu điểm này giúp CRAIG-CH áp dụng được cho các lớp có nhiều điểm mà CRAIG gặp khó về bộ nhớ.

Đối với mô hình học sâu, khoảng cách gradient thay đổi theo tham số nên coreset được chọn lại định kỳ. Nếu chọn lại E lần trong toàn bộ quá trình huấn luyện, mọi chi phí phân tích ở trên đều nhân với E, nên phần chi phí mà CRAIG-CH tiết kiệm được so với CRAIG cũng được khuếch đại theo số lần chọn lại. Đây là lý do bước lựa chọn coreset trở thành một thành phần đáng kể trong tổng thời gian huấn luyện, và việc giảm chi phí của nó có ý nghĩa thực tiễn rõ rệt.

2.5 Lựa chọn tham số và bảo đảm lý thuyết

2.5.1 *Lựa chọn ngân sách ứng viên k_H*

Ngân sách ứng viên $k_H = \alpha r$ là tham số điều khiển chính của CRAIG-CH, quyết định sự đánh đổi giữa chi phí và chất lượng coreset. Có thể hình dung vai trò của k_H qua hai trường hợp giới hạn:

- Khi $k_H \to r$ (tức $\alpha \to 1$): tập ứng viên vừa đúng bằng kích thước coreset, nên bước tham lam gần như không còn quyền lựa chọn và CRAIG-CH suy biến thành CHVS4 dừng sớm. Chi phí thấp nhất nhưng chất lượng coreset cũng thấp nhất.
- Khi $k_H \to n_c$ (tức α rất lớn): tập ứng viên bao trùm gần như toàn bộ lớp, nên bước tham lam tìm kiếm trên gần như toàn bộ dữ liệu và CRAIG-CH suy biến thành CRAIG gốc. Chất lượng coreset cao nhất nhưng chi phí cũng cao nhất.

Như vậy, k_H cho phép điều chỉnh liên tục vị trí của CRAIG-CH giữa CHVS4 và CRAIG. Một giá trị α nhỏ hơn nhiều so với tỉ lệ n_c/r nhưng đủ lớn hơn 1 sẽ cho bước tham lam đủ ứng viên để chọn ra một tập đại diện đa dạng, trong khi vẫn giữ chi phí thấp hơn hẳn CRAIG. Trong đề án, giá trị $\alpha = 3$ (tức $k_H = 3r$) được chọn làm mặc định vì cân bằng được hai yêu cầu: tập ứng viên đủ lớn để bước tham lam có thể cải thiện chất lượng so với CHVS4 dừng sớm, đồng thời vẫn nhỏ hơn nhiều lần so với số điểm của lớp nên chi phí thấp hơn hẳn CRAIG. Tăng α sẽ đẩy chất lượng coreset tiến gần CRAIG hơn nhưng đồng thời làm tăng chi phí của bước lựa chọn; các thực nghiệm ở Chương 3 được thực hiện với giá trị mặc định này.

2.5.2 *Bảo đảm lý thuyết được kế thừa*

Vì bước tham lam của CRAIG-CH vẫn tối đa hóa hàm vị trí cơ sở — một hàm dưới mô-đun đơn điệu — chỉ khác là trên tập ứng viên H thay vì toàn bộ lớp, nên bảo đảm của thuật toán tham lam vẫn được giữ nguyên. Theo Định lý 1.1 ở Chương 1, coreset thu được đạt ít nhất $(1 - 1/e)$ giá trị của tập con r phần tử tốt nhất lấy trong chính tập ứng viên H.

Từ đó, có thể phân rã sai số xấp xỉ gradient của CRAIG-CH thành hai thành phần: một là sai số do chỉ tìm kiếm trong tập ứng viên H thay vì toàn bộ lớp, hai là phần dư của bước tham lam trong H (được chặn bởi bảo đảm $(1 - 1/e)$). Thành phần thứ nhất nhỏ khi H bao phủ tốt các hướng cực biên của đám mây gradient — đúng mục tiêu mà bước lọc CHVS4 hướng tới. Vì $H \subseteq V_c$, tập con tốt nhất trong H không thể tốt hơn tập con tốt nhất trong cả lớp, nên chất lượng coreset của CRAIG-CH bị chặn trên bởi chất lượng của CRAIG. Điều này giải thích một cách nhất quán

vì sao, về mặt lý thuyết, chất lượng của CRAIG-CH được kỳ vọng nằm giữa CHVS4 và CRAIG: tốt hơn CHVS4 dừng sớm nhờ bước tối ưu độ bao phủ, nhưng khó vượt CRAIG vì không gian tìm kiếm đã bị thu hẹp.

Cuối cùng, do coreset có trọng số vẫn xấp xỉ gradient toàn cục với một sai số ϵ xác định, bảo đảm hội tụ của thuật toán IG trên coreset đã nêu ở Mục 2.1.3 vẫn áp dụng cho CRAIG-CH, với kích thước vùng lân cận điểm tối ưu phụ thuộc vào sai số ϵ của chính CRAIG-CH.

2.6 Ưu điểm kỳ vọng và sự đánh đổi

Phương pháp CRAIG-CH được kỳ vọng mang lại các lợi ích sau so với CRAIG và CHVS4 (dừng sớm):

- Giảm chi phí của bước lựa chọn so với CRAIG: như phân tích ở Mục 2.4, chi phí của bước lựa chọn giảm từ bậc hai $O(n_c^2 \cdot (C+r))$ xuống bậc một $O(n_c \cdot k_H \cdot (C+r))$ theo số điểm của lớp, đồng thời nhu cầu bộ nhớ cho ma trận khoảng cách giảm từ $O(n_c^2)$ xuống $O(n_c \cdot k_H)$. Bước lọc CHVS4 có thêm chi phí nhưng vẫn tuyến tính theo n_c, nên tổng thời gian của bước lựa chọn được kỳ vọng nhỏ hơn CRAIG.
- Cải thiện chất lượng đại diện so với CHVS4 dừng sớm: CHVS4 chỉ tìm các đỉnh cho đến khi đủ ngân sách r, và các đỉnh này có thể nằm gần nhau trên bề mặt bao lồi. Bằng cách chạy bước tham lam của CRAIG trên k_H ứng viên, CRAIG-CH chọn ra r điểm đại diện tốt hơn cho toàn bộ n_c điểm dữ liệu, thay vì chỉ lấy các đỉnh đầu tiên.
- Chất lượng coreset nằm giữa CHVS4 và CRAIG: vì bước tham lam chỉ tìm trong tập ứng viên k_H chứ không phải toàn bộ n_c điểm, sai số xấp xỉ gradient của CRAIG-CH được kỳ vọng thấp hơn CHVS4 dừng sớm và tiệm cận CRAIG, nhưng khó thấp hơn CRAIG. Đây là sự đánh đổi giữa chi phí lựa chọn và chất lượng coreset. Các kỳ vọng này được kiểm chứng bằng thực nghiệm ở Chương 3.

2.6.1 Hạn chế và phạm vi áp dụng

Lợi ích của bước lọc bao lồi không đồng đều trên mọi tình huống, mà phụ thuộc vào cấu trúc hình học của đám mây gradient và vào quy mô của từng lớp. CRAIG-CH phát huy tác dụng rõ nhất khi đám mây gradient của một lớp có cấu trúc cực biên tách bạch, tức phần lớn các điểm nằm bên trong và chỉ một số ít điểm cực biên định hình nên bao lồi. Trong trường hợp này, tập ứng viên H do CHVS4 dựng lên giữ lại đúng những điểm cực biên mang hướng gradient riêng biệt, còn hàng loạt điểm nội tâm gần như trùng nhau về gradient bị loại khỏi vòng tìm kiếm mà không làm mất thông

tin đáng kể. Lợi ích càng lớn khi số điểm của lớp n_c lớn: khi đó việc hạ chi phí của bước lựa chọn coreset từ bậc hai theo n_c xuống bậc một, cùng với việc thu nhỏ ma trận khoảng cách từ $O(n_c^2)$ xuống $O(n_c \cdot k_H)$, tạo ra khác biệt thực sự về thời gian và bộ nhớ, đúng như phân tích ở Mục 2.4.

Ngược lại, có những tình huống bước lọc mang lại ít lợi ích, thậm chí không đáng để áp dụng. Thứ nhất, khi số chiều gradient C rất lớn, chẳng hạn bài toán phân loại có nhiều lớp, thì trong không gian nhiều chiều hầu hết các điểm đều có xu hướng nằm gần biên, khiến khái niệm đỉnh bao lồi mất dần khả năng phân biệt; lúc này bước lọc khó tách được các điểm thực sự giàu thông tin khỏi phần còn lại, và ưu thế của tập ứng viên bị bào mòn. Thứ hai, khi ngân sách ứng viên k_H được đặt quá nhỏ, tập ứng viên có thể bỏ sót những hướng cực biên quan trọng, làm coreset mất đa dạng và đẩy sai số xấp xỉ gradient lên cao; đây là mặt trái của chính cơ chế thu hẹp không gian tìm kiếm, và cũng là giới hạn dưới của việc giảm ngân sách ứng viên đã bàn ở Mục 2.5.1. Thứ ba, khi n_c nhỏ, chi phí của bước lựa chọn coreset vốn đã thấp, nên phần chi phí tiết kiệm được không đủ bù cho chi phí phụ trội của bước lọc bao lồi, khiến cải tiến gần như không còn ý nghĩa thực tiễn.

