Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

# PHÂN LỚP DỮ LIỆU VỚI GIẢI THUẬT NEWTON SVM

Đỗ Thanh Nghị[1], Nguyễn Minh Trung[2] và Phạm Nguyên Khang[1]
[1] *Khoa Công nghệ Thông tin & Truyền thông, Trường Đại học Cần Thơ*
[2] *Khoa Khoa học Tự nhiên, Trường Đại học Cần Thơ*

***Thông tin chung:***
*Ngày nhận: 19/02/2014*
*Ngày chấp nhận: 30/06/2014*

***Title:***
*Data classification using The Newton Support Vector Machine algorithm*

***Từ khóa:***
*Giải thuật Newton support vector machine, trọng số thích nghi và kết hợp, ARC-x4, phân lớp dữ liệu lớn*

***Keywords:***
*Newton support vector machine algorithm, adaptive reweighting and combining, ARC-x4, classifying large datasets*

**ABSTRACT**

*In this paper, we propose a new machine learning algorithm, called the ARC-x4 of finite Newton Support Vector Machine (NSVM) for classifying very large datasets on standard personal computers (PCs). SVM and kernel related methods have provided accurate classification models but their learning tasks usually need a quadratic programming with the requirement of large memory capacity and long time. We extend the recent NSVM proposed by Mangasarian for building a boosting-SVM algorithm. We have used the Sherman-Morrison-Woodbury formula to adapt the NSVM to process datasets with a very large number of dimensions. We have also applied the ARC-x4 approach proposed by Breiman to NSVM for classifying massive datasets with a very large number of datapoints as well as a very large number of dimensions. We have evaluated its performance on bio-medical datasets with a PC (2.4 GHz Pentium IV, 2 GB RAM).*

**TÓM TẮT**

*Chúng tôi trình bày trong bài viết một giải thuật học mới, ARC-x4 Newton support vector machine (ARC-x4-NSVM), cho phân loại tập dữ liệu lớn trên máy tính cá nhân. Máy học véc-tơ hỗ trợ (SVM) và phương pháp hàm nhân cung cấp mô hình phân lớp dữ liệu chính xác nhưng quá trình huấn luyện mô hình cần giải bài toán quy hoạch toàn phương rất mất thời gian và cần nhiều bộ nhớ. Chúng tôi đề xuất mở rộng giải thuật học NSVM của Mangasarian để xây dựng giải thuật cải tiến SVM. Chúng tôi đề xuất áp dụng công thức Sherman-Morrison-Woodbury vào giải thuật NSVM để có thể xử lý dữ liệu có số chiều rất lớn. Tiếp theo sau, chúng tôi kết hợp với phương pháp ARC-x4 của Breiman để xây dựng giải thuật ARC-x4-NSVM có thể phân loại dữ liệu với kích thước lớn về số phần tử cũng như số chiều. Chúng tôi đánh giá hiệu quả của giải thuật đề xuất trên tập dữ liệu y sinh học sử dụng máy tính cá nhân (2.4 GHz Pentium IV, 2 GB RAM).*

## 1 GIỚI THIỆU

Từ khi được giới thiệu bởi Vapnik [15], giải thuật máy học véc-tơ hỗ trợ (SVM) trở thành phương pháp máy học hữu hiệu để giải quyết các vấn đề phân lớp, hồi quy. Máy học SVM đã được áp dụng thành công trong rất nhiều ứng dụng như nhận dạng mặt người, phân loại văn bản, phân loại bệnh ung thư (tham khảo tại [8]). Bằng việc kết hợp với phương pháp hàm nhân, máy học SVM cung cấp các mô hình hiệu quả chính xác cho các vấn đề phân lớp và hồi quy phi tuyến trong thực tế. Mặc dù có được những ưu điểm kể trên, việc huấn luyện của giải thuật máy học SVM rất mất thời

gian và tiêu tốn nhiều không gian bộ nhớ do phải giải bài toán quy hoạch toàn phương. Độ phức tạp tối thiểu huấn luyện của giải thuật máy học SVM luôn là bậc 2 so với số lượng phần tử dữ liệu. Do đó, cần thiết phải có những cải tiến để giải thuật học SVM có thể xử lý được các tập dữ liệu với kích thước lớn về số phần tử cũng như số chiều.

