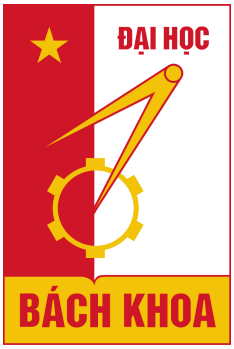
ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

o0o



BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: NHẬP MÔN PHƯƠNG PHÁP TỐI ƯU

Chủ đề: Nghiên cứu bài báo *Self-adaptive algorithms for quasiconvex programming and applications to machine learning*

Giảng viên hướng dẫn: **TS Trịnh Ngọc Hải**

Sinh viên thực hiện: **Nhóm 3**

1. Nguyễn Phong	MSSV: 202400066
2. Nguyễn Đăng Long	MSSV: 202400057
3. Phạm Gia Linh	MSSV: 202416262
4. Nguyễn Tuấn Long	MSSV: 202416269
5. Đoàn Đức Mạnh	MSSV: 202400058
6. Trần Đình Nam	MSSV: 202400063
7. Hoàng Thị Thu Phương	MSSV: 202400068
8. Đặng Bảo Quân	MSSV: 202416319
9. Phạm Thị Bích Phương	MSSV: 202400069
10. Nguyễn Hải Yến Nhi	MSSV: 202400064
11. Lương Đức Mạnh	MSSV: 20235152
12. Dương Thanh Minh	MSSV: 20230047

Hà Nội - 2025

Tóm tắt

Báo cáo này tóm lược và phân tích bài báo *Self-adaptive algorithms for quasiconvex programming and applications to machine learning* (Thang & Hai, 2024). Bài báo đề xuất một chiến lược kích thước bước tự điều chỉnh (self-adaptive step-size) cho một lớp rộng bài toán lập trình không lồi trên tập ràng buộc không bị chặn, mà không bao gồm các kỹ thuật tìm kiếm dòng (line-search). Phương pháp này thiết lập sự hội tụ của một cách tiếp cận chung dưới các giả định nhẹ, trong đó hàm mục tiêu có thể không thỏa mãn điều kiện lồi. Không giống như các thuật toán tìm kiếm dòng theo hướng giảm, thuật toán được đề xuất không cần hằng số Lipschitz đã biết để xác định kích thước bước đầu tiên. Đặc điểm quan trọng của quá trình này là sự giảm dần ổn định của kích thước bước cho đến khi một điều kiện nhất định được thỏa mãn. Cụ thể, nó có thể cung cấp một cách tiếp cận chiếu gradient mới cho các bài toán tối ưu hóa với tập ràng buộc không bị chặn. Hiệu quả của phương pháp được minh họa qua các thí nghiệm học máy quy mô lớn, gồm lựa chọn đặc trưng có giám sát, hồi quy logistic đa biến và phân loại bằng mạng nơ-ron. Báo cáo tập trung trình bày động lực, ý tưởng thuật toán và tổng hợp kết quả thực nghiệm, ứng dụng, kèm nhận xét về bài báo.

1 Giới thiệu

Các phương pháp giảm gradient (gradient descent methods) là một công cụ phổ biến cho một loạt các bài toán lập trình, từ lồi đến không lồi, và có nhiều ứng dụng thực tế (xem Boyd và Vandenberghe 2009 [3]; Cevher et al. 2014 [4]; Lan 2020 [10] và các tài liệu tham khảo trong đó). Tại mỗi lần lặp, các thuật toán giảm gradient cung cấp một chuỗi giải pháp lặp dựa trên hướng gradient và kích thước bước. Trong một thời gian dài, các nhà nghiên cứu đã tập trung vào việc tìm hướng để cải thiện tốc độ hội tụ của kỹ thuật, trong khi kích thước bước thường được xác định bằng một trong số ít các phương pháp nổi tiếng (xem Boyd và Vandenberghe 2009 [3]; Nesterov 2013 [14]).

Gần đây, các lĩnh vực ứng dụng học máy mới với chiều dữ liệu cao và hàm mục tiêu không lồi đã yêu cầu phát triển các quy trình chọn kích thước bước mới để giảm tổng chi phí tính toán của phương pháp (xem Cevher et al. 2014 [4]; Lan 2020 [10]). Kỹ thuật tìm kiếm dòng (chính xác hoặc xấp xỉ) gây ra chi phí tính toán đáng kể cho mỗi lần lặp, đặc biệt khi việc tính giá trị hàm gần giống với việc tính toán đạo hàm và đòi hỏi giải quyết các bài toán phụ trợ phức tạp (xem Boyd và Vandenberghe 2009 [3]). Để tránh tìm kiếm dòng, giá trị kích thước bước có thể được tính toán bằng cách sử dụng thông tin có sẵn trước, chẳng hạn như hằng số Lipschitz cho gradient. Tuy nhiên, điều này yêu cầu chỉ sử dụng một phần ước tính không chính xác, dẫn đến sự chậm lại của quá trình hội tụ. Điều này cũng đúng đối với quy tắc chuỗi phân kỳ nổi tiếng (xem Kiwiel 2001 [8]; Nesterov 2013 [14]).

Mặc dù Kiwiel đã phát triển phương pháp gradient cho lập trình quasiconvex vào năm 2001 (Kiwiel 2001 [8]), nhưng nó dẫn đến hội tụ chậm do sử dụng kích thước bước giảm dần. Sau đó, có một số cải tiến đối với phương pháp gradient, chẳng hạn như công trình của Yu et al. (2019) [19] và Hu et al. (2020) [7], sử dụng kích thước bước hằng số nhưng hàm mục tiêu phải thỏa mãn điều kiện Hölder. Phương pháp khác là cách tiếp cận neurodynamic, sử dụng mô hình mạng nơ-ron hồi quy (RNN) để giải các bài toán lập trình giả lồi với tập ràng buộc không bị chặn (xem Bian et al. 2018 [2]; Liu et al. 2022 [11]). Tuy nhiên, việc chọn kích thước bước của phương pháp này là cố định và không thích ứng.

Quy trình kích thước bước thích ứng đã được đề xuất trong Konnov (2018) [9] và Ferreira và Sosa (2022) [6]. Phương pháp được đưa ra trong Konnov (2018) [9], trong đó một thuật toán kích thước bước cho phương pháp gradient có điều kiện được đề xuất, có hiệu quả để giải các bài toán lập trình giả lồi với điều kiện tập hợp khả thi bị chặn. Phương pháp này sau đó đã được mở rộng trong Ferreira và Sosa (2022) [6] sang trường hợp riêng là tập hợp khả thi không bị chặn, nhưng lại không thể áp dụng cho trường hợp không ràng buộc.

Để giải quyết vấn đề này, bài báo *Self-adaptive algorithms for quasiconvex programming and applications to machine learning* (Thang & Hai, 2024) [16] đã đề xuất một thuật toán kích thước bước thích ứng mới và không tìm kiếm dòng cho một lớp rộng các bài toán lập trình trong đó hàm mục tiêu là không lồi và trơn, tập ràng buộc là không bị chặn, đóng và lồi. Một thành phần quan trọng của quy trình này là việc giảm dần kích thước bước cho đến khi một điều kiện được xác định trước được thỏa mãn. Mặc dù tính liên tục L-Lipschitz của gradient của hàm mục tiêu là cần thiết cho sự hội tụ, cách tiếp cận này không sử dụng các hằng số được xác định trước. Các thay đổi được đề xuất đã được chứng minh là hiệu quả trong các thử nghiệm tính toán sơ bộ.

Việc mở rộng kỹ thuật kích thước bước thích ứng sang trường hợp tập ràng buộc không bị chặn là một trong những đóng góp chính của bài báo. Nhiệm vụ này gặp một số khó khăn. Thứ nhất, sự hội tụ của thuật toán phải được đảm bảo mà không cần bất kỳ điều kiện phụ trợ bổ sung nào. Thứ hai, các toán tử chiếu đảm bảo rằng điểm x^{k+1} nằm trong tập hợp khả thi, nhưng cần chứng minh sự hội tụ của thuật toán khi có phép chiếu này. Cuối cùng, thuật toán thích ứng được đề xuất phải bao gồm trường hợp kích thước bước cố định. Trường hợp này cho thấy thuật toán được đề xuất là một sự mở rộng tự nhiên của thuật toán giảm gradient điển hình và có lợi thế cho các ứng dụng thực tế, đặc biệt là các hàm mục tiêu không lồi. Các ví dụ tính toán cho các bài toán quy mô lớn với hàm mục tiêu không lồi đã chứng minh cho khẳng định này.

Báo cáo này sẽ tập trung trình bày động lực, ý tưởng thuật toán và tổng hợp kết quả thực nghiệm, ứng dụng dựa trên việc cài đặt lại thuật toán, kèm nhận xét về bài báo.

Phần còn lại của báo cáo được tổ chức như sau: Mục 2 sẽ trình bày kiến thức cần biết; Mục 3 tóm lược thuật toán và các kết quả chính; Mục 4 giới thiệu các thí nghiệm số; Mục 5 nêu ra ứng dụng trong học máy; cuối cùng là kết luận và nhận xét.