Cần nhắc lại, như đã nêu ở Mục 2.6, rằng CRAIG-CH là một sự đánh đổi chứ không phải một cải tiến thắng ở mọi mặt: nó đổi một mức suy giảm độ chính xác có kiểm soát để lấy chi phí và bộ nhớ thấp hơn ở bước lựa chọn coreset. Điều đáng lưu ý thêm ở đây là mức đánh đổi này không cố định mà dịch chuyển theo đúng các điều kiện vừa nêu: nó có lợi rõ khi lớp đông điểm và cấu trúc cực biên tách bạch, nhưng thu hẹp dần khi số chiều gradient lớn, khi ngân sách ứng viên quá nhỏ, hoặc khi lớp có ít điểm. Mức đánh đổi cụ thể giữa tốc độ và độ chính xác được định lượng qua các thực nghiệm ở Chương 3.

Từ những hạn chế trên có thể phác ra một số hướng mở rộng. Một là điều chỉnh ngân sách ứng viên k_H một cách thích nghi theo từng lớp, thay vì cố định $\alpha = 3$ cho mọi lớp: những lớp có đám mây gradient trải rộng, cấu trúc cực biên phong phú có thể cần k_H lớn hơn, trong khi những lớp gọn và đồng nhất có thể dùng k_H nhỏ hơn để tiết kiệm chi phí. Hai là kết hợp một bước giảm chiều nhẹ trước khi lọc bao lồi, nhằm giảm bớt ảnh hưởng của số chiều gradient lớn tới khả năng phân biệt của đỉnh bao lồi, đồng thời hạ thêm chi phí tính khoảng cách; hướng này cần cân nhắc rủi ro mất thông tin cực biên như đã bàn ở Mục 2.2.2 khi khảo sát nhóm phương pháp loại 1. Ba là mở rộng đánh giá sang các bài toán có số lớp lớn hơn để xác định rõ ngưỡng mà tại đó bước lọc bao lồi bắt đầu mất tác dụng, qua đó khoanh vùng phạm vi áp dụng phù hợp nhất của phương pháp. Các hướng này đều nhằm mở rộng vùng mà sự đánh đổi của CRAIG-CH còn có lợi, và là những vấn đề để ngỏ cho các nghiên cứu tiếp theo.

Kết luận chương

Chương này đã phân tích hạn chế về chi phí tính toán của thuật toán CRAIG và đề xuất một phương pháp cải tiến là CRAIG-CH, bằng cách dùng thuật toán xấp xỉ bao lồi CHVS4 làm bộ lọc tiền xử lý để thu hẹp không gian tìm kiếm. Phương pháp lai này được kỳ vọng giảm chi phí của bước lựa chọn coreset, đồng thời cải thiện chất lượng đại diện so với việc chỉ dùng đỉnh bao lồi.

Các giả thuyết về ưu điểm của CRAIG-CH (giảm thời gian của bước lựa chọn coreset, và giữ chất lượng coreset ở mức giữa CHVS4 và CRAIG) sẽ được kiểm chứng qua các thực nghiệm ở Chương 3.

CHƯƠNG 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Sau khi trình bày cơ sở lý luận (Chương 1) và đề xuất phương pháp CRAIG-CH (Chương 2), chương này kiểm chứng phương pháp bằng thực nghiệm. Mục tiêu là so sánh và đánh giá định lượng CRAIG-CH với ba phương pháp nền tảng là lấy mẫu ngẫu nhiên (Random), CRAIG, CHVS4, cùng đường cơ sở Full Dataset, trên hai bộ dữ liệu ứng với hai lớp mô hình khác nhau. Các thí nghiệm được thiết kế để trả lời ba câu hỏi. Thứ nhất, CRAIG-CH có giảm được chi phí của bước lựa chọn coreset so với CRAIG hay không, và mức giảm đó thể hiện thế nào trên tổng thời gian thực thi. Thứ hai, việc thu hẹp không gian tìm kiếm ảnh hưởng ra sao đến chất lượng coreset, đo bằng độ chính xác và sai số xấp xỉ gradient. Thứ ba, đặt cạnh các phương pháp còn lại, CRAIG-CH nằm ở đâu trong sự đánh đổi giữa tốc độ và độ chính xác.

3.1 Môi trường và dữ liệu thực nghiệm

3.1.1 Môi trường thực nghiệm

Toàn bộ thí nghiệm được thực hiện trên cùng một máy chủ để bảo đảm tính nhất quán và khả năng tái lập:

- Phần cứng: CPU Intel Xeon Gold 6542Y, 96 GB RAM, GPU NVIDIA L40S 48 GB.
- Phần mềm: hệ điều hành Ubuntu 22.04 LTS, Python 3.10, PyTorch 2.6 (CUDA 12.4).

Mỗi cấu hình được chạy với ba hạt giống ngẫu nhiên (0, 1, 2); các số liệu báo cáo là trung bình của ba lần chạy, kèm độ lệch chuẩn ở bảng độ chính xác. Coreset được chọn lại sau mỗi 20 epoch. Ngân sách ứng viên của CRAIG-CH đặt bằng $k_H = 3r$ với r là kích thước coreset.

3.1.2 Dữ liệu thực nghiệm

Hai bộ dữ liệu có đặc tính khác nhau được dùng để đánh giá:

- FashionMNIST [22]: bộ ảnh xám các sản phẩm thời trang gồm 10 lớp, tổng cộng $N = 60.000$ mẫu huấn luyện. Mỗi mẫu là ảnh 28×28, được duỗi thành vector 784 chiều. Bộ dữ liệu này dùng cho bài toán phân loại với mô hình hồi quy logistic, tương ứng với bài toán lồi.
- CIFAR-10: bộ ảnh màu gồm 50.000 ảnh huấn luyện và 10.000 ảnh kiểm thử, kích

thước 32×32, thuộc 10 lớp. Bộ dữ liệu này dùng cho bài toán phân loại với mô hình ResNet-20, tương ứng với bài toán phi lồi.

3.2 Các phương pháp so sánh

Bốn phương pháp lựa chọn coreset được đánh giá, cùng với đường cơ sở Full Dataset:

1. Full Dataset: huấn luyện trên toàn bộ dữ liệu. Đây là giới hạn trên về độ chính xác và là mốc tham chiếu về thời gian.
2. Random: lấy mẫu ngẫu nhiên. Đây là phương pháp đơn giản nhất, không cần tính đặc trưng gradient, dùng làm tham chiếu.
3. CRAIG: chạy thuật toán tham lam tối ưu hóa hàm vị trí cơ sở trên toàn bộ n điểm dữ liệu của mỗi lớp.
4. CHVS4: lựa chọn các đỉnh bao lồi trong không gian gradient, không thực hiện thêm bước tối ưu hóa.
5. CRAIG-CH (phương pháp đề xuất): dùng CHVS4 lọc ra tập ứng viên bao lồi, rồi chạy bước tham lam của CRAIG trên tập ứng viên đó.

3.3 Các tiêu chí đánh giá

Bốn tiêu chí được sử dụng:

- Độ chính xác kiểm thử (%): đo tại epoch cuối (epoch 200 cho CIFAR-10, epoch 100 cho FashionMNIST).
- Thời gian bước lựa chọn coreset (giây/epoch): chi phí của riêng thuật toán lựa chọn, đã trừ phần tính đặc trưng gradient, tính trung bình trên một epoch. Đây là đại lượng phản ánh trực tiếp đóng góp của phương pháp đề xuất.
- Tổng thời gian thực thi (giây): tổng thời gian của cả quá trình, gồm bước lựa chọn coreset và bước huấn luyện mô hình. Đại lượng này cho phép so sánh trực tiếp với phương án huấn luyện trên toàn bộ dữ liệu.
- Sai số xấp xỉ gradient đã chuẩn hóa: bằng $\|g_{\text{toàn bộ}} - g_{\text{coreset}}\| / \|g_{\text{toàn bộ}}\|$, trong đó $g_{\text{toàn bộ}}$ là tổng gradient của cả lớp và g_{coreset} là tổng gradient có trọng số của coreset. Giá trị càng nhỏ thì coreset xấp xỉ gradient càng tốt.

Cần phân biệt rõ hai tiêu chí thời gian. Thời gian bước lựa chọn chỉ đo chi phí của thuật toán chọn coreset, dùng để so sánh trực tiếp giữa các phương pháp lựa chọn với nhau, và đây là đại lượng mà phương pháp đề xuất hướng tới việc giảm. Tổng thời

gian thực thi bao gồm cả bước huấn luyện, dùng để đánh giá lợi ích thực tế so với việc huấn luyện trên toàn bộ dữ liệu. Việc tách bạch này giúp thấy được mức tăng tốc ở bước lựa chọn chuyển thành lợi ích ở tổng thời gian tới đâu.