Để cải tiến việc huấn luyện giải thuật máy học SVM cho các tập dữ liệu lớn. Các công trình nghiên cứu trong [2], [4], [11] đã chia bài toán quy hoạch toàn phương gốc thành các bài toán con để giải quyết. Nghiên cứu của [12], [13] đã đề nghị xây dựng giải thuật học tăng trưởng, chỉ nạp dữ liệu từng phần rồi cập nhật mô hình theo dữ liệu mà không cần nạp toàn bộ tập dữ liệu trong bộ nhớ. Công trình nghiên cứu của [13] đề nghị giải thuật song song trên mạng để cải thiện tốc độ huấn luyện. Tong & Koller [14] đề nghị phương pháp chọn tập con dữ liệu thay vì phải học trên toàn bộ tập dữ liệu gốc.

Trong bài viết này, chúng tôi muốn trình bày một giải thuật học mới, ARC-x4-NSVM, dùng cho phân loại các tập dữ liệu lớn trên máy tính cá nhân. Chúng tôi mở rộng giải thuật học NSVM của Mangasarian [10]. Giải thuật học NSVM chỉ cần giải các hệ phương trình tuyến tính thay vì là bài toán quy hoạch toàn phương phức tạp hơn như giải thuật máy học SVM chuẩn. Chúng tôi đã phát triển theo hướng thích ứng giải thuật NSVM cho phân loại dữ liệu có số chiều lớn thường gặp trong các vấn đề về phân loại văn bản hay trong sinh tin học. Để đạt được mục tiêu này, chúng tôi áp dụng công thức Sherman-Morrison-Woodbury [7] giúp cho giải thuật cải biến của NSVM có thể làm việc cho dữ liệu có số chiều rất lớn nhưng có số phần tử (dòng) nhỏ. Sau đó, chúng tôi kết hợp với giải thuật ARC-x4 của Breiman trong [3] tiếp tục phát triển giải thuật ARC-x4-NSVM có thể phân loại được dữ liệu có kích thước lớn cả về số dòng và số lượng chiều. Chúng tôi cũng tiến hành đánh giá hiệu quả dựa trên tiêu chí như thời gian huấn luyện và độ chính xác của giải thuật ARC-x4-NSVM sử dụng máy tính cá nhân (Pentium 2.4 GHz, 2 GB RAM, Linux). Kết quả chạy thử nghiệm trên các tập dữ liệu lớn y sinh học [9] cho thấy giải thuật ARC-x4-NSVM của chúng tôi đề nghị có thời gian huấn học rất nhanh và cho độ chính xác cao khi so sánh với các giải thuật máy học SVM chuẩn như LibSVM [4].

Phần tiếp theo của bài được tổ chức như sau. Phần 2 sẽ trình bày tóm tắt về giải thuật máy học NSVM. Phần 3 sẽ chỉ ra cách xây dựng giải thuật ARC-x4-NSVM cho phân loại dữ liệu lớn. Kết quả chạy thử nghiệm sẽ được trình bày trong phần 4 trước khi kết thúc bằng kết luận và hướng phát triển.

Chúng tôi sử dụng các ký hiệu và khái niệm trong phần tiếp theo của bài viết như sau: tất cả các véc-tơ đều là véc-tơ cột, tích vô hướng của 2 véc-tơ $x$ và $y$ được ký hiệu là $x.y$, độ dài véc-tơ pháp tuyến của véc-tơ $x$ được ký hiệu là $||x||$, trong khi $x^T$ chính là chuyển vị của $x$ và $e$ là véc-tơ cột mà các thành phần bằng 1.

## 2 GIẢI THUẬT MÁY HỌC NSVM

Xét ví dụ phân lớp nhị phân tuyến tính như Hình 1. Cho $m$ phần tử $x_1, x_2, \ldots, x_m$ trong không gian $n$ chiều, biểu diễn bởi ma trận $A[mxn]$, có nhãn (lớp) của các phần tử là $y_1, y_2, \ldots, y_m$ có giá trị $1$ hoặc giá trị $-1$, biểu diễn bởi ma trận đường chéo $D[mxm]$ của 1, -1. $y_i = 1$, nếu $x_i$ thuộc lớp $+1$ (lớp dương, lớp chúng ta quan tâm), $y_i = -1$, nếu $x_i$ thuộc lớp $-1$ (lớp âm hay các lớp còn lại).