2 Kiến thức cần biết

Trong toàn bộ bài báo, chúng ta giả định rằng C là một tập hợp **khác rỗng**, **đóng** và **lồi** trong $\mathbb{R}^m$, hàm $f : \mathbb{R}^m \to \mathbb{R}$ là một hàm khả vi trên một tập mở chứa C, ánh xạ ∇f là liên tục L-Lipschitz, tức là tồn tại một hằng số $L > 0$ sao cho

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$$

L-Lipschitz continuous : điều kiện về độ "mịn" của hàm gradient ∇f, đảm bảo rằng độ dốc của hàm không thay đổi quá nhanh. Với mọi $\mathbf{x}, \mathbf{y} \in C$. Chúng ta xét bài toán tối ưu:

$$\min_{\mathbf{x} \in C} f(\mathbf{x}), \quad OP(f, C)$$

Giả sử tập nghiệm của OP(f, C) là **không rỗng**.
Với $\mathbf{x} \in \mathbb{R}^m$, ký hiệu $P_C(\mathbf{x})$ là **phép chiếu** của $\mathbf{x}$ lên C, tức là:

$$P_C(\mathbf{x}) := \arg\min\{\|\mathbf{z} - \mathbf{x}\| : \mathbf{z} \in C\}.$$

Mệnh đề 1 (Bauschke and Combettes 2011 [1]). *Các điều sau đúng:*

(i) $\|P_C(\mathbf{x}) - P_C(\mathbf{y})\| \leq \|\mathbf{x} - \mathbf{y}\|$ *với mọi* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.

(ii) $(\mathbf{y} - P_C(\mathbf{x}))^T(\mathbf{x} - P_C(\mathbf{x})) \leq 0$ *với mọi* $\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in C$.

Định nghĩa 1 (Mangasarian 1965 [13]). *Hàm* $f : \mathbb{R}^m \to \mathbb{R}$ *được gọi là:*

- ***Lồi*** *(convex) trên* C *nếu với mọi* $\mathbf{x}, \mathbf{y} \in C, \lambda \in [0, 1]$, *thì*
$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}).$$
- ***Giả lồi*** *(pseudoconvex) trên* C *nếu với mọi* $\mathbf{x}, \mathbf{y} \in C$, *điều kiện sau đúng:*
$$\nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \geq 0 \implies f(\mathbf{y}) \geq f(\mathbf{x}).$$
- ***Tựa lồi*** *(quasiconvex) trên* C *nếu với mọi* $\mathbf{x}, \mathbf{y} \in C, \lambda \in [0, 1]$, *điều kiện sau đúng:*
$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \max\{f(\mathbf{x}); f(\mathbf{y})\}.$$

Mệnh đề 2 (Dennis và Schnabel 1983 [5]). *Hàm khả vi* f *là tựa lồi trên* C *khi và chỉ khi*
$$f(\mathbf{y}) \leq f(\mathbf{x}) \Rightarrow \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \leq 0.$$

Đáng chú ý là "hàm f lồi" $\Rightarrow$ "hàm f giả lồi" $\Rightarrow$ "hàm f tựa lồi"

Mệnh đề 3 (Dennis và Schnabel 1983 [5]). *Giả sử* ∇f *là liên tục L-Lipschitz trên* C. *Với mọi* $\mathbf{x}, \mathbf{y} \in C$, *điều kiện sau đúng:*
$$|f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})| \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

Bổ đề 1 (Xu 2002 [18]). *Cho* $\{a_k\}$ *và* $\{b_k\}$ *là các dãy số dương sao cho* $a_{k+1} \leq a_k + b_k \forall k \geq 0$ *và* $\sum_{k=0}^{\infty} b_k < \infty$. *Khi đó, tồn tại giới hạn* $\lim_{k\to\infty} a_k = c \in \mathbb{R}$.

3 Các kết quả chính

Algorithm 1 (Gradient Descent Adaptive Algorithm-GDA)

Input: $x^0 \in C$; $\lambda_0 > 0$; $\sigma, \kappa \in (0, 1)$; $k = 0$
Output: Stationary point x^*
while true **do**
  $x^{k+1} \leftarrow P_C\big(x^k - \lambda_k \nabla f(x^k)\big)$
  if $f(x^{k+1}) \leq f(x^k) - \sigma\langle \nabla f(x^k), x^k - x^{k+1}\rangle$ **then**
    $\lambda_{k+1} \leftarrow \lambda_k$
  else
    $\lambda_{k+1} \leftarrow \kappa\lambda_k$
  end if
  if $x^{k+1} = x^k$ **then**
    break
  end if
  $k \leftarrow k + 1$
end while

Nhận xét 1. Nếu Thuật toán GDA dừng tại bước k, thì $\mathbf{x}^k$ là một điểm dừng của bài toán OP(f, C). Thật vậy, vì $\mathbf{x}^{k+1} = P_C(\mathbf{x}^k - \lambda_k \nabla f(\mathbf{x}^k))$, áp dụng Mệnh đề 1 - *(ii)*, ta có:

$$\langle \mathbf{z} - \mathbf{x}^{k+1}, \mathbf{x}^k - \lambda_k \nabla f(\mathbf{x}^k) - \mathbf{x}^{k+1} \rangle \leq 0, \quad \forall \mathbf{z} \in C \tag{1}$$

Nếu $\mathbf{x}^{k+1} = \mathbf{x}^k$, ta thu được:

$$\langle \nabla f(\mathbf{x}^k), \mathbf{z} - \mathbf{x}^k \rangle \geq 0, \quad \forall \mathbf{z} \in C \tag{2}$$

điều này có nghĩa là $\mathbf{x}^k$ là một điểm dừng của bài toán. Hơn nữa, nếu f là hàm giả lồi, từ (2) suy ra $f(\mathbf{z}) \geq f(\mathbf{x}^k)$ với mọi $\mathbf{z} \in C$, hay $\mathbf{x}^k$ là một nghiệm của OP(f, C).

Giả sử thuật toán sinh ra một dãy vô hạn. Chúng ta sẽ chứng minh rằng dãy này hội tụ về một nghiệm của bài toán OP(f, C).

Định lý 1. Giả sử dãy $\{\mathbf{x}^k\}$ được sinh bởi Thuật toán GDA. Khi đó, dãy $\{f(\mathbf{x}^k)\}$ hội tụ và mỗi điểm giới hạn (nếu có) của dãy $\{\mathbf{x}^k\}$ là một điểm dừng của bài toán. Hơn nữa:

- Nếu f là hàm tựa lồi trên C, thì dãy $\{\mathbf{x}^k\}$ hội tụ về một điểm dừng của bài toán.
- Nếu f là hàm giả lồi trên C, thì dãy $\{\mathbf{x}^k\}$ hội tụ về một nghiệm của bài toán.

Chứng minh. Áp dụng Mệnh đề 2 (tính chất Lipschitz), ta có:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2. \tag{3}$$

Trong (1), chọn $\mathbf{z} = \mathbf{x}^k \in C$, ta đi đến:

$$\langle z - x^{k+1}, x^k - \lambda_k \nabla f(x^k) - x^{k+1} \rangle \leq 0, \quad \forall z \in C$$

$$\text{Với } z = x^k: \quad \langle x^k - x^{k+1}, (x^k - x^{k+1}) - \lambda_k \nabla f(x^k) \rangle \leq 0$$

$$\langle x^k - x^{k+1}, x^k - x^{k+1} \rangle - \langle x^k - x^{k+1}, \lambda_k \nabla f(x^k) \rangle \leq 0$$

$$\|x^k - x^{k+1}\|^2 - \lambda_k \langle -(x^{k+1} - x^k), \nabla f(x^k) \rangle \leq 0$$

$$\|x^{k+1} - x^k\|^2 + \lambda_k \langle x^{k+1} - x^k, \nabla f(x^k) \rangle \leq 0$$

$$\lambda_k \langle \nabla f(x^k), x^{k+1} - x^k \rangle \leq -\|x^{k+1} - x^k\|^2$$

$$\langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle \leq -\frac{1}{\lambda_k} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2. \tag{4}$$

Kết hợp (3) và (4), ta thu được:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \sigma \langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1} \rangle - \left(\frac{1 - \sigma}{\lambda_k} - \frac{L}{2} \right) \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2. \tag{5}$$

Ta khẳng định rằng $\{\lambda_k\}$ bị chặn dưới dương, hay nói cách khác, bước nhảy chỉ thay đổi hữu hạn lần. Thật vậy, giả sử ngược lại rằng $\lambda_k \to 0$. Từ (5), tồn tại $k_0 \in \mathbb{N}$ thỏa mãn:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \sigma\langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle \quad \forall k \geq k_0.$$

Theo cách xây dựng λ_k, bất đẳng thức cuối cùng ngụ ý rằng $\lambda_k = \lambda_{k_0}$ với mọi $k \geq k_0$. Điều này mâu thuẫn. Do đó, tồn tại $k_1 \in \mathbb{N}$ sao cho với mọi $k \geq k_1$, ta có $\lambda_k = \lambda_{k_1}$ và:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \sigma\langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle. \tag{6}$$

Lưu ý rằng $\langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle \geq 0$, ta suy ra dãy $\{f(\mathbf{x}^k)\}$ hội tụ và:

$$\sum_{k=0}^{\infty}\langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle < \infty; \quad \sum_{k=0}^{\infty}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 < \infty. \tag{7}$$