3.4 Kết quả thực nghiệm

3.4.1 Kiểm nghiệm với mô hình hồi quy logistic trên bộ dữ liệu FashionMNIST

a, Thiết lập thực nghiệm

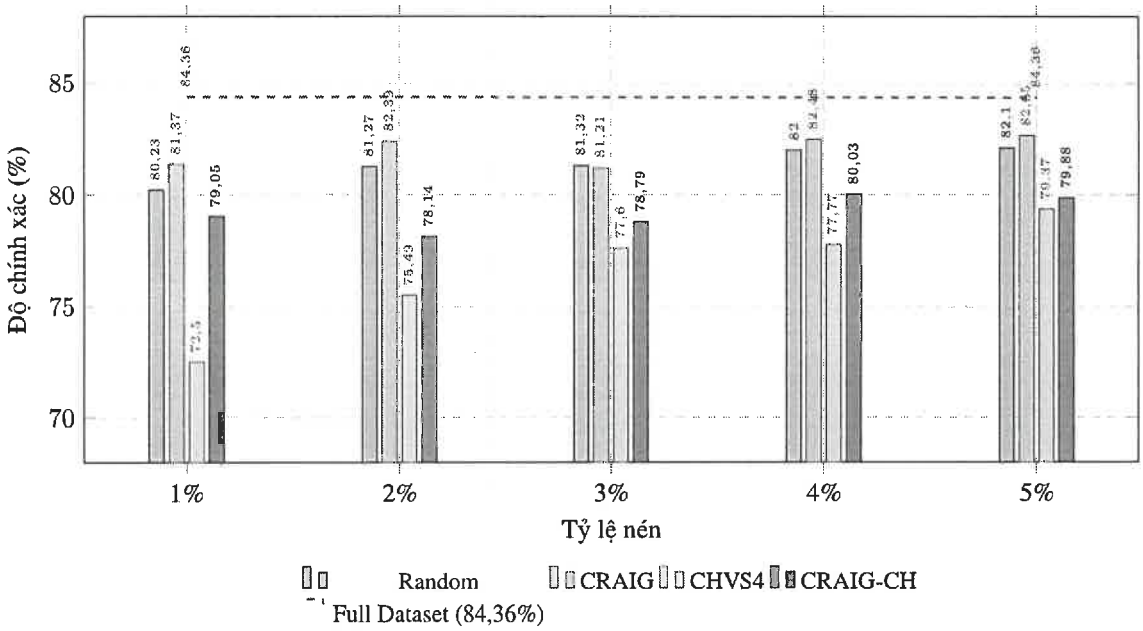
- Bộ dữ liệu: FashionMNIST (10 lớp), $N = 60.000$ mẫu huấn luyện.
- Mô hình: hồi quy logistic.
- Số epoch: 100.
- Các mức tỷ lệ nén: 1%, 2%, 3%, 4% và 5%.

b, So sánh độ chính xác

Bảng 3.1 trình bày độ chính xác kiểm thử tại epoch 100, và Hình 3.1 minh họa trực quan các số liệu này.

Bảng 3.1. Độ chính xác kiểm thử trên FashionMNIST tại epoch 100 (đơn vị: %)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH
1% (600 mẫu)	80,23	81,37	72,50	79,05
2% (1200 mẫu)	81,27	82,39	75,49	78,14
3% (1800 mẫu)	81,32	81,21	77,60	78,79
4% (2400 mẫu)	82,00	82,48	77,77	80,03
5% (3000 mẫu)	82,10	82,65	79,37	79,88
Full Dataset		84,36		



Hình 3.1. So sánh độ chính xác các phương pháp trên FashionMNIST.

Từ kết quả trên có thể rút ra một số nhận xét:

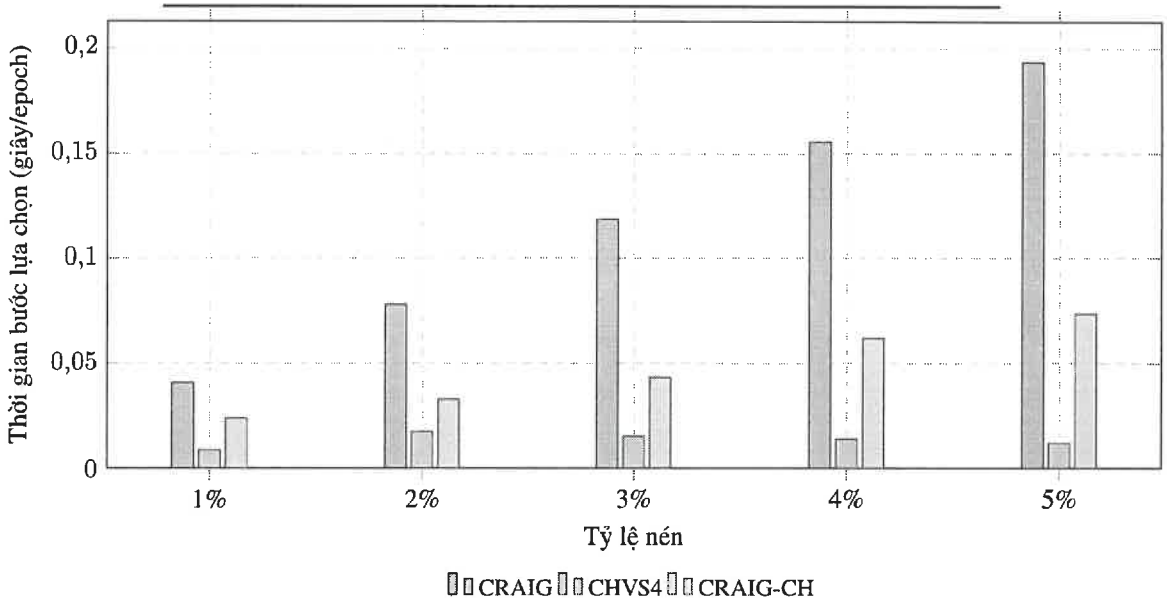
- Random và CRAIG cho độ chính xác cao nhất ở gần như mọi tỷ lệ nén, hai phương pháp bám sát nhau (khoảng 80%–83%). Đây là đặc tính của bài toán lồi: với hồi quy logistic, một tập con ngẫu nhiên đã đủ đại diện cho phân phối dữ liệu, nên bước tối ưu hóa của CRAIG chỉ cải thiện thêm chút ít.
- CRAIG-CH thấp hơn CRAIG ở mọi tỷ lệ nén, chênh lệch từ khoảng 2,3 đến 4,3 điểm phần trăm. Việc giới hạn không gian tìm kiếm về các đỉnh bao lồi khiến coreset kém đại diện hơn so với khi tìm trên toàn bộ lớp.
- CHVS4 thấp nhất, rõ nhất ở tỷ lệ nén nhỏ (72,50% tại 1%). Khi coreset rất nhỏ, các đỉnh bao lồi đơn thuần là những điểm cực biên nên chưa bao phủ được vùng trung tâm của phân phối.
- Khoảng cách tới Full Dataset (84,36%) vẫn đáng kể ở mọi tỷ lệ nén, cho thấy đây là bài toán mà việc có đủ dữ liệu vẫn giữ vai trò quan trọng.

c, So sánh thời gian

Bảng 3.2 trình bày thời gian của bước lựa chọn coreset và hệ số tăng tốc của CRAIG-CH so với CRAIG; Hình 3.2 minh họa các số liệu này.

Bảng 3.2. Thời gian bước lựa chọn coreset trên FashionMNIST (đơn vị: giây/epoch)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH	Tăng tốc
1%	0,0004	0,0409	0,0089	0,0237	1,73×
2%	0,0002	0,0783	0,0175	0,0328	2,39×
3%	0,0002	0,1184	0,0154	0,0433	2,73×
4%	0,0002	0,1556	0,0140	0,0620	2,51×
5%	0,0002	0,1934	0,0120	0,0737	2,62×



Hình 3.2. Thời gian bước lựa chọn coreset trên FashionMNIST. CRAIG-CH rẻ hơn CRAIG nhờ chỉ tìm trên tập ứng viên đã lọc.

Bảng 3.3 trình bày tổng thời gian thực thi của cả quá trình 100 epoch, đặt cạnh mốc Full Dataset là 151,3 giây.

Bảng 3.3. Tổng thời gian thực thi trên FashionMNIST, 100 epoch (đơn vị: giây)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH
1%	12,3	25,3	23,0	25,4
2%	12,8	30,4	24,7	27,1
3%	14,9	36,7	26,7	30,4
4%	16,0	41,6	30,2	33,3
5%	17,8	46,9	30,0	36,3
Full Dataset		151,3		

Từ hai bảng thời gian có thể rút ra một số nhận xét:

- Ở bước lựa chọn, CRAIG-CH nhanh hơn CRAIG từ 1,73 đến 2,73 lần. Đúng như thiết kế, việc thu hẹp không gian tìm kiếm từ toàn bộ n điểm xuống còn $k_H = 3r$ ứng viên làm giảm chi phí của thuật toán tham lam.

- CHVS4 nhanh nhất trong nhóm có tính chọn lọc vì không có bước tối ưu hóa, còn Random gần như không tốn chi phí do không cần tính đặc trưng gradient.
- Trên tổng thời gian, mức tăng tốc thu hẹp lại: CRAIG-CH nhanh hơn CRAIG khoảng 1,0 đến 1,3 lần. Nguyên nhân là chi phí lựa chọn chỉ chiếm một phần trong tổng chi phí, mà phần lớn chi phí lựa chọn lại nằm ở bước tính đặc trưng gradient, vốn giống nhau ở CRAIG và CRAIG-CH.
- So với Full Dataset (151,3 giây), mọi phương pháp coreset đều rút ngắn tổng thời gian đáng kể, từ khoảng 3 đến 12 lần. Đây là lợi ích chung của hướng tiếp cận coreset, không riêng của phương pháp đề xuất.

d, So sánh sai số xấp xỉ gradient

Để lý giải chênh lệch độ chính xác ở trên, Bảng 3.4 trình bày sai số xấp xỉ gradient đã chuẩn hóa của ba phương pháp có dùng gradient.

Bảng 3.4. Sai số xấp xỉ gradient đã chuẩn hóa trên FashionMNIST

Tỷ lệ nén	CRAIG	CHVS4	CRAIG-CH
1%	0,0187	0,0209	0,0385
2%	0,0128	0,0039	0,0071
3%	0,0028	0,0009	0,0011
4%	0,0073	0,0026	0,0031
5%	0,0063	0,0020	0,0036

Từ bảng trên có thể rút ra một số nhận xét:

- Từ tỷ lệ nén 2% trở lên, CHVS4 cho sai số xấp xỉ gradient nhỏ nhất, còn CRAIG-CH bám sát CHVS4. Điều này cho thấy bước lọc bao lồi giữ lại được cấu trúc gradient tốt trên bài toán lồi.
- Sai số xấp xỉ gradient nhỏ không kéo theo độ chính xác cao: CHVS4 có sai số nhỏ nhất nhưng độ chính xác thấp nhất. Nguyên nhân là đại lượng này đo mức khớp của tổng gradient, trong khi trọng số của coreset bao lồi tập trung vào ít điểm cực biên nên khả năng tổng quát hóa không tương xứng.