### 2.1 Giải thuật SVM

Giải thuật học SVM của Vapnik [15] tìm siêu phẳng tối ưu (xác định bởi véc-tơ pháp tuyến $w$ và độ lệch của siêu phẳng $b$) dựa trên 2 siêu phẳng hỗ trợ của 2 lớp.

Các phần tử $\boldsymbol{A_i}$ của lớp +1 nằm bên phải của siêu phẳng hỗ trợ cho lớp +1, các phần tử $\boldsymbol{A_j}$ lớp -1 nằm phía bên trái của siêu phẳng hỗ trợ cho lớp -1.

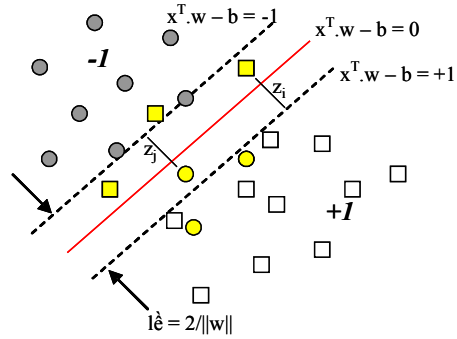
$$A_i .w - b \geq 1, \ \forall i \text{ có } D[i, i] = 1 \quad (1)$$

$$A_j .w - b \leq -1, \ \forall j \text{ có } D[j, j] = -1 \quad (2)$$

Kết hợp (1) và (2) ta được:

$$D(Aw - eb) \geq e \quad (3)$$

trong đó $\boldsymbol{e}$ là véctơ cột mà tất cả các phần tử của nó đều bằng 1.




**Hình 1: Phân lớp tuyến tính với máy học SVM**

Khoảng cách giữa 2 siêu phẳng hỗ trợ gọi là lề (margin) và được tính bằng:

$$margin = \frac{2}{\|w\|} \quad (4)$$

trong đó $\|w\|$ là độ dài của véc-tơ $w$.

Siêu phẳng kết quả ($w$, $b$) phân chia tập các điểm thành 2 lớp nằm ở giữa 2 siêu phẳng hỗ trợ. Bất cứ điểm $x_i$ nào nằm sai phía so với siêu phẳng hỗ trợ của nó được xem là lỗi. Khoảng cách lỗi được biểu diễn bởi $z_i \geq 0$ (với $x_i$ nằm đúng phía của siêu phẳng hỗ trợ của nó thì khoảng cách lỗi tương ứng $z_i = 0$, còn ngược lại thì $z_i > 0$ là khoảng cách từ điểm $\boldsymbol{x_i}$ đến siêu phẳng hỗ trợ tương ứng của nó). Việc tìm kiếm siêu phẳng tối ưu của giải thuật máy học SVM bằng với việc cực đại hóa lề (lề càng lớn, mô hình phân lớp càng an toàn) và cực tiểu hóa lỗi. Giải thuật SVM dẫn đến bài toán quy hoạch toàn phương sau:

$$\min_{z,w,b} f = C\|z\| + \frac{1}{2}\|w\|^2 \quad (5)$$

với ràng buộc D(Aw – eb) + z ≥ e

Trong đó $C > 0$ là hằng số cho phép điều chỉnh mức độ lỗi ($z \geq 0$) và độ rộng (lề) của 2 siêu phẳng hỗ trợ.

Giải bài toán quy hoạch toàn phương (5), chúng ta thu được siêu phẳng ($w$, $b$). Việc phân loại cho phần tử mới dựa trên siêu phẳng kết quả ($w$, $b$) được tính theo công thức sau:

$$predict\ (x) = sign(w.x + b) \quad (6)$$

Giải thuật SVM cơ bản chỉ giải quyết được bài toán phân lớp tuyến tính, tuy nhiên nếu ta kết hợp SVM với phương pháp hàm nhân (kernel-based method) sẽ cho phép giải quyết lớp các bài toán phân lớp phi tuyến. Có thể tham khảo chi tiết hơn trong các tài liệu [1], [5].