Từ (1), với mọi $\mathbf{z} \in C$, ta có:

$$\begin{aligned}\|\mathbf{x}^{k+1} - \mathbf{z}\|^2 &= \|\mathbf{x}^k - \mathbf{z}\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 + 2\langle \mathbf{x}^{k+1} - \mathbf{x}^k, \mathbf{x}^{k+1} - \mathbf{z}\rangle \\ &\leq \|\mathbf{x}^k - \mathbf{z}\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 + 2\lambda_k\langle \nabla f(\mathbf{x}^k), \mathbf{z} - \mathbf{x}^{k+1}\rangle. \end{aligned} \tag{8}$$

Gọi $\bar{\mathbf{x}}$ là một điểm giới hạn của $\{\mathbf{x}^k\}$. Tồn tại một dãy con $\{\mathbf{x}^{k_i}\} \subset \{\mathbf{x}^k\}$ sao cho $\lim_{i\to\infty}\mathbf{x}^{k_i} = \bar{\mathbf{x}}$. Trong (8), đặt $k = k_i$ và lấy giới hạn khi $i \to \infty$. Lưu ý rằng $\|\mathbf{x}^{k_i} - \mathbf{x}^{k_i+1}\| \to 0$ và ∇f liên tục, ta thu được:

$$\langle \nabla f(\bar{\mathbf{x}}), \mathbf{z} - \bar{\mathbf{x}}\rangle \geq 0 \quad \forall \mathbf{z} \in C,$$

điều này có nghĩa $\bar{\mathbf{x}}$ là một điểm dừng của bài toán.

Giả sử f là hàm tựa lồi trên C. Đặt:

$$U := \{\mathbf{x} \in C : f(\mathbf{x}) \leq f(\mathbf{x}^k), \forall k \geq 0\}.$$

Lấy $\hat{\mathbf{x}} \in U$. Vì $f(\mathbf{x}^k) \geq f(\hat{\mathbf{x}})$ với mọi $k \geq 0$, suy ra:

$$\langle \nabla f(\mathbf{x}^k), \hat{\mathbf{x}} - \mathbf{x}^k\rangle \leq 0, \quad \forall k \geq 0. \tag{9}$$

Kết hợp (8) và (9), ta có:

$$\|\mathbf{x}^{k+1} - \hat{\mathbf{x}}\|^2 \leq \|\mathbf{x}^k - \hat{\mathbf{x}}\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 + 2\lambda_k\langle \nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle. \tag{10}$$

Áp dụng Bổ đề 1 với $a_k = \|\mathbf{x}^{k+1} - \hat{\mathbf{x}}\|^2$ và $b_k = 2\lambda_k\nabla f(\mathbf{x}^k)^T(\mathbf{x}^k - \mathbf{x}^{k+1})$, ta suy ra dãy $\{\|\mathbf{x}^k - \hat{\mathbf{x}}\|\}$ hội tụ với mọi $\hat{\mathbf{x}} \in U$. Vì dãy $\{\mathbf{x}^k\}$ bị chặn, tồn tại dãy con $\{\mathbf{x}^{k_j}\} \subset \{\mathbf{x}^k\}$ sao cho $\lim_{j\to\infty}\mathbf{x}^{k_j} = \bar{\mathbf{x}} \in C$.

Từ (6), ta biết dãy $\{f(\mathbf{x}^k)\}$ không tăng và hội tụ. Suy ra $\lim_{k\to\infty} f(\mathbf{x}^k) = f(\bar{\mathbf{x}})$ và $f(\bar{\mathbf{x}}) \leq f(\mathbf{x}^k)$ với mọi $k \geq 0$. Điều này có nghĩa là $\bar{\mathbf{x}} \in U$ và dãy $\{\|\mathbf{x}^k - \bar{\mathbf{x}}\|\}$ hội tụ. Do đó:

$$\lim_{k\to\infty}\|\mathbf{x}^k - \bar{\mathbf{x}}\| = \lim_{j\to\infty}\|\mathbf{x}^{k_j} - \bar{\mathbf{x}}\| = 0.$$

Lưu ý rằng mỗi điểm giới hạn của $\{\mathbf{x}^k\}$ là một điểm dừng của bài toán. Khi đó, toàn bộ dãy $\{\mathbf{x}^k\}$ hội tụ về $\bar{\mathbf{x}}$ – một điểm dừng của bài toán. Hơn nữa, khi f là hàm giả lồi, điểm dừng này trở thành một nghiệm của OP(f, C). □

Nhận xét 2. Trong Thuật toán GDA, ta có thể chọn $\lambda_0 = \lambda$, với hằng số $\lambda \leq 2(1-\sigma)/L$. Khi đó, ta có $(1-\sigma)/\lambda_0 - L/2 \geq 0$. Kết hợp với (5), điều này ngụ ý rằng điều kiện $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \sigma\langle\nabla f(\mathbf{x}^k), \mathbf{x}^k - \mathbf{x}^{k+1}\rangle$ được thỏa mãn và bước nhảy $\lambda_k = \lambda$ cho mọi bước k. Do đó, Thuật toán GDA vẫn áp dụng được cho bước nhảy hằng số $\lambda \leq 2(1-\sigma)/L$.

Với bất kỳ $\lambda \in (0, 2/L)$, tồn tại $\sigma \in (0,1)$ sao cho $\lambda \leq 2(1-\sigma)/L$. Kết quả là, nếu giá trị hằng số Lipschitz L được biết trước, ta có thể chọn bước nhảy hằng số $\lambda \in (0, 2/L)$ như trong thuật toán gradient descent (GD) để giải các bài toán quy hoạch lồi. Thuật toán GD này đã được đề xuất trong các công trình trước đây. Vì nó là một trường hợp đặc biệt của Thuật toán GDA, sự hội tụ của nó được đảm bảo như các khẳng định trong Định lý 1

Algorithm 2 (Gradient Descent Algorithm-GD)

1: **Input:** $x^0 \in C$; $\lambda_0 \in (0, 2/L)$; $k = 0$
2: **Output: Stationary point** x^*
3: **while** true **do**
4: $\quad x^{k+1} \leftarrow P_C\big(x^k - \lambda_k \nabla f(x^k)\big)$
5: $\quad$ **if** $x^{k+1} = x^k$ **then**
6: $\quad\quad$ **break**
7: $\quad$ **end if**
8: $\quad k \leftarrow k+1$
9: **end while**

Lưu ý rằng tất cả những đánh giá ở Định lý 1 vẫn đều đúng đối với dãy $\{x^k\}$ sinh bởi thuật toán GD. Giờ chúng ta sẽ đánh giá tốc độ hội tụ của thuật toán GDA trong việc giải quyết bài toán tối ưu không ràng buộc.

Hệ quả 1. *Giả sử f lồi, $C = \mathbb{R}^m$ và $\{x^k\}$ là dãy được sinh ra bởi thuật toán GDA. Khi đó,*

$$f(x^k) - f(x^*) = O\left(\frac{1}{k}\right).$$

với x^* là nghiệm tối ưu của bài toán.

Chứng minh. Gọi x^* là nghiệm tối ưu của bài toán. Đặt $\Delta_k := f(x^k) - f(x^*)$. Từ (6), với chú ý rằng $x^k - x^{k+1} = \lambda_k \nabla f(x^k)$, ta thu được

$$\Delta_{k+1} \leq \Delta_k - \sigma\lambda_{k_1}\|\nabla f(x^k)\|^2 \quad \forall k \geq k_1. \tag{11}$$

Mặt khác, do dãy $\{x^k\}$ bị chặn và f lồi nên

$$\begin{aligned} 0 \leq \Delta_k &\leq \big\langle \nabla f(x^k), x^k - x^* \big\rangle \\ &\leq M\|\nabla f(x^k)\|, \end{aligned} \tag{12}$$

trong đó $M := \sup\{\|x^k - x^*\| : k \geq k_1\} < \infty$. Từ (11) và (12), ta được

$$\Delta_{k+1} \leq \Delta_k - Q\Delta_k^2 \quad \forall k \geq k_1, \tag{13}$$

trong đó $Q := \frac{\sigma\lambda_{k_1}}{M^2}$. Lưu ý rằng $\Delta_{k+1} \leq \Delta_k$, nên từ (13), ta thu được

$$\frac{1}{\Delta_{k+1}} \geq \frac{1}{\Delta_k} + Q \geq ... \geq \frac{1}{\Delta_{k_1}} + (k - k_1)Q,$$

ngụ ý rằng

$$f(x^k) - f(x^*) = O\left(\frac{1}{k}\right).$$

□

Để kết thúc mục này, một biến thể ngẫu nhiên của Thuật toán GDA được trình bày nhằm ứng dụng trong học sâu quy mô lớn. Hãy xem xét bài toán sau:

$$\min_x \mathbb{E}[f_\xi(x)],$$

trong đó ξ là tham số ngẫu nhiên và hàm f_ξ là L-smooth. Bài báo tạo ra gradient ngẫu nhiên $\nabla f_{\xi^k}(x^k)$ bằng cách lấy mẫu ξ tại mỗi lần lặp k.