Kết luận cho hồi quy logistic với FashionMNIST

1. Trên bài toán lồi, Random và CRAIG cho độ chính xác cao nhất; CRAIG-CH và CHVS4 chấp nhận độ chính xác thấp hơn do giới hạn không gian tìm kiếm về các đỉnh bao lồi.
2. Đóng góp của CRAIG-CH nằm ở tốc độ của bước lựa chọn; tuy nhiên trên tổng thời gian, mức lợi thu hẹp lại vì bước lựa chọn chiếm tỷ trọng nhỏ trong toàn bộ

chi phí ở bài toán quy mô nhỏ này.

3. Với bài toán lồi và quy mô nhỏ, lợi ích thực tế của việc chọn lọc coreset còn hạn chế; giá trị của phương pháp thể hiện rõ hơn ở bài toán phi lồi quy mô lớn được trình bày tiếp theo.

3.4.2 Kiểm nghiệm với mô hình ResNet-20 trên bộ dữ liệu CIFAR-10

a, Thiết lập thực nghiệm

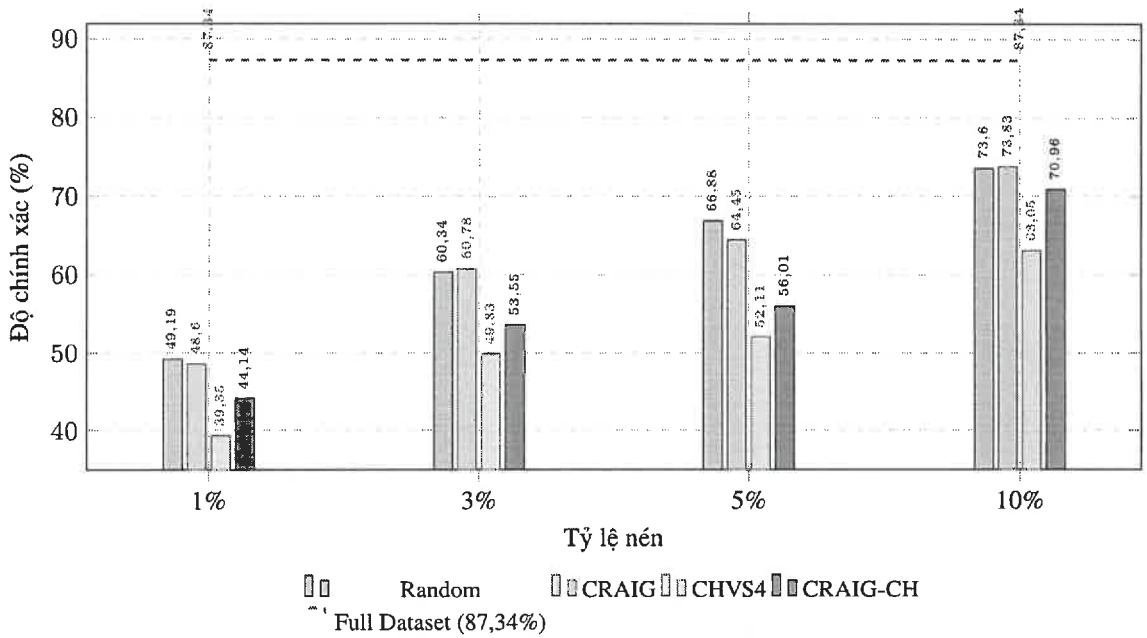
- Bộ dữ liệu: CIFAR-10, 50.000 ảnh huấn luyện.
- Mô hình: ResNet-20.
- Số epoch: 200.
- Các mức tỷ lệ nén: 1%, 3%, 5% và 10%.

b, So sánh độ chính xác

Bảng 3.5 trình bày độ chính xác kiểm thử tại epoch 200, và Hình 3.3 minh họa trực quan các số liệu này.

Bảng 3.5. Độ chính xác kiểm thử trên CIFAR-10 tại epoch 200 (đơn vị: %)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH
1%	49,19	48,60	39,35	44,14
3%	60,34	60,78	49,83	53,55
5%	66,88	64,45	52,11	56,01
10%	73,60	73,83	63,05	70,96
Full Dataset		87,34		



Hình 3.3. So sánh độ chính xác các phương pháp trên CIFAR-10.

Từ kết quả trên có thể rút ra một số nhận xét:

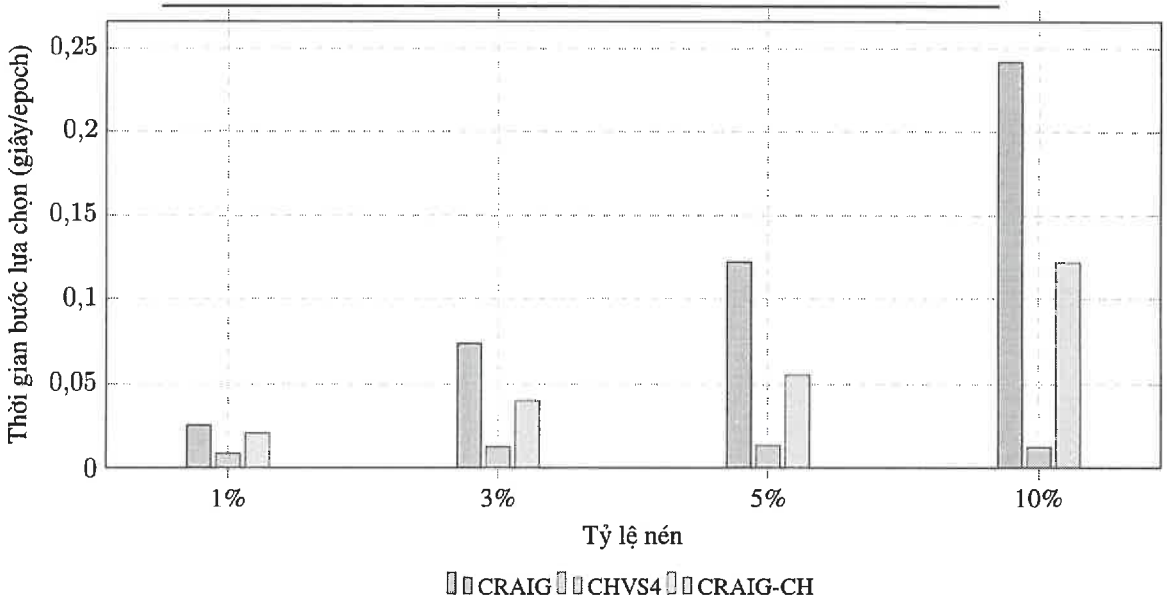
- Độ chính xác phụ thuộc mạnh vào tỷ lệ nén: mọi phương pháp đều tăng đều đặn khi tỷ lệ nén tăng từ 1% lên 10%. Điều này đúng với kỳ vọng, vì coreset càng lớn thì càng bao phủ tốt phân phối dữ liệu.
- Random và CRAIG dẫn đầu và bám sát nhau ở mọi tỷ lệ nén. Random nhỉnh hơn ở 1% và 5%, CRAIG nhỉnh hơn ở 3% và 10%; chênh lệch giữa hai phương pháp đều nhỏ.
- CRAIG-CH thấp hơn CRAIG ở mọi tỷ lệ nén, chênh lệch lớn nhất ở tỷ lệ nén 5% (8,4 điểm phần trăm) rồi thu hẹp còn 2,9 điểm phần trăm ở 10%, khi tập ứng viên bao lồi lớn hơn và bước tham lam có nhiều lựa chọn hơn.
- CHVS4 thấp nhất ở mọi tỷ lệ nén, cho thấy nếu chỉ dùng đỉnh bao lồi mà không có bước tối ưu hóa thì chất lượng coreset chưa đủ; bước tham lam trong CRAIG-CH cải thiện rõ so với CHVS4.

c, So sánh thời gian

Bảng 3.6 trình bày thời gian của bước lựa chọn coreset và hệ số tăng tốc của CRAIG-CH so với CRAIG; Hình 3.4 minh họa các số liệu này.

Bảng 3.6. Thời gian bước lựa chọn coreset trên CIFAR-10 (đơn vị: giây/epoch)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH	Tăng tốc
1%	0,0003	0,0254	0,0083	0,0208	1,22×
3%	0,0002	0,0741	0,0123	0,0399	1,86×
5%	0,0002	0,1219	0,0132	0,0554	2,20×
10%	0,0002	0,2422	0,0122	0,1216	1,99×



Hình 3.4. Thời gian bước lựa chọn coreset trên CIFAR-10. Chi phí của CRAIG tăng nhanh theo tỷ lệ nén, còn CRAIG-CH tăng chậm hơn.

Bảng 3.7 trình bày tổng thời gian thực thi của cả quá trình 200 epoch, đặt cạnh mốc Full Dataset là 652,3 giây.

Bảng 3.7. Tổng thời gian thực thi trên CIFAR-10, 200 epoch (đơn vị: giây)

Tỷ lệ nén	Random	CRAIG	CHVS4	CRAIG-CH
1%	36,4	65,5	60,1	59,7
3%	48,8	94,8	76,6	76,3
5%	71,5	111,5	85,8	92,7
10%	101,4	178,1	114,7	139,1
Full Dataset	652,3			

Từ hai bảng thời gian có thể rút ra một số nhận xét:

- Ở bước lựa chọn, chi phí của CRAIG tăng nhanh theo tỷ lệ nén, từ 0,0254 lên 0,2422 giây/epoch, trong khi CRAIG-CH tăng chậm hơn. Nhờ đó CRAIG-CH nhanh hơn CRAIG từ 1,22 đến 2,20 lần; mức tăng tốc nhìn chung cao hơn ở các tỷ lệ nén lớn, đạt 2,20 lần tại 5%.