Độ phức tạp tính toán của bài toán quy hoạch toàn phương (5) tối thiểu là $O(m^2)$ trong đó $m$ là số lượng phần tử được dùng để huấn luyện. Điều này làm cho giải thuật SVM không phù hợp với dữ liệu lớn.

### 2.2 Giải thuật Newton SVM (NSVM)

Giải thuật NSVM do Mangasarian [10] đề nghị, cải biên bài toán SVM gốc bằng cách:

– Sử dụng hàm lỗi bình phương nhỏ nhất $\frac{C}{2}\|z\|^2$ (thay vì $C\|z\|$);

– Cực đại lề phân hoạch bằng $\frac{1}{2}\|w,b\|^2$ (thay vì $\frac{1}{2}\|w\|^2$).

Giải thuật SVM được viết lại dưới dạng (7):

$$\min_{z,w,b} f = \frac{C}{2}\|z\|^2 + \frac{1}{2}\|w,b\|^2 \quad (7)$$

với ràng buộc D(Aw – eb) + z ≥ e

Trong đó $C > 0$ là hằng số cho phép điều chỉnh mức độ lỗi ($z \geq 0$) và độ rộng (lề) của 2 siêu phẳng hỗ trợ.

Bằng cách thay thế $z = (e – D(Aw – eb))_+$ (với $(x)_+$ thay thế các thành phần âm của véc-tơ $x$ bởi giá trị 0) từ ràng buộc vào hàm mục tiêu $f$ của (7), ta thu được bài toán tối ưu không ràng buộc (8):

$$\min_{w,b} f = \frac{C}{2}\|(e - D(Aw - eb))_+\|^2 + \frac{1}{2}\|w,b\|^2 \quad (8)$$

Bằng cách đặt $u = [w_1\ w_2\ \ldots\ w_n\ b]^T$ và $H = [A\ -e]$, công thức SVM trong (8) được viết lại như (9):

$$\min_{w,b} f = \frac{C}{2}\|(e - DHu)_+\|^2 + \frac{1}{2}u^T u \quad (9)$$

Mangasarian [10] đã đề xuất giải thuật lặp Newton để giải quyết vấn đề tối ưu không ràng buộc như của SVM trong (9). Giải thuật được mô tả như Bảng 1. Mangasarian cũng chứng minh rằng dãy các giá trị $\{u_i\}$ của giải thuật lặp Newton hội tụ đến nghiệm tối ưu toàn cục. Trong hầu hết các trường hợp kiểm thử trong thực tế thì giải thuật lặp Newton hội tụ đến nghiệm trong khoảng từ 5 đến 8 bước lặp. Chú ý rằng, giải thuật lặp NSVM chỉ yêu cầu giải các hệ phương trình tuyến tính (10) có ($n+1$) biến (thay vì là bài toán quy hoạch toàn phương như giải thuật SVM chuẩn). Do đó, nếu số chiều dữ liệu $n$ trong bậc nhỏ hơn $100$ thì thậm chí số phần tử dữ liệu $m$ lên đến hàng triệu, giải thuật lặp NSVM có thể phân lớp chúng trong vài giây trên một máy tính cá nhân.

**Bảng 1: Giải thuật lặp NSVM**

- Đầu vào: tập dữ liệu huấn luyện, *A[mxn]* và *D[mxm]*
- Bắt đầu với $u_0 \in R^{n+1}$ and $i = 0$
- Repeat
  1) $u_{i+1} = u_i - \partial^2 f(u_i)^{-1} \nabla f(u_i)$ (10)
  2) $i = i + 1$

  Until $\nabla f(u_i) = 0$
- Return $u_i$

Với đạo hàm của $f$ tại $u_i$,

$$\nabla f(u_i) = C(-DH)^T(e - DHu_i)_+ + u_i \quad (11)$$

Và ma trận Hesse, đạo hàm từng phần bậc 2 của $f$ tại $u_i$,

$$\partial^2 f(u_i) = C(-DH)^T diag([e - DHu_i]_*)(-DH) + I \quad (12)$$

Với $diag([e - DHu_i]_*)$ là ma trận đường chéo $(n+1)x(n+1)$ mà thành phần chéo thứ $j$ *là* đạo hàm thành phần của hàm $(e - DHu_i)_+$