Biến thể ngẫu nhiên của phương pháp hướng giảm gradient, đặc biệt trong bối cảnh học sâu quy mô lớn, đóng vai trò quan trọng trong việc tối ưu hóa hiệu quả các mô hình phức tạp. Khi chúng ta xem xét bài toán tối ưu hóa dưới dạng:

$$\min_x \mathbb{E}[f_\xi(x)],$$

trong đó x biểu diễn các tham số của mô hình (như trọng số trong mạng nơ-ron), ξ là tham số ngẫu nhiên, và $f_\xi(x)$ là một hàm L-smooth, ta đang làm việc với một tình huống trong đó hàm mục tiêu được định nghĩa là kỳ vọng của một hàm ngẫu nhiên $f_\xi(x)$. Trong lĩnh vực học máy, $f_\xi(x)$ thường là hàm mất mát được tính trên một tập con (batch) của dữ liệu huấn luyện, và ξ đại diện cho tính ngẫu nhiên của việc chọn tập con này.

Thuật toán SGDA sau đây chứa mô tả chi tiết. Các kết quả lý thuyết chặt chẽ của Thuật toán SGDA được các tác giả của bài báo để lại cho nghiên cứu trong tương lai.

Algorithm 3 (Stochastic Gradient Descent Algorithm-SGDA)

Input: $x^0 \in C$; $\lambda_0 > 0$; $\sigma, \kappa \in (0, 1)$; $k = 0$
Output: Stationary point x^*
while true **do**
    Lấy mẫu ngẫu nhiên ξ_k
    $x_{k+1} \leftarrow P_C\big(x_k - \lambda_k \nabla f_{\xi_k}(x_k)\big)$
    if $f_{\xi_k}(x_{k+1}) \leq f_{\xi_k}(x_k) - \sigma\langle \nabla f_{\xi_k}(x_k), x_k - x_{k+1}\rangle$ **then**
        $\lambda_{k+1} \leftarrow \lambda_k$
    else
        $\lambda_{k+1} \leftarrow \kappa\lambda_k$
    end if
    if $x_{k+1} = x_k$ **then**
        break
    end if
    $k \leftarrow k + 1$
end while

4 Các thí nghiệm số

Trong mục này, chúng tôi sử dụng hai bài toán có sẵn và hai ví dụ quy mô lớn với kích thước biến thay đổi để kiểm chứng hiệu quả của phương pháp được đề xuất. Các thí nghiệm ứng dụng trong học máy sẽ được trình bày ở mục tiếp theo. Mã nguồn được công bố tại: https://github.com/nguyenphongg233/NMPPTU-Team3-SAAQP.

Tiêu chí dừng trong các ví dụ sau là: "số vòng lặp $\leq$ #Iter", trong đó #Iter là số vòng lặp tối đa. Kí hiệu x^* là điểm giới hạn của dãy $\{x_k\}$ và Time là thời gian CPU của thuật toán GDA theo tiêu chí dừng.

Các hàm mục tiêu và ràng buộc phi lồi được mô tả trong Ví dụ 1 và 2 từ Liu et al. (2022) [11]. Trong đó Ví dụ 2 phức tạp hơn Ví dụ 1. Ví dụ 3 sử dụng hàm mục tiêu lồi với số chiều thay đổi n để đánh giá thuật toán GDA so với GD. Ví dụ 4 dùng hàm mục tiêu giả lồi và nhiều giá trị n để so sánh với thuật toán RNN.

Để so sánh với phương pháp Neurodynamic (RNN) của Liu et al. (2022) [11], xét bài toán $\mathrm{OP}(f, C)$ với:

$$C = \{x \in \mathbb{R}^n \mid g(x) \leq 0,\ Ax = b\},$$

trong đó

$$g(x) := (g_1(x), g_2(x), \ldots, g_m(x))^\top, \qquad g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m,$$

là các hàm quasi-convex khả vi, ma trận $A \in \mathbb{R}^{p\times n}$ và $b = (b_1, b_2, \ldots, b_p)^\top \in \mathbb{R}^p$.

Trong Liu et al. (2022) [11], tác giả định nghĩa:

$$\Psi(s) = \begin{cases} 1, & s > 0, \\ [0,1], & s = 0, \\ 0, & s < 0, \end{cases} \qquad P(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\}.$$

Thuật toán RNN dạng bao hàm vi phân:

$$\frac{d}{dt}x(t) \in -c(x(t))\nabla f(x(t)) - \partial P(x(t)) - \partial\|Ax(t) - b\|_1,$$

trong đó

$$c(x(t)) = \sum_{i=1}^{m+p} c_i(t), \qquad c_i(t) \in 1 - \Psi(J_i(x(t))),$$

với

$$J(x) = (g_1(x), \ldots, g_m(x), |A_1x - b_1|, \ldots, |A_px - b_p|)^\top.$$

Phần tử dưới đạo hàm của $P(x)$:

$$\partial P(x) = \begin{cases} 0, & x \in \mathrm{int}(X), \\ \sum_{i\in I_0(x)}[0,1]\nabla g_i(x), & x \in \mathrm{bd}(X), \\ \sum_{i\in I_+(x)} \nabla g_i(x) + \sum_{i\in I_0(x)}[0,1]\nabla g_i(x), & x \notin X, \end{cases}$$

với

$$X = \{x : g_i(x) \leq 0\}, \quad I_+(x) = \{i : g_i(x) > 0\}, \quad I_0(x) = \{i : g_i(x) = 0\}.$$

Với $\|Ax - b\|_1$:

$$\partial \|Ax - b\|_1 = \sum_{i=1}^{p} (2\Psi(A_i x - b_i) - 1)\, A_i^\top.$$

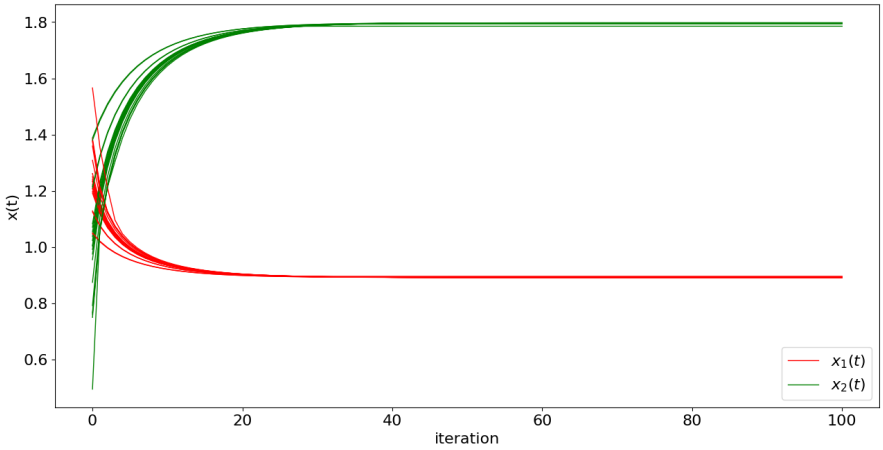
Ví dụ 1

Xét bài toán phi lồi:

$$\min f(x) = \frac{x_1^2 + x_2^2 + 3}{1 + 2x_1 + 8x_2},$$

với ràng buộc:

$$C = \{(x_1, x_2)^\top \in \mathbb{R}^2 \mid -x_1^2 - 2x_1 x_2 \leq -4,\ x_1, x_2 \geq 0\}.$$



Hình 1: Example 1

Hàm f là pseudoconvex trên C. Thuật toán GDA hội tụ đến nghiệm tối ưu:

$$x^* = (0.893870,\ 1.790527),$$

giá trị tối ưu của GDA là 0.409362, tốt hơn phương pháp RNN (0.410100).

#	Time (s)	Iters	Value	$\boldsymbol{x}^*$
1	1.489085	**28**	0.409364	[0.894247, 1.789397]
2	1.235719	32	**0.409362**	[0.893870, 1.790527]
3	1.082686	31	0.409363	[0.893946, 1.790301]
4	**1.047740**	31	0.409363	[0.894092, 1.789862]
5	1.440842	32	0.409364	[0.894284, 1.789285]
6	1.139069	30	0.409367	[0.895131, 1.786744]
7	1.126395	32	0.409363	[0.894047, 1.789995]
8	1.564762	33	0.409363	[0.893926, 1.790359]
9	2.008716	33	0.409363	[0.893998, 1.790141]
10	1.510566	32	**0.409362**	[0.893911, 1.790403]

Bảng 1: Experimental results

Nhận xét: Hình 1 minh họa động lực hội tụ của các thành phần nghiệm theo số vòng lặp. Các quỹ đạo nghiệm có sự thay đổi nhanh ở giai đoạn đầu, sau đó tiến dần tới trạng thái ổn định, phản ánh đặc trưng hội tụ của phương pháp gradient thích nghi. Việc các quỹ đạo hội tụ về cùng một điểm cho thấy nghiệm tối ưu có tính hấp dẫn và thuật toán GDA không nhạy với nhiễu nhỏ trong quá trình lặp. Bảng 1 trình bày kết quả của 20 lần chạy độc lập cho Ví dụ 1. Kết quả cho thấy giá trị hàm mục tiêu thu được có độ biến thiên rất nhỏ giữa các lần chạy, chứng tỏ thuật toán có tính ổn định cao đối với điều kiện khởi tạo và các yếu tố ngẫu nhiên. Mặc dù thời gian tính toán giữa các lần chạy có sự dao động nhất định, giá trị tối ưu đạt được vẫn nhất quán, cho thấy thuật toán ưu tiên chất lượng nghiệm hơn sự tối ưu tuyệt đối về thời gian trong từng lần chạy riêng lẻ. Điều này phản ánh đặc điểm thực nghiệm quan trọng: chất lượng nghiệm và chi phí tính toán là hai tiêu chí độc lập, và thuật toán GDA thể hiện sự cân bằng tốt giữa hai yếu tố này.