- Trên tổng thời gian, CRAIG-CH nhanh hơn CRAIG từ 1,10 lần (tại 1%) đến 1,28 lần (tại 10%). Tại tỷ lệ nén 10%, CRAIG-CH hoàn thành trong 139,1 giây so với 178,1 giây của CRAIG.
- So với Full Dataset (652,3 giây), mọi phương pháp coreset đều nhanh hơn nhiều: CRAIG-CH nhanh hơn từ 4,69 lần (tại 10%) đến 10,93 lần (tại 1%). Ở bài toán phi lồi quy mô lớn này, lợi ích của hướng tiếp cận coreset thể hiện rõ hơn so với FashionMNIST.

d, So sánh sai số xấp xỉ gradient

Để lý giải chênh lệch độ chính xác ở trên, Bảng 3.8 trình bày sai số xấp xỉ gradient đã chuẩn hóa của ba phương pháp có dùng gradient.

Bảng 3.8. Sai số xấp xỉ gradient đã chuẩn hóa trên CIFAR-10

Tỷ lệ nén	CRAIG	CHVS4	CRAIG-CH
1%	0,0178	0,0290	0,0264
3%	0,0120	0,0204	0,0168
5%	0,0088	0,0160	0,0148
10%	0,0052	0,0178	0,0092

Từ bảng trên có thể rút ra một số nhận xét:

- CRAIG cho sai số xấp xỉ gradient nhỏ nhất, CHVS4 lớn nhất, còn CRAIG-CH nằm giữa hai phương pháp ở mọi tỷ lệ nén. Đây đúng là điều lý thuyết dự đoán: bước tham lam cải thiện được chất lượng của tập ứng viên bao lồi, nhưng vì chỉ tìm trong $3r$ ứng viên nên chưa bằng CRAIG tìm trên toàn bộ lớp.
- Thứ tự sai số xấp xỉ gradient này khớp với thứ tự độ chính xác ở Bảng 3.5, giải thích được vì sao CRAIG-CH đứng dưới CRAIG nhưng trên CHVS4 về độ chính xác.

Kết luận cho ResNet-20 với CIFAR-10

1. Độ chính xác phụ thuộc mạnh vào tỷ lệ nén (từ khoảng 44% đến 71% với CRAIG-CH). CRAIG-CH đứng dưới CRAIG nhưng trên CHVS4, đúng với vị trí của nó về sai số xấp xỉ gradient.
2. CRAIG-CH nhanh hơn CRAIG ở cả bước lựa chọn lẫn tổng thời gian, và nhanh hơn nhiều so với Full Dataset; các hệ số cụ thể đã nêu ở Bảng 3.6 và Bảng 3.7.
3. Kết quả phản ánh một sự đánh đổi: CRAIG-CH giảm chi phí bước lựa chọn coreset, đổi lại chấp nhận độ chính xác thấp hơn CRAIG một mức nhất định.

Kết luận chương

Chương này đã trình bày kết quả thực nghiệm trên hai bộ dữ liệu (Fashion-MNIST và CIFAR-10) với bốn phương pháp lựa chọn coreset và đường cơ sở Full Dataset. Có thể tóm lại như sau:

1. CRAIG-CH giảm được chi phí của bước lựa chọn coreset so với CRAIG, nhanh hơn từ 1,22 đến 2,73 lần tùy bộ dữ liệu và tỷ lệ nén. Đây là đóng góp cốt lõi của phương pháp và đúng như thiết kế lý thuyết ở Chương 2: thu hẹp không gian tìm kiếm của bước tham lam từ $O(n \cdot r)$ xuống $O(k_H \cdot r)$.
2. Mức tăng tốc ở bước lựa chọn chuyển thành lợi ích khoảng 1,0 đến 1,3 lần trên tổng thời gian thực thi, do bước lựa chọn chỉ chiếm một phần chi phí, phần còn lại là tính đặc trưng gradient và huấn luyện, vốn giống nhau giữa CRAIG và CRAIG-CH. So với Full Dataset, mọi phương pháp coreset đều rút ngắn tổng thời gian từ khoảng 4 đến 11 lần.
3. Việc thu hẹp không gian tìm kiếm đi kèm một sự đánh đổi về độ chính xác: CRAIG-CH đứng dưới CRAIG ở mọi cấu hình đã thử, nhưng đứng trên CHVS4. Trên CIFAR-10, sai số xấp xỉ gradient của CRAIG-CH nằm giữa hai phương pháp, phù hợp với thứ tự độ chính xác đó.

Từ các kết quả trên, có thể định vị CRAIG-CH như sau: đây là phương pháp hướng tới giảm chi phí của bước lựa chọn coreset, trong khi giữ chất lượng coreset ở mức giữa CHVS4 và CRAIG. Phương pháp phù hợp với những tình huống mà chi phí lựa chọn chiếm tỷ trọng lớn, chẳng hạn khi số điểm dữ liệu n rất lớn hoặc coreset phải chọn lại nhiều lần, và người dùng chấp nhận đánh đổi một phần độ chính xác để lấy tốc độ.

Hạn chế của kết quả cũng cần được nêu rõ. Thứ nhất, độ chính xác của CRAIG-CH luôn thấp hơn CRAIG, nên phương pháp không phù hợp khi độ chính xác là ưu tiên cao nhất. Thứ hai, ở quy mô dữ liệu đã thử, lợi ích trên tổng thời gian còn khiêm tốn vì bước lựa chọn chưa phải thành phần chi phối; lợi ích này được kỳ vọng sẽ rõ hơn khi n tăng lên nhiều lần, và việc kiểm chứng ở quy mô lớn hơn là một hướng phát triển tiếp theo.

KẾT LUẬN

Đề án bắt đầu từ một thách thức kép trong huấn luyện mô hình học máy hiện đại: nhu cầu về tập dữ liệu lớn và chi phí tính toán ngày càng tăng. Phương pháp lựa chọn coreset CRAIG có cơ sở lý thuyết vững nhưng chi phí của bước lựa chọn lớn, do phải chạy thuật toán tham lam trên toàn bộ dữ liệu. Để giảm chi phí này, đề án đã đề xuất, hiện thực hóa và đánh giá một phương pháp lai mang tên CRAIG-CH. Phần kết luận tổng kết các kết quả chính và nêu các hướng phát triển tiếp theo.

Các kết quả đạt được

Qua nghiên cứu lý thuyết và kiểm chứng thực nghiệm, đề án đã đạt được các kết quả sau:

- Đề xuất và hiện thực hóa phương pháp CRAIG-CH. Đề án đã đề xuất một phương pháp kết hợp, dùng thuật toán xấp xỉ bao lồi CHVS4 làm bước lọc tiền xử lý để chọn ra tập ứng viên, rồi chạy bước tham lam của CRAIG trên tập ứng viên đó. Phương pháp được hiện thực hóa thành một kịch bản Python cho phép so sánh bốn phương pháp lựa chọn coreset (Random, CRAIG, CHVS4, CRAIG-CH) cùng đường cơ sở Full Dataset, trên hai mô hình (hồi quy logistic, ResNet-20) và hai bộ dữ liệu (FashionMNIST, CIFAR-10).
- Cung cấp bằng chứng thực nghiệm về chi phí và chất lượng của CRAIG-CH. Các thí nghiệm ở Chương 3 kiểm chứng các giả thuyết từ Chương 2:
  1. Về chi phí bước lựa chọn: CRAIG-CH nhanh hơn CRAIG từ 1,22 đến 2,73 lần tùy bộ dữ liệu và tỷ lệ nén, nhờ thu hẹp không gian tìm kiếm của bước tham lam từ $O(n \cdot r)$ xuống $O(k_H \cdot r)$. Đây là đóng góp cốt lõi của phương pháp.
  2. Về tổng thời gian thực thi: mức tăng tốc ở bước lựa chọn chuyển thành lợi ích khoảng 1,0 đến 1,3 lần trên tổng thời gian so với CRAIG, do bước lựa chọn chỉ chiếm một phần chi phí, phần còn lại là tính đặc trưng gradient và huấn luyện, vốn giống nhau giữa hai phương pháp. So với Full Dataset, mọi phương pháp coreset rút ngắn tổng thời gian từ khoảng 4 đến 11 lần.
  3. Về chất lượng coreset: độ chính xác của CRAIG-CH thấp hơn CRAIG nhưng cao hơn CHVS4 trên cả hai bộ dữ liệu. Trên CIFAR-10, sai số xấp xỉ gradient đã chuẩn hóa của CRAIG-CH cũng nằm giữa CRAIG và CHVS4, phù hợp với thứ tự độ chính xác đó. Kết quả này cho thấy việc thu hẹp không gian tìm kiếm đi kèm một sự đánh đổi: giảm chi phí lựa chọn, đổi lại chấp nhận độ

chính xác thấp hơn CRAIG một mức nhất định.

- Phân tích các điều kiện và giới hạn áp dụng. Lợi ích của CRAIG-CH rõ nhất khi chi phí bước lựa chọn chiếm tỷ trọng lớn trong tổng chi phí, chẳng hạn khi số điểm dữ liệu n rất lớn hoặc coreset phải chọn lại nhiều lần. Ngược lại, ở quy mô dữ liệu đã thử, bước lựa chọn chưa phải thành phần chi phối nên lợi ích trên tổng thời gian còn khiêm tốn. Ngoài ra, khi số chiều của không gian gradient tăng cao, chi phí của bước lọc bao lồi cũng tăng, làm giảm lợi ích về tốc độ.

Đề xuất, khuyến nghị

Về việc chọn phương pháp cho từng tình huống: khi độ chính xác là ưu tiên cao nhất, CRAIG gốc là lựa chọn tốt hơn; khi cần tốc độ lựa chọn tối đa và chấp nhận độ chính xác thấp hơn, CHVS4 hoặc lấy mẫu ngẫu nhiên là phương án đơn giản. CRAIG-CH nằm giữa hai thái cực này, phù hợp khi muốn giảm chi phí lựa chọn mà vẫn giữ chất lượng coreset cao hơn CHVS4.