## 3 GIẢI THUẬT ARC-X4-NSVM CHO PHÂN LỚP TẬP DỮ LIỆU LỚN

### 3.1 Giải thuật NSVM cho dữ liệu có số chiều lớn nhưng ít phần tử

Trong các ứng dụng như sinh tin học hay phân loại văn bản, các tập dữ liệu cần xử lý có số chiều $n$ thường rất lớn (hơn *1000*) nhưng có số phần tử $m$ nhỏ (khoảng *50* đến *250* phần tử). Với những tập dữ liệu dạng này, ma trận vuông $(n+1)x(n+1)$ Hesse, $\partial^2 f(u_i)$ trong (12) có kích thước lớn và việc lấy nghịch đảo ($\partial^2 f(u_i)^{-1}$) hay giải hệ phương trình tuyến tính trong (10) trở nên quá phức tạp, mất thời gian. Để thích ứng giải thuật NSVM cho trường hợp này, chúng tôi áp dụng định lý Sherman-Morrison-Woodbury [7] để tính ma trận nghịch đảo ($\partial^2 f(u_i)^{-1}$) trong công thức (10) như sau.

Đặt $Q = diag(sqrt(C[e - DHu_i]_*))$ và $P = Q(-DH)$, ma trận nghịch đảo ($\partial^2 f(u_i)^{-1}$) có thể được viết lại như công thức (13):

$$\partial^2 f(u_i)^{-1} = (I + P^T P)^{-1} \quad (13)$$

Định lý Sherman-Morrison-Woodbury được cho như (14):

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1} \quad (14)$$

Tiếp đến, áp dụng định lý Sherman-Morrison-Woodbury (14) vào vế phải của (13), chúng ta có được ma trận nghịch đảo ($\partial^2 f(u_i)^{-1}$) như (15):

$$\partial^2 f(u_i)^{-1} = (I + P^T P)^{-1} = I - P^T(I + PP^T)^{-1} P \quad (15)$$

Chúng ta có thể thấy rằng việc tính ma trận nghịch đảo ($\partial^2 f(u_i)^{-1}$) như trong công thức (15) chỉ phụ thuộc vào nghịch đảo ma trận cấp $(m)x(m)$ là $(I + PP^T)$ thay vì là phải nghịch đảo ma trận cấp $(n+1)x(n+1)$ như trong (10). Vì thế, độ phức tạp tính toán của NSVM lúc này chỉ phụ thuộc vào số lượng các phần tử chứ không phải số chiều. Cách biến đổi lại bài toán như thế này có thể xử lý được các tập dữ liệu có số chiều rất lớn nhưng ít phần tử.

### 3.2 Giải thuật ARC-x4-NSVM cho dữ liệu có số phần tử và số chiều lớn

Đối với việc phân lớp các tập dữ liệu vừa có số chiều lớn đồng thời số lượng phần tử cũng rất lớn (trên $10^4$), ví dụ như trong phân loại văn bản, sẽ có ít nhất 2 vấn đề cần giải quyết: thời gian huấn luyện tăng lên tỉ lệ thuận với số phần tử dùng để huấn luyện và bộ nhớ dùng để lưu trữ dữ liệu cũng phải tăng lên.

Để giải quyết bài toán trong trường hợp dữ liệu có số chiều và số phần tử đều lớn, chúng tôi áp dụng cách tiếp cận boosting như Adaboost của Freund & Schapire [6], ARC-x4 của Breiman [3] lên NSVM. Việc làm này mang lại 2 lợi ích:

(i) xử lý được dữ liệu lớn và

(ii) cải thiện độ chính xác.

Giải thuật Adaboost là một phương pháp tổng quát để cải tiến độ chính xác của các giải thuật phân lớp yếu. Giải thuật lặp lại việc huấn luyện bằng các giải thuật phân lớp yếu $t$ lần, mỗi lần trên một tập con được lấy mẫu từ tập dữ liệu dùng để huấn luyện. Lần lặp sau tập trung huấn luyện trên các phần tử bị phân loại sai trong lần lặp trước đó. Để làm được điều này, mỗi phần tử được gán một trọng số.