Ví dụ 2

Xét bài toán giả lồi, không trơn:

$$\begin{aligned} \min \ f(x) &= \frac{e^{|x_2-3|}-30}{x_1^2+x_3^2+2x_4^2+4}, \\ \text{với ràng buộc} \quad g_1(x) &= (x_1+x_3)^3+2x_4^2 \le 10, \\ g_2(x) &= (x_2-1)^2 \le 1, \\ 2x_1+4x_2+x_3 &= -1. \end{aligned}$$

Do $x_2 \neq 3$ với mọi x khả thi:

$$\nabla|x_2-3| = \frac{x_2-3}{|x_2-3|}.$$

GDA hội tụ đến:

$$x^* = (-1.067557,\ 0.417538,\ -0.535037,\ 0.000020)^\top,$$

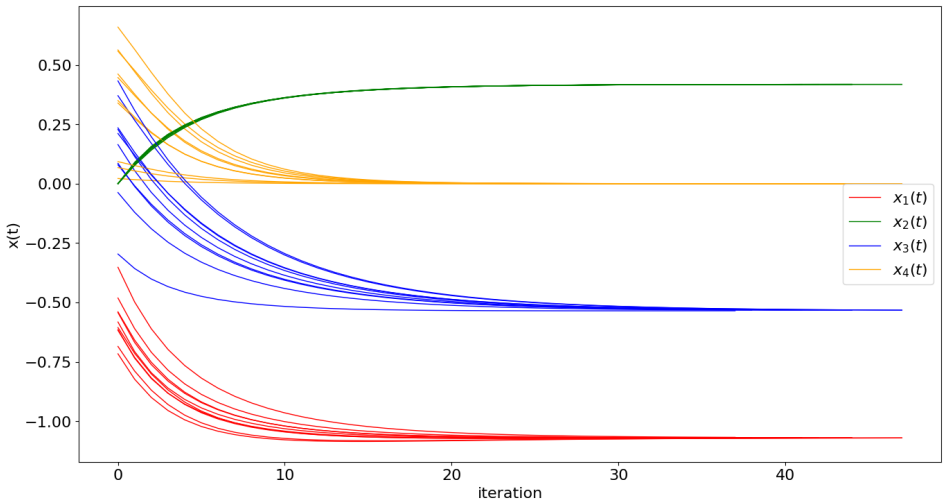
giá trị tối ưu:

$$f(x^*) = -3.090768,$$

tốt hơn RNN (-3.084900).

#	Time (s)	Iters	Value	$\boldsymbol{x}^*$
1	3.125859	38	-3.090767	[-1.069016, 0.417647, -0.532558, 0.000022]
2	3.376504	42	-3.090767	[-1.069795, 0.417966, -0.532273, 0.000000]
3	3.359701	41	-3.090767	[-1.069610, 0.417896, -0.532363, 0.000014]
4	3.596762	44	-3.090767	[-1.069956, 0.418058, -0.532320, 0.000005]
5	3.124562	41	-3.090767	[-1.069584, 0.417901, -0.532434, 0.000007]
6	3.700269	47	-3.090767	[-1.070102, 0.418157, -0.532424, 0.000004]
7	3.406564	44	-3.090767	[-1.069881, 0.418059, -0.532475, 0.000001]
8	**2.954161**	**37**	**-3.090768**	[-1.067557, 0.417538, -0.535037, 0.000020]
9	3.307867	47	-3.090767	[-1.070151, 0.418162, -0.532345, 0.000000]
10	3.687243	44	-3.090767	[-1.069958, 0.418055, -0.532303, 0.000009]

Bảng 2: Experimental results



Hình 2: Example 2

Nhận xét: Mặc dù nghiệm x^* và số vòng lặp có sự dao động rất nhỏ giữa các lần thử nghiệm, giá trị hàm mục tiêu đạt được gần như không thay đổi, cho thấy tính ổn định và độ tin cậy cao của thuật toán GDA đối với bài toán này.

So với phương pháp Neurodynamic (RNN), thuật toán GDA đạt được giá trị hàm mục tiêu nhỏ hơn (RNN cho giá trị -3.084900), đồng thời yêu cầu số vòng lặp và thời gian tính toán thấp hơn. Kết quả này khẳng định hiệu quả của GDA khi áp dụng cho các bài toán tối ưu giả lồi, không trơn với ràng buộc hỗn hợp.

Ví dụ 3

Cho $e := (1, \ldots, n) \in \mathbb{R}^n$ là một vector, $\alpha > 0$ và $\beta > 0$ là các hằng số thỏa mãn điều kiện tham số

$$2\alpha > 3\beta^{3/2}\sqrt{n}.$$

Xét bài toán OP(f, C) (Ví dụ 4.5 trong Ferreira và Sosa (2022) [6]) với hàm mục tiêu

$$f(x) := a^\top x + \alpha x^\top x + \frac{\beta}{\sqrt{1+\beta x^\top x}}\, e^\top x,$$

ràng buộc:

$$C = \{x \in \mathbb{R}^n_{++} : 1 \le x_1 x_2 \cdots x_n\}.$$

Ta chọn $\beta = 0.741271$, $\alpha = 3\beta^{3/2}\sqrt{n+1}$. Hệ số Lipschitz:

$$L = 4\beta^{3/2}\sqrt{n} + 3\alpha.$$

GD: $\lambda = 1/L$. GDA: $\lambda_0 = 5/L$.

n	GDA			GD		
	$f(x^*)$	#Iter	Time(s)	$f(x^*)$	#Iter	Time(s)
10	**80.080568**	**3**	**0.2447**	80.080573	7	0.3834
20	**219.714451**	**4**	**0.6099**	219.714455	8	0.6726
50	**851.700027**	**4**	**0.9028**	851.700030	9	2.5383
100	**2390.314031**	**4**	**5.6882**	2390.314040	9	13.3452
200	**2390.314040**	**5**	**41.3877**	6721.095887	10	53.0749
500	**26426.158664**	**5**	**91.4714**	26426.158682	10	262.8033
600	**34711.054331**	**5**	**160.1987**	34711.054335	11	353.6806
700	**43704.962204**	**5**	**199.9534**	43704.962212	16	1726.5777
800	**53372.127462**	**5**	**352.0395**	53372.127469	11	930.0608
900	**63653.989656**	**11**	**1589.4307**	63653.989663	11	2123.5983
1000	**74522.902256**	**7**	**1260.0110**	74522.902265	11	1617.5378

Bảng 3: Comparison between GDA and GD

Nhận xét: Bảng 3 cho thấy thuật toán GDA đạt được giá trị hàm mục tiêu tối ưu với số vòng lặp nhỏ và ổn định khi số chiều n tăng. So với GD, GDA yêu cầu ít vòng lặp hơn và có thời gian tính toán ngắn hơn, đặc biệt trong các bài toán quy mô lớn.

Ví dụ 4

Để so sánh thuật toán GDA với thuật toán RNN trong Liu et al. (2022) [11], xét bài toán OP(f, C) với hàm mục tiêu:

$$f(x) = -\exp\left(-\sum_{i=1}^{n} \frac{x_i^2}{e_i^2}\right),$$

hàm này là giả lồi trên miền ràng buộc lồi:

$$C := \{Ax = b,\ g(x) \le 0\}.$$

Ở đây $x \in \mathbb{R}^n$, vector tham số $e = (e_1, e_2, \ldots, e_n)^T$ với $e_i > 0$. Ma trận $A = (a_1, \ldots, a_n) \in \mathbb{R}^{1\times n}$ được xác định bởi

$$a_i = \begin{cases} 1, & 1 \le i \le n/2, \\ 3, & n/2 < i \le n, \end{cases} \qquad b = 16.$$

Các ràng buộc bất đẳng thức được cho bởi

$$g_i(x) = x^2_{10(i-1)+1} + x^2_{10(i-1)+2} + \cdots + x^2_{10(i-1)+10} - 20, \qquad i = 1, 2, \ldots, n/10.$$

Bảng 4: Computational results for Example 4

n	Algorithm GDA (proposed)			Algorithm RNN		
	$-\ln(-f(x^*))$	#Iter	Time	$-\ln(-f(x^*))$	#Iter	Time
10	**5.1200**	**50**	**2.597**	5.2014	5000	1123
20	**2.56000**	**50**	**4.542**	2.66910	5000	1264
30	**1.70667**	**50**	**5.448**	2.0824	5000	1562
40	**1.28000**	**50**	**16.3**	1.38552	5000	1662.3
50	**1.02400**	**50**	**12.31**	1.13281	5000	1741.3
60	**0.85334**	**500**	**248.5**	1.07326	5000	1984.8
70	**0.73143**	**500**	**78.4**	2.30464	5000	1443.2
80	**0.64000**	**500**	**298**	7.98621	5000	2338.6
90	**0.56889**	**500**	**272.3**	9.12685	5000	2743.2
100	**0.51200**	**500**	**617.64**	10.88262	5000	2985.26
150	**0.34134**	**500**	**1508**	17.69445	5000	3240.36
200	**0.25600**	**500**	**1748.3**	18.99267	5000	4004.87

Bảng 4 trình bày kết quả tính toán của hai thuật toán GDA và RNN. Lưu ý rằng hàm $-\ln(-z)$ là hàm đơn điệu tăng đối với mọi $z < 0$. Do đó, để so sánh kết quả khi số chiều n lớn, ta dùng chỉ số $-\ln(-f(x^*))$ thay cho $f(x^*)$, với x^* là nghiệm gần đúng của bài toán.