Hướng nghiên cứu tiếp theo

Phương pháp CRAIG-CH còn mở ra một số hướng phát triển:

- Kiểm chứng ở quy mô lớn hơn. Cần kiểm chứng trên các tập dữ liệu lớn hơn nhiều lần, nơi chi phí bước lựa chọn tăng theo $O(n \cdot r)$ và lợi thế của việc thu hẹp không gian tìm kiếm được kỳ vọng thể hiện rõ hơn.
- Thu hẹp khoảng cách độ chính xác so với CRAIG. Có thể khảo sát việc tăng ngân sách ứng viên k_H, hoặc kết hợp tiêu chí hình học với các tiêu chí lựa chọn khác như lấy mẫu theo độ nhạy, để tập ứng viên đa dạng hơn trước khi đưa vào bước tham lam, nhằm giảm phần độ chính xác bị đánh đổi.
- Phân bổ ngân sách theo lớp một cách thích ứng. Đề án chọn coreset trên từng lớp với ngân sách như nhau. Hướng tiếp theo có thể phân bổ ngân sách r và k_H thích ứng cho mỗi lớp, dựa trên độ phức tạp, số lượng mẫu hoặc độ trải của đám mây gradient của lớp đó.
- Xây dựng coreset động. Phương pháp hiện tại chọn lại coreset theo tần suất cố định. Một hướng đi là chỉ chọn lại khi phát hiện thay đổi đáng kể trong không gian gradient, hoặc khi sai số xấp xỉ gradient đo trên một tập kiểm định nhỏ vượt một ngưỡng cho trước.
- Tối ưu hóa bước lọc bao lồi cho không gian nhiều chiều. Hiệu năng của CRAIG-CH phụ thuộc vào bước lọc bao lồi. Nghiên cứu các thuật toán xấp xỉ bao lồi hiệu quả hơn khi số chiều rất cao, chẳng hạn dùng phép chiếu ngẫu nhiên để giảm số

chiều trước khi lọc, là hướng cần thiết để mở rộng phương pháp.

Tài liệu tham khảo

[1] Bertsekas, D. P. (2015), *Convex optimization algorithms*. Belmont, MA: Athena Scientific.

[2] Bertsekas, D. P. (2015), "Incremental gradient, subgradient, and proximal methods for convex optimization: A survey." arXiv preprint arXiv:1507.01030.

[3] Chai, C., Wang, J., Tang, N., Yuan, Y., Liu, J., Deng, Y., & Wang, G. (2023), "Efficient coreset selection with cluster based methods." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (KDD 2023).

[4] Chen, J. (2025), "Efficient core set selection for deep learning through squared loss minimization." In *International Conference on Machine Learning* (ICML).

[5] Defazio, A., Bach, F., & Lacoste-Julien, S. (2014), "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives." In *Advances in Neural Information Processing Systems*, pp. 1646–1654.

[6] Ding, S., Nie, X., Qiao, H., & Zhang, B. (2017), "A fast algorithm of convex hull vertices selection for online classification." *IEEE Transactions on Neural Networks and Learning Systems* 29(4), pp. 792–806.

[7] Garg, P., & Roy, K. (2023), "Samples with low loss curvature improve data efficiency." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR 2023), pp. 20290–20300.

[8] Gurbuzbalaban, M., Ozdaglar, A., & Parrilo, P. A. (2017), "On the convergence rate of incremental aggregated gradient algorithms." *SIAM Journal on Optimization* 27(2), pp. 1035–1048.

[9] Har-Peled, S., & Mazumdar, S. (2004), "On coresets for k-means and k-median clustering." In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pp. 291–300.

[10] Jayaram, M. A., & Fleyeh, H. (2016), "Convex hulls in image processing: A scoping review." *American Journal of Intelligent Systems* 6(2), pp. 48–58.

[11] Johnson, R., & Zhang, T. (2013), "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in Neural Information Processing Systems*, pp. 315–323.

[12] Linh, N. K., Hieu, V. T., & Thu, V. H. (2025), "Improving machine learning training efficiency via convex hull-guided CRAIG coreset selection." Manuscript.

[13] Lucic, M., Faulkner, M., Krause, A., & Feldman, D. (2017), "Training Gaussian mixture models at scale via coresets." *Journal of Machine Learning Research* 18(1), pp. 5885–5909.

[14] Minoux, M. (1978), "Accelerated greedy algorithms for maximizing submodular set functions." In *Optimization Techniques*, pp. 234–243. Springer.

[15] Mirzasoleiman, B., Bilmes, J., & Leskovec, J. (2020), "Coresets for data-efficient training of machine learning models." In *Proceedings of the 37th International Conference on Machine Learning*, PMLR 119, pp. 6938–6948.

[16] Nedić, A., & Bertsekas, D. (2001), "Convergence rate of incremental subgradient algorithms." In *Stochastic Optimization: Algorithms and Applications*, pp. 223–264. Springer.

[17] Nemhauser, G., Wolsey, L., & Fisher, M. (1978), "An analysis of approximations for maximizing submodular set functions-i." *Mathematical Programming* 14(1), pp. 265–294.

[18] Roux, N. L., Schmidt, M., & Bach, F. R. (2012), "A stochastic gradient method with an exponential convergence rate for finite training sets." In *Advances in Neural Information Processing Systems*, pp. 2663–2674.

[19] Strubell, E., Ganesh, A., & McCallum, A. (2019), "Energy and policy considerations for deep learning in NLP." arXiv preprint arXiv:1906.02243.

[20] Wei, K., Iyer, R., & Bilmes, J. (2015), "Submodularity in data subset selection and active learning." In *Proceedings of the International Conference on Machine Learning*, pp. 1954–1963.

[21] Xia, X., Liu, J., Yu, J., Shen, X., Han, B., & Liu, T. (2023), "Moderate coreset: A universal method of data selection for real world data efficient deep learning." In *International Conference on Learning Representations* (ICLR).

[22] Xiao, H., Rasul, K., & Vollgraf, R. (2017), "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms." *arXiv preprint* arXiv:1708.07747.

[23] Xu, X., Zhang, J., Liu, F., Sugiyama, M., & Kankanhalli, M. (2023), "Efficient adversarial contrastive learning via robustness aware coreset selection." In *Advances in Neural Information Processing Systems* (NeurIPS 2023).

[24] Yang, Y., Kang, H., & Mirzasoleiman, B. (2023), "Towards sustainable learning: Coresets for data-efficient deep learning." In *Proceedings of the 40th International Conference on Machine Learning*, vol. 202 of *Proceedings of Machine Learning Research*, pp. 39314–39330.

[25] Zhang, J., Qin, Y., Pi, R., Zhang, W., Pan, R., & Zhang, T. (2025), "TAGCOS: Task agnostic gradient clustered coreset selection for instruction tuning data." In *Findings of the Association for Computational Linguistics: NAACL 2025*, Albuquerque, NM, USA: Association for Computational Linguistics, pp. 4671–4686. doi: 10.18653/v1/2025.findings.naacl.264.

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung đề án qua phần mềm Turnitin một cách trung thực và đạt kết quả mức độ tương đồng 17% toàn bộ nội dung đề án. Bản đề án kiểm tra qua phần mềm là bản cứng đề án đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Hà Nội, ngày ... tháng ... năm 2026

HỌC VIÊN CAO HỌC/NCS

(Ký và ghi rõ họ tên)



Chu Quang Hưng

17% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn

Nguồn hàng đầu

14% Nguồn Internet
9% Ấn bản
7% Bài tập được nộp (bài của học sinh)

18% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn

Nguồn hàng đầu

16% Nguồn Internet
11% Ấn bản
7% Bài tập được nộp (bài của học sinh)

GVHD

Nguyễn Hoài Long

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ
BƯU CHÍNH VIỄN THÔNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA
VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BÁO CÁO GIẢI TRÌNH
SỬA CHỮA, HOÀN THIỆN ĐỀ ÁN TỐT NGHIỆP

Họ và tên học viên: Chu Quang Hưng
Chuyên ngành: Khoa học máy tính Khóa: .....
Tên đề tài: Cải tiến CRAIG bằng Convex Hull để xây dựng Coreset nâng cao hiệu suất huấn luyện mô hình học máy.
Người hướng dẫn khoa học: TS. Nguyễn Kiều Linh
Ngày bảo vệ: .....

Các nội dung học viên đã sửa chữa, bổ sung trong đề án tốt nghiệp theo ý kiến đóng góp của Hội đồng chấm đề án tốt nghiệp:

TT	Ý kiến hội đồng	Sửa chữa của học viên
1	Rà soát, chỉnh sửa lỗi soạn thảo, lỗi trình bày và lỗi chính tả.	Học viên đã rà soát toàn bộ đề án, thống nhất thuật ngữ và cách trình bày, chỉnh sửa các lỗi soạn thảo và trình bày trong toàn quyển.
2	Con số "nhanh hơn 3,3 lần trên CIFAR-10" ở phần kết luận chưa chính xác, cần làm rõ.	Học viên đã chạy lại thực nghiệm và sửa lại số liệu. Hệ số tăng tốc của bước lựa chọn coreset của CRAIG-CH so với CRAIG là 1,22 đến 2,73 lần, tùy bộ dữ liệu và tỷ lệ nén (chi tiết ở bảng thời gian, Chương 3).
3	Đề án thiếu số liệu tổng thời gian thực thi để so sánh với huấn luyện trên toàn bộ dữ liệu.	Học viên đã bổ sung bảng tổng thời gian thực thi, gồm cả bước huấn luyện, vào Chương 3. So với huấn luyện trên toàn bộ dữ liệu, CRAIG-CH rút ngắn tổng thời gian từ 4,2 đến 10,9 lần; so với CRAIG, nhanh hơn từ 1,0 đến 1,3 lần.