Khởi tạo các trọng số này bằng nhau. Sau mỗi bước lặp các phần tử bị phân lớp sai sẽ được tăng trọng số lên. Sau quá trình lặp, ta có *t* bộ phân lớp yếu. Việc phân loại phần tử mới sử dụng kết quả bình chọn trọng số từ *t* bộ phân lớp yếu. ARC-x4 cũng tương tự như Adaboost ngoại trừ ARC-x4 tập trung huấn luyện trên các phần tử bị phân loại sai trong tất cả các lần lặp trước đó và việc phân loại phần tử mới dựa trên kết quả bình chọn số đông (không có trọng số).

Chúng ta có thể xem NSVM như một bộ phân lớp yếu vì việc huấn luyện trong từng bước lặp chỉ thực hiện trên tập con của tập dữ liệu huấn luyện. Và như thế ta có thể áp dụng Adaboost hay ARC-x4 lên NSVM.

Với tập dữ liệu có số phần tử lớn, số chiều nhỏ hơn *100*, thì sử dụng NSVM như trong (10). Với dữ liệu có số chiều lớn hay đồng thời lớn về số chiều và số phần tử, boosting của NSVM vẫn có thể giải quyết được nhanh chóng vì mặc dù số chiều lớn nhưng trong mỗi lần lặp số lượng phần tử để huấn luyện sẽ nhỏ nên có thể áp dụng (15) để giải.

Chú ý rằng thời gian huấn luyện NSVM nhanh hơn rất nhiều so với SVM chuẩn trên các tập con (ít hơn rất nhiều so với toàn bộ tập huấn luyện). Vì vậy, chúng tôi quan tâm boosting của NSVM hơn là SVM chuẩn.

Chúng tôi đã xây dựng giải thuật ARC-x4-NSVM để giải quyết bài toán dữ liệu lớn. Theo Breiman công bố trong công trình [3], thực nghiệm cho thấy ARC-x4 cho kết quả tương đương với Adaboost. Tuy nhiên về mặt tính toán so với Adaboost thì ARC-x4 đơn giản hơn nhiều. Đây là cũng là lý do chúng tôi chọn ARC-x4 thay vì Adaboost.

## 4 KẾT QUẢ THỰC NGHIỆM

Để tiến hành đánh giá hiệu quả của giải thuật ARC-x4-NSVM cho phân lớp các tập dữ liệu lớn, chúng tôi đã cài đặt giải thuật bằng ngôn ngữ lập trình C/C++. Ngoài ra, chúng tôi cũng cần so sánh hiệu quả về thời gian và độ chính xác phân lớp của giải thuật đề xuất ARC-x4-NSVM với một giải thuật SVM chuẩn, được sử dụng phổ biến trong cộng đồng máy học là LibSVM [4]. Tất cả các giải thuật đều được thực hiện trên một máy tính cá nhân (2.4 GHz Pentium IV, 2GB RAM) chạy hệ điều hành Linux.

**Bảng 2: Mô tả các tập dữ liệu y sinh học**

| Tập dữ liệu | Số lớp | Số phần tử | Số chiều | Nghi thức kiểm tra |
|---|---|---|---|---|
| ALL-AML Leukemia | 2 | 72 | 7129 | 38 trn - 34 tst |
| Breast Cancer | 2 | 97 | 24481 | 78 trn - 19 tst |
| Ovarian Cancer | 2 | 253 | 15154 | leave-1-out |
| Lung Cancer | 2 | 181 | 12533 | 32 trn - 149 tst |
| Translation Initiation Sites | 2 | 13375 | 927 | hold-out |

