**Tối ưu hóa tối thiểu tuần tự:**

**Thuật toán nhanh để đào tạo hỗ trợ máy vectơ**

John C. Platt

Nghiên cứu của Microsoft jplatt@microsoft.com

Báo cáo kỹ thuật MSR-TR-98-14

21 tháng 4 năm 1998

© 1998 John Platt

# trừu tượng

Bài viết này đề xuất một thuật toán mới để đào tạo các máy vectơ hỗ *trợ: Tối ưu hóa tối thiểu tuần* tự *hoặc SMO*. Đào tạo một máy vectơ hỗ trợ đòi hỏi giải pháp của một vấn đề tối ưu hóa lập trình bậc hai (QP) rất lớn. SMO phá vỡ vấn đề QP large này thành một loạt các vấn đề QP nhỏ nhất có thể. Những vấn đề QP nhỏ này được giải quyết một cách phân tích, tránh sử dụng tối ưu hóa QP số tốn thời gian làm vòng lặp bên trong. Dung lượng bộ nhớ cần thiết cho SMO là tuyến tính trong kích thước thiết lập training, cho phép SMO xử lý các bộ đào tạo rất lớn. Bởi vì tính toán ma trận được tránh, SMO vảy ở đâu đó giữa tuyến tính và bậc hai trong kích thước tập huấn luyện cho các vấn đề kiểm tra khác nhau, trong khi thuật toán SVM chunking tiêu chuẩn mở rộng ở đâu đó giữa tuyến tính và khối trong kích thước tập đào tạo. Thời gian tính toán của SMO bị chi phối bởi đánh giá SVM, do đó SMO nhanh nhất đối với SVM tuyến tính và bộ dữ liệu thưa thớt. Trên các bộ dữ liệu thưa thớt trong thế giới thực, SMO có thể nhanh hơn 1000 lần so với thuật toán chunking.

# 1. GIỚI THIỆU

Trong vài năm qua, đã có sự quan tâm tăng vọt đối với Máy vectơ hỗ trợ (SVM) [19] [20] [4]. SVM đã được chứng minh theo kinh nghiệm để cung cấp hiệu suất khái quát hóa tốt về một loạt các vấn đề như nhận dạng ký tự viết và viết h[12], phát hiện khuôn mặt [15], phát hiện người đi bộ [14] và phân loại văn bản [9].

Tuy nhiên, việc sử dụng SVM vẫn chỉ giới hạn ở một nhóm nhỏ các nhà nghiên cứu. Một lý do có thể là các thuật toán đào tạo cho SVM là slow, đặc biệt là đối với các vấn đề lớn. Một lời giải thích khác là các thuật toán đào tạo SVM rất phức tạp, tinh tế và khó thực hiện đối với một kỹ sư trung bình.

Bài viết này mô tả một thuật toán học tập SVM mới đơn giản về mặt khái niệm, dễ thực hiện , thường nhanh hơn và có đặc tính mở rộng tốt hơn chocác vấn đề SVM khó khăn so với thuật toán đào tạo SVM tiêu chuẩn. Thuật toán học SVM mới được gọi là Tối ưu hóa tối thiểu tuần *tự* (hoặc *SMO*). Thay vì các thuật toán học SVM trước đây sửdụng lập trình bậc hai số (QP) làm vòng lặp bên trong, SMO sử dụng bước QP phân tích.

Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM. Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM. Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM. Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM. Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM. Bài viết trước tiên cung cung cấp một cái cái tổng quan về SVM và giá cả các cuộc toán đào tạo SVM.

Thuật toán SMO sau đó được trình bày chi tiết, bao gồm giải pháp cho bước QP phân tích, heuristics để chọn biến nào sẽ tối ưu hóa trong l oop bên trong, mô tả cách đặt ngưỡngcủa SVM, một số tối ưu hóa cho các trường hợp đặc biệt, mã giả của thuật toán và mối quan hệ của SMO với các thuật toán khác.

SMO đã được thử nghiệm trên hai bộ dữ liệu trong thế giới thực và hai bộ dữ liệu nhân tạo. Bài viết này trình bày kết quả cho SMO thời gian so với thuật toán "chunking" tiêu chuẩn cho các tập dữ liệu này và trình bày kết luận dựa trên các thời điểm này. Cuối cùng, có một phụ lục mô tả nguồn gốc của tối ưu hóa phân tích.