TT	Ý kiến hội đồng	Sửa chữa của học viên
4	Tham số ngân sách ứng viên bao lồi k_H chưa được nêu trong phần thực nghiệm.	Học viên đã bổ sung giá trị $k_H = 3r$ vào phần thiết lập thực nghiệm ở Chương 3, kèm giải thích quan hệ giữa k_H và kích thước coreset r.
5	Bổ sung phân tích độ phức tạp của thuật toán và giải thích kết quả theo đặc tính dữ liệu.	Học viên đã bổ sung một mục riêng phân tích độ phức tạp và tài nguyên ở Chương 2, kèm bảng so sánh giữa các phương pháp CHVS4, CRAIG và CRAIG-CH. Chi phí của bước lựa chọn coreset giảm từ mức bậc hai theo số điểm mỗi lớp (ở CRAIG) xuống mức tuyến tính $O(n_c k_H)$ với $k_H \ll n_c$ (ở CRAIG-CH), và nhu cầu bộ nhớ ma trận khoảng cách giảm từ $O(n_c^2)$ xuống $O(n_c k_H)$; đồng thời học viên giải thích kết quả theo đặc tính bài toán lồi (Fashion-MNIST) và phi lồi (CIFAR-10) ở Chương 3.
6	Nâng cao độ tin cậy của kết quả thực nghiệm.	Học viên đã chạy lại toàn bộ thực nghiệm với ba hạt giống ngẫu nhiên và báo cáo kết quả trung bình kèm độ lệch chuẩn.

TT	Ý kiến hội đồng	Sửa chữa của học viên
7	Phần cơ sở lý luận (Chương 1) và phương pháp đề xuất (Chương 2) cần được trình bày đầy đủ và chi tiết hơn.	Học viên đã bổ sung, làm sâu sắc cả hai chương. Ở Chương 1: khái niệm coreset mạnh và coreset yếu; phân tích sâu hơn về hàm dưới mô-đun; một mục riêng về phương pháp gradient tăng dần (là đối tượng mà CRAIG hướng tới tăng tốc); các biến thể của thuật toán hạ gradient ngẫu nhiên; và phần điểm cực biên, siêu phẳng tựa cùng thuật toán tính bao lồi. Ở Chương 2: phần chứng minh cận trên của sai số xấp xỉ gradient; dẫn giải vì sao dùng gradient theo logits của lớp cuối; giả mã và phân tích chi phí của thuật toán CHVS4; một ví dụ minh họa bằng số; mục lựa chọn tham số k_H và các bảo đảm lý thuyết được kế thừa; cùng mục hạn chế, phạm vi áp dụng của CRAIG-CH. Ngoài ra, học viên bổ sung các hình minh họa cơ chế CHVS4, cách gán trọng số coreset và một bảng so sánh độ phức tạp, giúp hai chương được trình bày chi tiết và trực quan hơn đáng kể.

Hà Nội, ngày tháng năm 2026

CHỦ TỊCH HỘI ĐỒNG CHẤM ĐỀ ÁN	THƯ KÝ HỘI ĐỒNG	NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN
Nguyễn Duy Phương	Đào Thị Thúy Quỳnh	Nguyễn Kiều Linh	Chu Quang Hưng

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN
HỌP HỘI ĐỒNG CHẤM ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Căn cứ quyết định số Quyết định số 1949/QĐ-HV ngày 07 tháng 07 năm 2026 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ. Hội đồng đã họp vào hồi 9 giờ 00 phút, ngày 21 tháng 07 năm 2026 tại Học viện Công nghệ Bưu chính Viễn thông để chấm đề án tốt nghiệp thạc sĩ cho:

Học viên: **Chu Quang Hưng**

Tên đề án tốt nghiệp: **Cải tiến CRAIG bằng Convex Hull để Xây dựng Coreset nâng cao hiệu suất huấn luyện Mô hình Học máy**

Chuyên ngành: **Khoa học máy tính** Mã số: **8480101**

Các thành viên của Hội đồng chấm đề án tốt nghiệp có mặt: 5 / 05

TT	HỌ VÀ TÊN	TRÁCH NHIỆM TRONG HĐ	GHI CHÚ
1	**TS. Nguyễn Duy Phương**	Chủ tịch	
2	**TS. Đào Thị Thúy Quỳnh**	Thư ký	
3	**TS. Đào Ngọc Phong**	Phản biện 1	
4	**PGS.TS. Đặng Văn Đức**	Phản biện 2	
5	**TS. Trần Minh Tân**	Uỷ viên	

Các nội dung thực hiện:

1. Chủ tịch Hội đồng điều khiển buổi họp. Công bố quyết định của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ.
2. Người hướng dẫn khoa học hoặc thư ký đọc lý lịch khoa học và các điều kiện bảo vệ đề án tốt nghiệp của học viên. (có bản lý lịch khoa học và kết quả các môn học cao học của học viên kèm theo).
3. Học viên trình bày tóm tắt đề án tốt nghiệp.
4. Phản biện 1 đọc nhận xét (có văn bản kèm theo)
5. Phản biện 2 đọc nhận xét (có văn bản kèm theo)
6. Các câu hỏi của thành viên Hội đồng:

1) Mục tiêu của đề án là giảm chi phí tính toán để nâng cao hiệu suất huấn luyện. Tuy nhiên, có thấy thời gian chọn coreset của CRAIG và CRAIG - Cải cũng lớn. Trong đề án không có số liệu về thời gian huấn luyện full data set để so sánh?
2) Trang 29, Học viên tính bằng "giảm đáng kể thời gian xây dựng coreset so với CRAIG gốc", đề nghị học viên làm rõ nội dung này?

7. Trả lời của học viên:

1, Ở đây học viên đưa thành so sánh là thời gian của bước chạy coreset chứ không phải của toàn thời gian huấn luyện

2, Học viên tiếp thu và chỉnh sửa, bổ sung vào đề án

8. Thư ký đọc nhận xét về quá trình thực hiện đề án tốt nghiệp của học viên *(có văn bản kèm theo).*

9. Hội đồng họp riêng:

- Bầu Ban kiểm phiếu:
  1. Trưởng Ban kiểm phiếu: TS. Nguyễn Duy Phương
  2. Uỷ viên Ban kiểm phiếu: TS. Đào Ngọc Phong
  3. Uỷ viên Ban kiểm phiếu: TS. Đào Thị Thúy Quỳnh.
- Hội đồng chấm đề án tốt nghiệp bằng bỏ phiếu kín.
- Ban kiểm phiếu làm việc:
- Trưởng Ban kiểm phiếu báo cáo kết quả kiểm phiếu *(có Biên bản họp Ban kiểm phiếu kèm theo)*
- **Điểm trung bình của đề án tốt nghiệp:** 8,0

Kết luận:

1. Các nội dung cần chỉnh sửa, hoàn thiện sau bảo vệ đề án tốt nghiệp:

- Làm rõ ý nghĩa của bài toán tính toán tập dữ liệu coreset.
- Chi tiết hoá từng nội dung trong đề án
- làm rõ kết quả thực nghiệm trên tập dữ liệu và làm rõ hơn lợi ích phương pháp đề xuất
- Bổ sung nội dung các chương của đề án

2. Đề nghị Học viện công nhận (hoặc không) và cấp bằng (hoặc không) thạc sĩ cho học viên: đồng đề nghị công nhận

3. Đề án tốt nghiệp có thể phát triển thành đề tài nghiên cứu cho NCS..... không.....

Buổi làm việc kết thúc vào..... cùng ngày.

Chủ tịch	Thư ký
Nguyễn Duy Phương	Đào Thị Thúy Quỳnh.

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

(Dùng cho người phản biện)

Tên đề tài đề án tốt nghiệp: Cải tiến CRAIG bằng Convex Hull để xây dựng Coreset nâng cao hiệu suất huấn luyện mô hình học máy.

Chuyên ngành : Khoa học máy tính

Mã chuyên ngành: 8.48.01.01

Họ và tên học viên: Chu Quang Hưng

Họ và tên người nhận xét: Đào Ngọc Phong

Học hàm, học vị: Tiến sĩ

Chuyên ngành:Hệ thống thông tin

Cơ quan công tác: Khoa Công nghệ thông tin 1

Số điện thoại: 0912239696 E-mail: phongdn@ptit.edu.vn

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài:

Trong kỷ nguyên dữ liệu lớn và sự bùng nổ của các kiến trúc học sâu (Deep Learning), chi phí tính toán, thời gian huấn luyện và lượng năng lượng tiêu thụ để tối ưu hóa các tham số mô hình đã trở thành những rào cản kỹ thuật lớn. Hướng tiếp cận xây dựng tập dữ liệu rút gọn (coreset) – tập con mang tính đại diện cao cho toàn bộ tập dữ liệu gốc – là một giải pháp đột phá theo triết lý "data-centric AI".

Phương pháp CRAIG là một cột mốc lý thuyết quan trọng dựa trên tối ưu hóa hàm dưới mô-đun để xấp xỉ gradient. Tuy nhiên, rào cản thực tế của CRAIG là chi phí lựa chọn tham lam động. Đề tài lựa chọn hướng nghiên cứu "Cải tiến CRAIG bằng Convex Hull" (thuật toán CRAIG-CH) nhằm lọc trước không gian tìm kiếm bằng bao lồi xấp xỉ hình học là vô cùng cấp thiết, có tính thời sự cao và trực diện giải quyết điểm nghẽn vật lý của các phương pháp coreset hiện nay.

Đề án dựa trên các nền tảng lý thuyết toán học và công nghệ thông tin bao gồm: lý thuyết tối ưu hóa tổ hợp hàm dưới mô-đun (submodular optimization), lý thuyết coreset xấp xỉ sai số gradient toàn cục (thuật toán CRAIG), và lý thuyết hình học tính toán về bao lồi (Convex Hull).