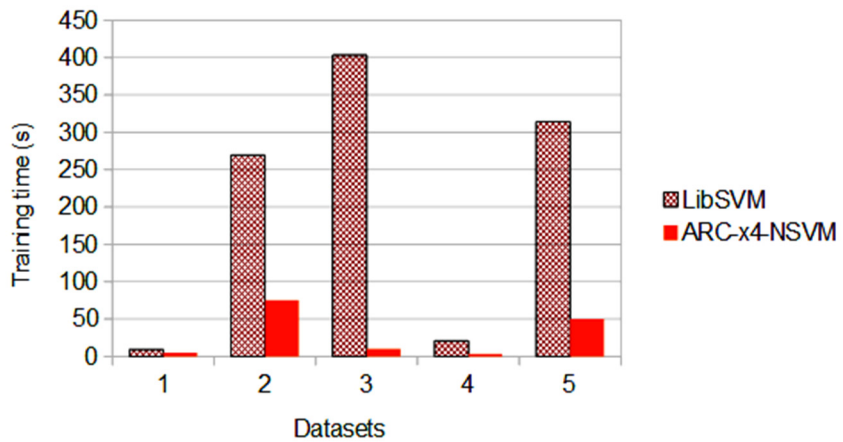
Các tập dữ liệu thực nghiệm được lấy về từ website của Jinyan & Huiqing [9]. Đây là các tập dữ liệu y sinh học, có số chiều lớn, bậc hàng ngàn và số phần tử từ vài chục đến hàng chục ngàn. Xét tập dữ liệu như Ovarian Cancer, với 253 phần tử, 15154 chiều. Nếu sử dụng giải thuật gốc NSVM của Mangasarian [10], mỗi bước lặp phải lấy nghịch đảo ma trận kích thước 15155x15155 ($(n+1)x(n+1)$), độ phức tạp tương ứng $15155^3$. Trong khi giải thuật cải tiến của chúng tôi đề xuất ở mỗi bước lặp thực hiện nghịch đảo ma trận 253x253 (mxm) với độ phức tạp tương ứng là $253^3$. Giảm độ phức tạp khoảng 210000 lần. Thậm chí với một máy tính PC, rất khó thực hiện việc lấy nghịch đảo ma trận kích thước 15155x15155. Qua đây có thể thấy rằng giải thuật cải tiến hiệu quả so với giải thuật gốc trong trường hợp dữ liệu y sinh học.

Bảng 2 mô tả các tập dữ liệu thực nghiệm. Cột cuối của bảng 2 mô tả nghi thức kiểm tra của các giải thuật. Ba tập dữ liệu ALL-AML Leukemia, Breast Cancer, Lung Cancer đều được chia thành tập huấn luyện (trn) và tập kiểm tra (tst). Đối với các tập dữ liệu này, chúng ta cần huấn luyện mô hình học sử dụng tập huấn luyện, tính thời gian huấn luyện và dùng tập kiểm tra để tính độ chính xác cho phân lớp. Tập Ovarian Cancer thì sử dụng nghi thức leave-1-out, lặp lại 253 lần huấn luyện và kiểm tra, mỗi lần chỉ lấy 1 phần tử làm tập kiểm tra độ chính xác và 252 phần tử còn lại làm tập huấn luyện để tính thời gian huấn luyện, cuối cùng tính trung bình thời gian huấn luyện và độ chính xác. Tập Translation Initiation Sites thì sử dụng nghi thức hold-out, lấy ngẫu nhiên 2/3 tập dữ liệu gốc làm tập huấn luyện để tính thời gian huấn luyện mô hình học và 1/3 còn lại làm tập kiểm tra để tính độ chính xác khi phân lớp.

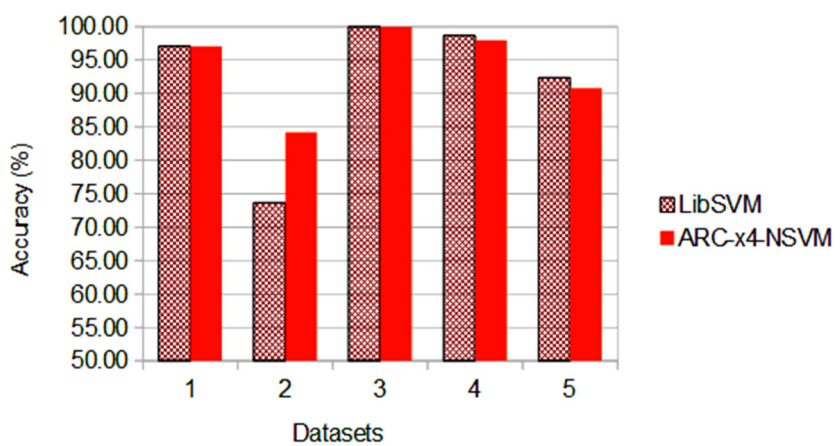
**Bảng 3: Kết quả phân lớp dữ liệu y sinh học**

| Tập dữ liệu | LibSVM | | ARC-x4-NSVM | |
|---|---|---|---|---|
| | Thời gian huấn luyện (giây) | Độ chính xác phân lớp (%) | Thời gian huấn luyện (giây) | Độ chính xác phân lớp (%) |
| ALL-AML Leukemia | 9,14 | **97,06** | **5,01** | **97,06** |
| Breast Cancer | 269,66 | 73,68 | **75,43** | **84,21** |
| Ovarian Cancer | 403,60 | **100** | **10,13** | **100** |
| Lung Cancer | 20,80 | **98,66** | **3,51** | 98,00 |
| Translation Initiation Sites | 314,00 | **92,41** | **50,27** | 90,80 |