Với mỗi giá trị n, ta giải bài toán OP(f, C), thu được các giá trị $-\ln(-f(x^*))$, số vòng lặp (#Iter), và thời gian tính toán (Time). Kết quả trong Bảng 4 cho thấy thuật toán GDA vượt trội rõ rệt so với thuật toán RNN về giá trị tối ưu gần đúng, số vòng lặp và thời gian tính toán, đặc biệt khi kích thước bài toán lớn.

5 Ứng dụng trong học máy

Phương pháp được đề xuất, giống như thuật toán GD, có nhiều ứng dụng trong học máy. Chúng tôi phân tích ba ứng dụng học máy phổ biến, cụ thể là lựa chọn đặc trưng có giám sát, hồi

quy và phân loại, để chứng minh độ chính xác và hiệu quả tính toán so với các phương pháp thay thế khác.

Đầu tiên, bài toán lựa chọn đặc trưng có thể được mô hình hóa thành một bài toán cực tiểu hóa của một hàm phân thức giả lồi trên một tập lồi, đây là một lớp con của Bài toán OP(f, C). Bài toán này được sử dụng để so sánh phương pháp đề xuất với các phương pháp thần kinh động lực học (neurodynamic approaches).

Thứ hai, vì bài toán hồi quy logistic đa biến là một bài toán quy hoạch lồi, thuật toán GDA và các biến thể có sẵn của thuật toán GD có thể được sử dụng để giải quyết nó.

Cuối cùng, một mô hình mạng nơ-ron cho bài toán phân loại hình ảnh cũng tương tự như một bài toán quy hoạch với hàm mục tiêu không lồi cũng không phải giả lồi (quasiconvex). Để huấn luyện mô hình này, chúng tôi sử dụng biến thể ngẫu nhiên của phương pháp GDA (Thuật toán SGDA) như một kỹ thuật heuristic. Mặc dù sự hội tụ của thuật toán không thể được đảm bảo như trong các trường hợp của hàm mục tiêu giả lồi và tựa lồi, nghiên cứu lý thuyết đã chỉ ra rằng nếu dãy các điểm có một điểm giới hạn, nó sẽ hội tụ về một điểm dừng (stationary point) của bài toán (xem Định lý 1). Các thực nghiệm tính toán chỉ ra rằng phương pháp đề xuất vượt trội hơn các phương pháp thần kinh động lực học và giảm gradient (gradient descent) hiện có.

5.1 Supervised feature selection

Bài toán lựa chọn đặc trưng được thực hiện trên tập dữ liệu với tập p-đặc trưng $\mathcal{F} = \{F_1, ..., F_p\}$ và tập n-mẫu $\{(x_i, y_i)|i = 1, ..., n\}$, trong đó $x_i = (x_{i1}, ..., x_{ip})^T$ là một vectơ đặc trưng p-chiều của mẫu thứ i và $y_i \in \{1, ..., m\}$ đại diện cho các nhãn tương ứng chỉ ra các lớp hoặc giá trị mục tiêu. Trong Wang et al. (2021) [17], một tập hợp con tối ưu gồm k đặc trưng $\{F_1, ..., F_k\} \subseteq \mathcal{F}$ được chọn với sự dư thừa ít nhất và mức độ liên quan cao nhất đến lớp mục tiêu y.

Sự dư thừa của đặc trưng được đặc trưng bởi một ma trận bán xác định dương Q. Do đó, mục tiêu đầu tiên là cực tiểu hóa hàm bậc hai lồi w^TQw. Mức độ liên quan của đặc trưng được đo lường bằng ρ^Tw, trong đó $\rho = (\rho_1, ..., \rho_p)^T$ là một vectơ tham số liên quan. Vì vậy, mục tiêu thứ hai là cực đại hóa hàm tuyến tính ρ^Tw. Kết hợp hai mục tiêu này ta suy ra bài toán tương đương như sau:

$$\begin{aligned} \text{minimize} \quad & \frac{w^TQw}{\rho^Tw} \\ \text{subject to} \quad & e^Tw = 1 \\ & w \geq 0, \end{aligned} \tag{14}$$

trong đó $w = (w_1, ..., w_p)^T$ là vectơ điểm số đặc trưng cần được xác định. Vì hàm mục tiêu của bài toán (14) là phân thức của một hàm lồi trên một hàm tuyến tính dương, nên nó là hàm giả lồi trên tập ràng buộc. Do đó, chúng ta có thể giải bài toán (14) bằng Thuật toán GDA.

Trong thực nghiệm, chúng tôi triển khai các thuật toán với tập dữ liệu Parkinsons, bao gồm 23 đặc trưng và 197 mẫu, được tải xuống tại https://archive.ics.uci.edu/ml/datasets/parkinsons. Ma trận hệ số tương đồng Q được xác định bởi $Q = \delta I_p + S$ (xem Wang et al. 2021), trong đó ma trận $p \times p$ là $S = (s_{ij})$ với:

$$s_{ij} = \max\left\{0, \frac{I(F_i; F_j; y)}{H(F_i) + H(F_j)}\right\}$$

entropy thông tin của một vectơ biến ngẫu nhiên $\hat{X}$ là:

$$H(\hat{X}) = -\sum_{\hat{x}\in\hat{X}} p(\hat{x}) \log p(\hat{x}),$$

đa thông tin (multi-information) của ba vectơ ngẫu nhiên $\hat{X}$, $\hat{Y}$, $\hat{Z}$ là $I(\hat{X};\hat{Y};\hat{Z}) = I(\hat{X};\hat{Y}) - I(\hat{X};\hat{Y}|\hat{Z})$ với thông tin tương hỗ của hai vectơ ngẫu nhiên $\hat{X}$, $\hat{Y}$ được định nghĩa bởi:

$$I(\hat{X};\hat{Y}) = \sum_{\hat{x}\in\hat{X}} \sum_{\hat{y}\in\hat{Y}} p(\hat{x},\hat{y}) \log \frac{p(\hat{x},\hat{y})}{p(\hat{x})p(\hat{y})}$$

và thông tin tương hỗ có điều kiện giữa $\hat{X}$, $\hat{Y}$ và $\hat{Z}$ được định nghĩa bởi:

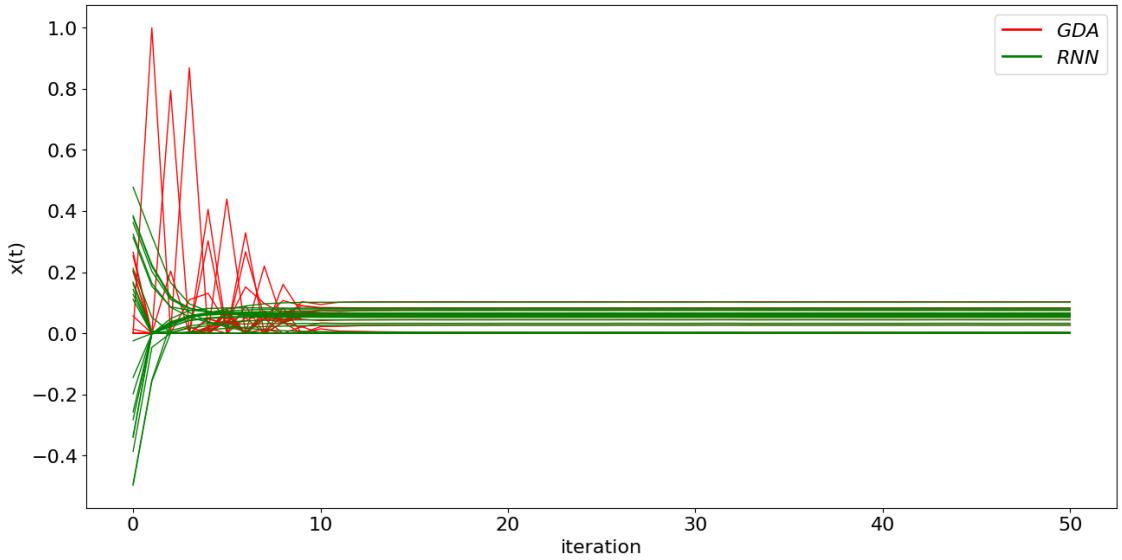
$$I(\hat{X};\hat{Y}|\hat{Z}) = \sum_{\hat{x}\in\hat{X}} \sum_{\hat{y}\in\hat{Y}} \sum_{\hat{z}\in\hat{Z}} p(\hat{x},\hat{y},\hat{z}) \log \frac{p(\hat{x},\hat{y}|\hat{z})}{p(\hat{x}|\hat{z})p(\hat{y}|\hat{z})}.$$

Vectơ mức độ liên quan đặc trưng $\rho = (\rho_1, ..., \rho_p)^T$ được xác định bởi điểm số Fisher:

$$\rho(F_i) = \frac{\sum_{j=1}^{K} n_j(\mu_{ij} - \mu_i)^2}{\sum_{j=1}^{K} n_j \sigma_{ij}^2},$$

trong đó n_j biểu thị số lượng mẫu trong lớp j, μ_{ij} biểu thị giá trị trung bình của đặc trưng F_i cho các mẫu trong lớp j, μ_i là giá trị trung bình của đặc trưng F_i, và σ_{ij}^2 biểu thị phương sai của đặc trưng F_i cho các mẫu trong lớp j.