Đề tại "Cải tiến CRAIG bằng Convex Hull để xây dựng CoreSet nâng cao hiệu suất huấn luyện mô hình học máy" có ý nghĩa khoa học và thực tiễn.

II/ Nội dung của đề án tốt nghiệp, các kết quả đã đạt được:

Đề án đã hệ thống hóa các kiến thức nền tảng về Coreset (lấy mẫu độ nhạy, tối ưu hàm dưới mô-đun Facility Location), các thuật toán hạ gradient giảm phương sai (SGD, SVRG, SAGA) và lý thuyết hình học bao lồi đa diện trong không gian nhiều chiều.

Phân tích chi tiết và định lượng được các hạn chế của thuật toán CRAIG gốc, đặc biệt là chi phí tính toán của bước lựa chọn tham lam động và nguy cơ chọn trùng lặp điểm trong các cụm dữ liệu đặc.

Đề xuất CRAIG-CH và cài đặt thuật toán, thực hiện đánh giá và so sánh kết quả. Thực hiện phân tích, đánh giá định lượng sâu sắc về hành vi và mối tương quan giữa hàm mất mát (loss), độ chính xác (accuracy) và thời gian thực thi của các phương pháp coreset trên hai không gian tối ưu hóa lồi và phi lồi khác nhau.

III/ Những vấn đề cần giải thích thêm:

Một số góp ý:

- Nội dung đề án là 30 trang, do đó, học viên nên xem xét bổ sung thêm đặc biệt là chương 2 và chương 3 để làm rõ thêm thuật toán và kết quả thực nghiệm.
- Đề án còn một số lỗi trình bày, lỗi chính tả.

Một số câu hỏi:

- Mục tiêu của đề án là giảm chi phí tính toán để nâng cao hiệu suất huấn luyện mô hình. Tuy nhiên, kết quả tại thực nghiệm cho thấy thời gian chọn CoreSet của CRAIG nói chung và CRAIG-CH cũng rất lớn, lên tới 340.02 giây/epoch ở tỷ lệ nén 5% và 1251.27 giây/epoch ở tỷ lệ nén 10%. Trong khi đề án không có số liệu về thời gian huấn luyện của full Dataset để làm tham chiếu.
- Các kết quả thực nghiệm cho thấy với tập dữ liệu FashionMNIST thì việc Ramdom cho kết quả tốt về độ chính xác, thời gian, giá trị mất mát. Vậy việc thực nghiệm của CRAIG, CRAIG-CH còn nhiều ý nghĩa (khi so sánh với nhau) trong khi kết quả chung thì đều kém hơn Ramdom rất nhiều.
- Theo phần mô tả thuật toán, CRAIG-CH phụ thuộc chặt chẽ vào tham số ngân sách ứng viên bao lồi k_H, tham số này quyết định trực tiếp đến tốc độ lọc của CHVS4 và quy mô không gian tối ưu của CRAIG. Tuy nhiên, trong phần thực nghiệm không thấy học viên đề cập đến tham số này.
- Tại kết luận trang 29, "giảm đáng kể thời gian xây dựng coreset (selection time) so với CRAIG gốc (ví dụ: nhanh hơn 3.3 lần trên CIFAR-10)", đề nghị học viên làm rõ nội dung này đối chiếu với các bảng dữ liệu trong chương 3.

V/ Kết luận:

Đồng ý cho phép học viên bảo vệ đề án tốt nghiệp.

Ngày 20 tháng 07 năm 2026

NGƯỜI NHẬN XÉT

Đào Ngọc Phong

TẬP ĐOÀN BƯU CHÍNH VIỄN THÔNG VN
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH-VIỄN THÔNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SỸ

(Dùng cho cán bộ phản biện)

Tên đề tài đề án: ***Cải tiến CRAIG bằng Convex Hull để xây dựng coreset nâng cao hiệu suất huấn luyện mô hình học máy***

Chuyên ngành: Khoa học máy tính

Mã số: 8.48.01.01

Tên học viên: **Chu Quang Hưng**

Cán bộ hướng dẫn: TS. Nguyễn Kiều Linh

Họ và tên người nhận xét: PGS.TS Đặng Văn Đức

Chuyên ngành: Công nghệ thông tin

Cơ quan công tác: Viện Công nghệ thông tin, Viện Hàn lâm KH&CN Việt Nam

NỘI DUNG NHẬN XÉT

I. Cơ sở khoa học và thực tiễn, tính cấp thiết của đề án

Đề án của học viên *Chu Quang Hưng* nghiên cứu bài toán xây dựng *coreset* nhằm giảm chi phí huấn luyện mô hình học máy trên các tập dữ liệu lớn. Đây là hướng nghiên cứu có ý nghĩa trong bối cảnh các mô hình học sâu ngày càng đòi hỏi nhiều tài nguyên tính toán.

Đề án được xây dựng trên cơ sở lý thuyết tương đối đầy đủ, kết hợp giữa phương pháp lựa chọn *coreset CRAIG*, lý thuyết tối ưu hóa *gradient* và hình học tính toán thông qua khái niệm bao lồi (*Convex Hull*). Việc sử dụng thuật toán xấp xỉ bao lồi để thu hẹp không gian tìm kiếm trước khi áp dụng CRAIG là một hướng tiếp cận hợp lý, có cơ sở khoa học và phù hợp với mục tiêu nghiên cứu.

Về mặt thực tiễn, phương pháp đề xuất hướng tới giảm thời gian xây dựng *coreset* trong khi vẫn duy trì chất lượng mô hình, góp phần nâng cao hiệu quả huấn luyện đối với các bài toán học máy có quy mô dữ liệu lớn.

II. Về nội dung, chất lượng của đề án, các kết quả đã đạt được

Đề án được trình bày với bố cục hợp lý gồm ba chương: cơ sở lý thuyết, phương pháp đề xuất và thực nghiệm đánh giá. Nội dung giữa các chương có sự liên kết logic, các bảng biểu, hình vẽ và tài liệu tham khảo được trình bày tương đối đầy đủ.

Về nội dung chuyên môn, học viên đã phân tích được hạn chế của phương pháp CRAIG trong chi phí lựa chọn *coreset*, từ đó đề xuất phương pháp CRAIG-CH bằng cách sử dụng thuật toán CHVS4 để tạo tập ứng viên trước khi thực hiện lựa chọn tham lam. Ý tưởng đề xuất có tính kế thừa, đồng thời thể hiện sự chủ động trong việc kết hợp giữa tối ưu hóa và hình học tính toán.

Phần thực nghiệm được xây dựng tương đối công phu. Phương pháp được đánh giá trên

cả bài toán lồi (*Logistic Regression* với *FashionMNIST*) và bài toán phi lồi (*ResNet-20* với *CIFAR-10*), đồng thời so sánh với các phương pháp *Random*, CRAIG, CHVS4 và *Full Dataset*. Các kết quả thực nghiệm cho thấy CRAIG-CH giúp giảm đáng kể thời gian lựa chọn *coreset* so với CRAIG trong khi vẫn duy trì chất lượng mô hình ở mức chấp nhận được. Đây là đóng góp chính của đề án.

Bên cạnh những kết quả đạt được, đề án vẫn còn một số điểm cần hoàn thiện:

- Cần phân tích đầy đủ độ phức tạp tính toán của toàn bộ thuật toán CRAIG-CH.
- Cần phân tích sâu hơn dưới góc độ đặc tính dữ liệu và thuật toán khi giải thích kết quả thực nghiệm, ví dụ khi giải thích phương pháp *Random* đạt kết quả tốt hơn trên *FashionMNIST*.
- Nên bổ sung các đường cong hội tụ (*learning curves*) để đánh giá rõ hơn tốc độ hội tụ và tính ổn định của phương pháp, thay vì chỉ dựa trên kết quả tại epoch cuối cùng.
- Bản đề án còn một số lỗi trình bày, lỗi chính tả cần được rà soát, chỉnh sửa trước khi hoàn thiện bản chính thức.

Câu hỏi dành cho học viên:

a. Tiêu chí để lựa chọn tham số k_H trong thuật toán CHVS4? Mối quan hệ giữa k_H và kích thước *coreset r* ảnh hưởng như thế nào đến hiệu quả của phương pháp CRAIG-CH?

b. Đối với mô hình học sâu, khoảng cách giữa các *gradient* được xấp xỉ như thế nào trong quá trình xây dựng *coreset*? Việc xấp xỉ này ảnh hưởng ra sao đến chất lượng của phương pháp đề xuất?

c. Trong thực nghiệm trên *FashionMNIST*, phương pháp *Random* cho kết quả tốt hơn CRAIG và CRAIG-CH. Theo tác giả, nguyên nhân của hiện tượng này là gì và trong trường hợp đó ưu thế của *coreset* được thể hiện ở những khía cạnh nào?

d. So sánh thời gian thực thi của CRAIG-CH (bao gồm cả thời gian lựa chọn coreset và thời gian huấn luyện mô hình) với CRAIG và *Full Dataset*?

III. Kết luận

Bản đề án tốt nghiệp thạc sỹ đã trình bày được các nội dung cơ bản để đạt được mục tiêu đề ra. Tuy vẫn còn một số điểm cần bổ sung và hoàn thiện như đã nêu trên, đề án nhìn chung đáp ứng yêu cầu của đề án thạc sĩ kỹ thuật định hướng ứng dụng. Tôi đồng ý để học viên *Chu Quang Hưng* được bảo vệ tại Hội đồng chấm đề án thạc sỹ của Học viện CNBCVT.

Hà Nội, ngày tháng 7 năm 2026

NGƯỜI NHẬN XÉT

PGS.TS Đặng Văn Đức