Giải thuật ARC-x4-NSVM xây dựng 30 mô hình NSVM, mỗi mô hình được xây dựng trên mẫu 30% tập dữ liệu gốc. Cả LibSVM và NSVM đều sử dụng hàm nhân tuyến tính với hằng số *C = 1000* (điều chỉnh độ rộng lề phân hoạch và lỗi) cho kết quả tốt nhất. Kết quả thu được như trình bày trong Bảng 3. Hình 2, 3 cung cấp đồ thị so sánh về thời gian huấn luyện và độ chính xác phân lớp.




**Hình 2: So sánh thời gian huấn luyện**




**Hình 3: So sánh độ chính xác phân lớp**

So sánh kết quả cho thấy được giải thuật ARC-x4-NSVM thì nhanh hơn LibSVM về thời gian huấn luyện (từ 2 đến 40 lần). Độ chính xác khi phân lớp của ARC-x4-NSVM có thể xem là tương đương với LibSVM.

## 5 KẾT LUẬN VÀ ĐỀ XUẤT

Chúng tôi vừa trình bày giải thuật máy học mới ARC-x4-NSVM cho phân lớp tập dữ liệu lớn trên máy tính cá nhân. Chúng tôi đề xuất mở rộng giải thuật NSVM của Mangasarian để xây dựng giải thuật ARC-x4-NSVM. Chúng tôi đã áp dụng định lý Sherman-Morrison-Woodbury để thích ứng giải thuật NSVM khi phân lớp dữ liệu có số chiều lớn nhưng số phần tử nhỏ, thường gặp trong ứng dụng sinh tin học, phân lớp văn bản. Đề xuất áp dụng ARC-x4 lên NSVM cho phép giải thuật mới có thể phân lớp nhanh dữ liệu có đồng thời số phần tử và số chiều cùng lớn trên máy tính cá nhân. Kết quả thực nghiệm trên các tập dữ liệu y sinh học cho thấy rằng giải thuật ARC-x4-NSVM học nhanh, phân lớp chính xác khi so sánh với giải thuật LibSVM.

Trong tương lai gần, chúng tôi phát triển giải thuật song song cho phép tăng tốc quá trình huấn luyện và phân lớp của giải thuật SVM.

## TÀI LIỆU THAM KHẢO


1. K. Bennett and C. Campbell. Support vector ma chines: Hype or hallelujah?. *SIGKDD Explorations*, 2(2): 1-13, 2000.
2. B. Boser, I. Guyon, and V. Vapnik. An training algorithm for optimal margin classifiers. ACM Annual Workshop on Computational Learning Theory, pages 144-152, 1992.
3. L. Breiman. Arcing classifiers. *The annals of statistics*, 26(3): 801–849, 1998.
4. C-C. Chang and C-J. Lin. Libsvm - a library for support vector machines. 2001-2014.
5. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
6. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. in EuroCOLT, 1995, pp. 23–37.
7. G. Golub and C. van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland, 1996.
8. I. Guyon. Web page on svm applications. 1999-2014.
9. L. Jinyan and L. Huiqing. Kent ridge bio-medical dataset repository. 2002.
10. O. Mangasarian. A finite newton method for classification problems. Data Mining Institute Technical Report 01-11, Computer Sciences Department, University of Wisconsin, 2001.
11. J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, pages 185-208, 1999.
12. F. Poulet and T-N. Do. Mining very large datasets with support vector machine algorithms. *Enterprise Information Systems V*, pages 177-184, 2004.
13. Liu H. Syed, N. and K. Sung. Incremental learning with support vector machines. ACM SIGKDD, 1999.
14. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. ICML, pages 999-1006, 2000.
15. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.