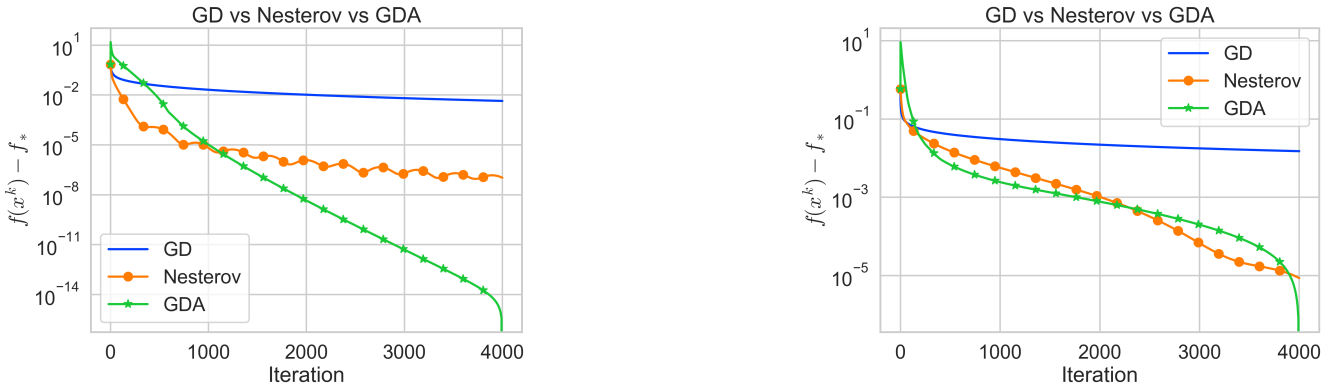
Giá trị tối ưu xấp xỉ của bài toán (14) là $f(w^*) = 0.153478$ với thời gian tính toán $T = 10.034473s$ cho thuật toán đề xuất, trong khi $f(w^*) = 0.153911$, $T = 49.324688s$ cho Thuật toán RNN. So với Thuật toán RNN, thuật toán của chúng tôi vượt trội hơn cả về độ chính xác và thời gian tính toán.



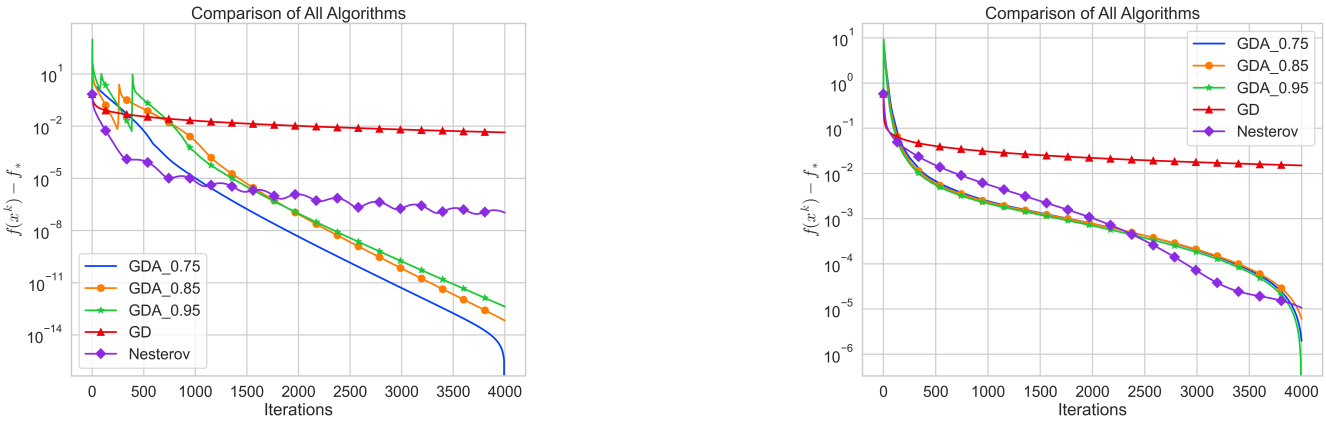
Hình 3: Kết quả bài toán lựa chọn đặc trưng có giám sát

5.2 Multi-variable logistic regression

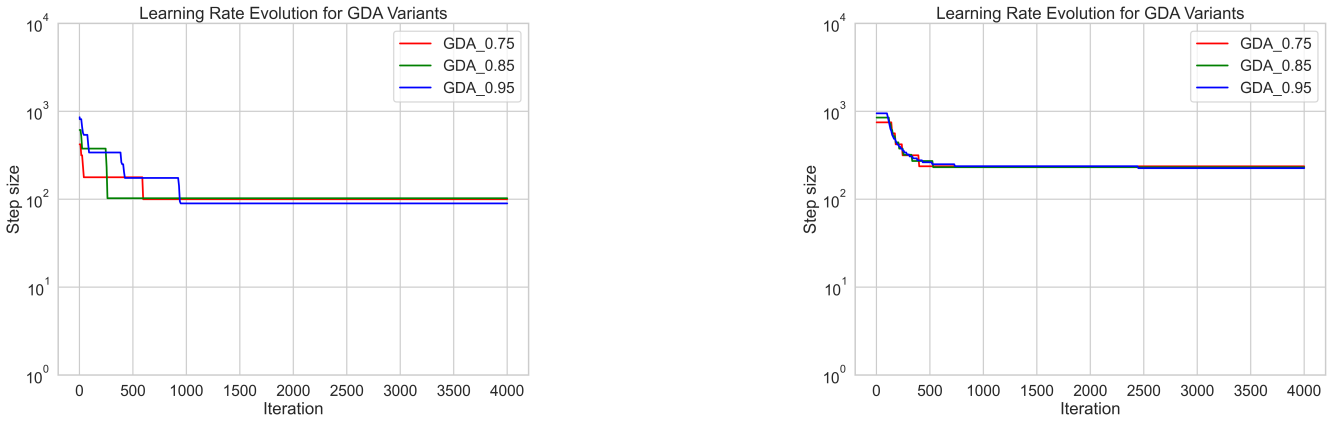
Các thực nghiệm được thực hiện trên bộ dữ liệu bao gồm $\mathbf{N}$ quan sát $(\mathbf{a}_i, b_i) \in \mathbb{R}^d \times \mathbb{R}, i = 1, \ldots, n$. Hàm mất mát entropy chéo cho bài toán hồi quy logistic đa biến được xác định bởi $J(x) = -\sum_{i=1}^{\mathbf{N}} (b_i \log(\sigma(-x^T \mathbf{a}_i)) + (1 - b_i) \log(1 - \sigma(-x^T \mathbf{a}_i)))$, trong đó σ là hàm sigmoid. Kết hợp với điều chuẩn ℓ_2, ta có hàm mất mát được chuẩn hóa $\bar{J}(x) = J(x) + \frac{1}{2\mathbf{N}} \|x\|^2$. Hệ số Lipschitz L được ước lượng bởi $\frac{1}{2\mathbf{N}}(\|A\|^2/2 + 1)$, trong đó $A = (a_1^T, \ldots, a_n^T)^T$. Chúng tôi so sánh các thuật toán huấn luyện cho bài toán hồi quy logistic bằng cách sử dụng các bộ dữ liệu Mushrooms và W8a (xem Malitsky và Mishchenko 2020 [12]). Phương pháp GDA được so sánh với thuật toán GD có kích thước bước nhảy là $1/L$ và phương pháp tăng tốc Nesterov (Nesterov's accelerated method). Các kết quả tính toán được hiển thị tương ứng trong Hình 4 và 5. Các hình này cho thấy thuật toán GDA vượt trội hơn thuật toán GD và phương pháp tăng tốc Nesterov về các giá trị hàm mục tiêu trong quá trình lặp. Cụ thể, Hình 5 thể hiện sự thay đổi của các giá trị hàm mục tiêu theo các hệ số κ khác nhau. Trong hình này, ký hiệu GDA_0.75 tương ứng với $\kappa = 0.75$. Hình 6 trình bày sự giảm kích thước bước từ một kích thước bước ban đầu liên quan đến các kết quả trong Hình 5.



Hình 4: Kết quả so sánh đối với dataset Mushrooms (bên trái) và W8a (bên phải)



Hình 5: Kết quả so sánh đối với dataset Mushrooms (bên trái) và W8a (bên phải)

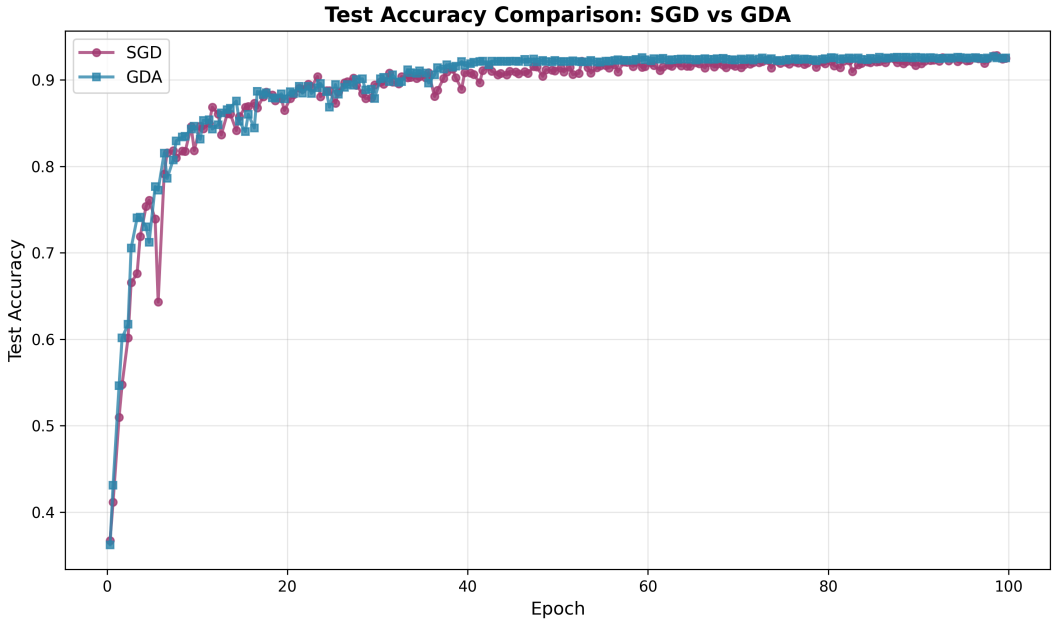


Hình 6: Sự thay đổi kích thước bước đối với dataset Mushrooms (bên trái) và W8a (bên phải)

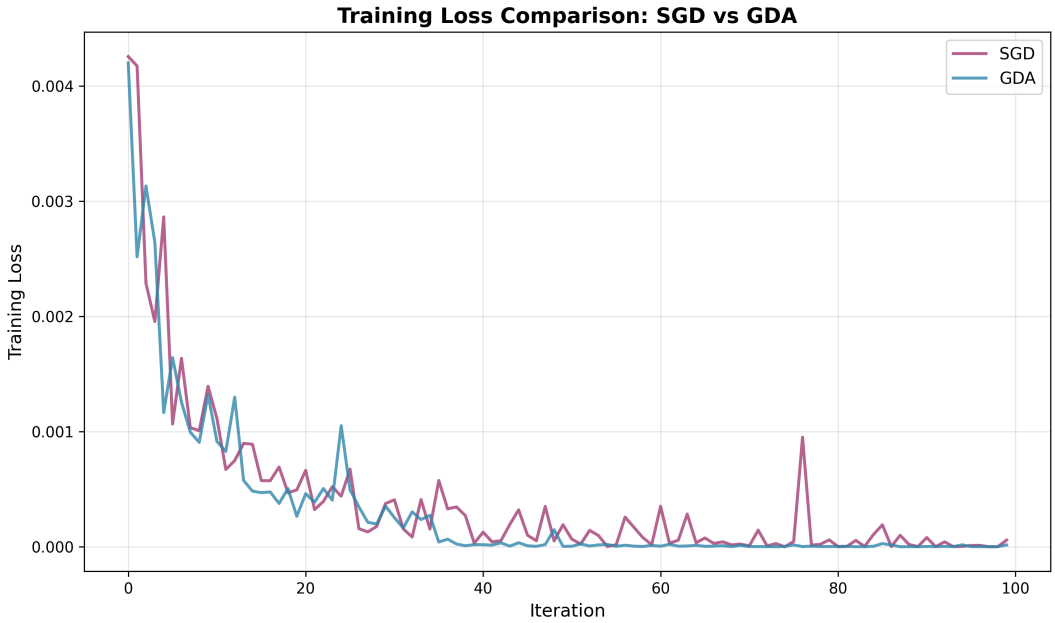
5.3 Neural networks for classification

Để cung cấp một ví dụ về cách thức thuật toán được đề xuất có thể được triển khai vào một mô hình huấn luyện mạng nơ-ron, chúng tôi sẽ sử dụng các kiến trúc ResNet-18 tiêu chuẩn đã được cài đặt trong PyTorch và sẽ huấn luyện chúng để phân loại các hình ảnh được lấy từ bộ dữ liệu Cifar10 (xem https://www.cs.toronto.edu/~kriz/cifar.html), đồng thời tính đến hàm mất mát entropy chéo. Trong các nghiên cứu với ResNet-18, chúng tôi đã sử dụng các thiết lập mặc định của Adam cho các tham số của nó.

Để huấn luyện mô hình mạng nơ-ron này, chúng tôi sử dụng biến thể ngẫu nhiên của phương pháp GDA (thuật toán SGDA) để so sánh với các thuật toán SGD. Các kết quả tính toán được hiển thị trong Hình 7 và 8 cho thấy thuật toán SGDA vượt trội hơn thuật toán SGD về độ chính xác trên tập kiểm tra và độ mất mát huấn luyện qua các vòng lặp.



Hình 7: Độ chính xác trên tập kiểm tra của mô hình ResNet-18 qua các vòng lặp



Hình 8: Giá trị hàm mất mát trên tập huấn luyện của mô hình ResNet-18 qua các vòng lặp

6 Kết luận

Bài báo đã đề xuất một quy trình bước nhảy thích nghi mới và đơn giản, áp dụng cho một lớp rộng các phương pháp giải bài toán tối ưu hóa với hàm mục tiêu không lồi. Cách tiếp cận này không yêu cầu bất kỳ kỹ thuật tìm kiếm theo đường hay kiến thức tiên nghiệm nào, mà thay vào đó xem xét hành vi của dãy lặp. Kết quả là, so với các phương pháp tìm kiếm theo đường xuống thang, phương pháp này giúp giảm đáng kể chi phí tính toán tại mỗi bước lặp. Bài báo đã chứng minh được sự hội tụ của kỹ thuật này dưới các giả thiết đơn giản, đồng thời chỉ ra rằng quy trình mới này tạo nên một nền tảng tổng quát cho các phương pháp tối ưu hóa. Các kết quả thực nghiệm mô phỏng số ban đầu đã chứng minh tính hiệu quả của quy trình mới này. Thuật toán xuống thang với bước nhảy thích nghi này có thể được mở rộng cho lớp bài toán quy hoạch với hàm mục tiêu không trơn, hoặc mở rộng hơn nữa cho các bài toán tối ưu hóa đa mục tiêu dựa trên các nghiên cứu trước đó.

Tài liệu

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[2] W. Bian, L. Ma, S. Qin, and X. Xue, "Neural network for nonsmooth pseudoconvex optimization with general convex constraints," *Neural Networks*, vol. 101, pp. 1–14, 2018.

[3] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Cambridge, 2009.

[4] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data," *Signal Processing Magazine*, vol. 31, pp. 32–4, 2014.

[5] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, New Jersey, 1983.

[6] O. P. Ferreira and W. S. Sosa, "On the Frank-Wolfe algorithm for non-compact constrained optimization problems," *Optimization*, vol. 71, no. 1, pp. 197–211, 2022.

[7] Y. Hu, J. Li, and C. K. Yu, "Convergence rates of subgradient methods for quasiconvex optimization problems," *Computational Optimization and Applications*, vol. 77, pp. 183–212, 2020.

[8] K. C. Kiwiel, "Convergence and efficiency of subgradient methods for quasiconvex minimization," *Mathematical Programming, Series A*, vol. 90, pp. 1–25, 2001.

[9] I. V. Konnov, "Simplified versions of the conditional gradient method," *Optimization*, vol. 67, no. 12, pp. 2275–2290, 2018.

[10] G. H. Lan, *First-order and Stochastic Optimization Methods for Machine Learning*. Springer Series in the Data Sciences, Springer Nature, 2020.

[11] N. Liu, J. Wang, and S. Qin, "A one-layer recurrent neural network for nonsmooth pseudoconvex optimization with quasiconvex inequality and affine equality constraints," *Neural Networks*, vol. 147, pp. 1–9, 2022.

[12] Y. Malitsky and K. Mishchenko, "Adaptive gradient descent without descent," *Proceedings of Machine Learning Research*, vol. 119, pp. 6702–6712, 2020.

[13] O. Mangasarian, "Pseudo-convex functions," *SIAM Control*, vol. 8, pp. 281–290, 1965.

[14] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Springer Science & Business Media, 2013.

[15] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, Princeton, 1970.

[16] T. N. Thang and T. N. Hai, "Self-adaptive algorithms for quasiconvex programming and applications to machine learning", 2024.

[17] Y. Wang, X. Li, and J. Wang, "A neurodynamic optimization approach to supervised feature selection via fractional programming," *Neural Networks*, vol. 136, pp. 194–206, 2021.

[18] H. K. Xu, "Iterative algorithms for nonlinear operators," *Journal of the London Mathematical Society*, vol. 66, pp. 240–256, 2002.

[19] C. K. Yu, Y. Hu, X. Yang, and S. K. Choy, "Abstract convergence theorem for quasi-convex optimization problems with applications," *Optimization*, vol. 68, no. 7, pp. 1289–1304, 2019.