## 1.1 Tổng quan về máy vectơ hỗ trợ

Ví dụ dụ dụ tích cực

Ví dụ như tiêu cực

Tối đa hóa

Điểm

Không gian đầu tư có thể

**Biểu đồ 1**

Một máy tính vector hỗ trợ tuyến đường

Vladimir Vapnik đã phát minh ra Máy vector hỗ trợ vào năm 1979 [19]. Ở dạng tuyến tính đơn giản nhất, SVM là một siêu hành tinh tách một tập hợp các ví dụ tích cực khỏi một tập hợp các ví dụ tiêu cực với lề tối đa (xem hình 1). Trong trường hợp tuyến tính, margin được xác định bởi khoảng cách của siêu hành tinh đến gần nhất của các ví dụ tích cực và tiêu cực. Công thức cho đầu ra của SVM tuyến tính là

r r

### u = w x⋅ −b, (1)

trong *đó w* là vectơ bình thường đến siêu hành tinh và *x* là vectơ đầu vào. Siêu hành tinh tách biệt là máy bay u *=*0. Các điểm gần nhất nằm trên các mặt phẳng *u*=±1. Do đó, *lề m* là như vậy

1

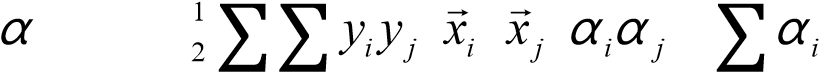
|  |  |
| --- | --- |
| *m* = .  || || *w*  2  Tối đa hóa lề có thể được thể hiện thông qua vấn đề tối ưu hóa sau [4]: | (2) |
| r r r phút || || 1  *w*  2  tùy thuộc vào *y w xi*  ( ⋅  *i*  −*b*) ≥1,∀*i*, | (3) |

*w b*r, 2

trong *đó xi* là ví dụ đào tạo *thứ i,*và *yi* là đầu ra chính xác của SVM cho ví dụ đào tạo *thứ*i. Giá trị *y*i*là* +1 cho các ví dụ tích cực trong một lớp và -1 cho các ví dụ tiêu cực.

Sử dụng một Lagrangian, vấn đề tối ưu hóa này có thể được chuyển đổi thành một hình thức kép đó là một vấn đề QP wở đây hàm mục tiêu Ψ chỉ phụ thuộc vào một tập hợp các hệ số nhân Lagrange α*i*,

|  |  |
| --- | --- |
| *i*=1 *j*=1 *i*=1  (trong đó *N* là số lượng ví dụ đào tạo), chịu những ràng buộc bất bình đẳng, |  |
| α i ≥ 0,∀*i*,  và một ràng buộc bình đẳng tuyến tính, | (5) |
| *N. yi*α*I.* = 0.  ∑ | (6) |

 r *N N N*

minr Ψ( ) = minr  ( ⋅ ) − , (4) α α

*i*=1

Có mối quan hệ một-một giữa mỗi hệ số nhân Lagrange và mỗi ví dụ đào tạo. r

Một khi hệ số nhân Lagrange được xác định, vectơ bình *thường w* và ngưỡng *b có* thể được

|  |  |  |
| --- | --- | --- |
| có nguồn gốc từ hệ số nhân Lagrange: |  |  |
| r *N* r  *w* = ∑ *y*i*αi*  *xi*  , | r r *b*  = *w x*⋅  *k*  −  *yk cho*  một số αk  > 0. | (7) |

*i*=1

R.

Bởi *vì w* có thể được tính toán thông qua phương trình (7) từ dữ liệu đào tạo trước khi sử dụng, lượng tính toán cần thiết để đánh giá SVM tuyến tính là không đổi về số lượng vectơ hỗ trợ không bằng không.

Tất nhiên, không phải tất cả các tập dữ liệu đều có thể tách rời tuyến tính. Có thể không có hyperplane nào tách các ví dụ tích cực khỏi các ví dụ tiêu cực. Trong công thức trên, trường hợp không thể tách rời sẽ tương ứng với một giải pháp vô hạn. Tuy nhiên, vào năm 1995, Cortes &Vapnik [7] đã đề xuất một sửa đổi đối với tuyên bố tối ưu hóa ban đầu (3) cho phép, nhưng phạt, sự thất bại của một ví dụ để đạt được lề chính xác. Sửa đổi đó là:

r *N* r r

1 2

phút || || r r 2  *w*  +*C*∑ξi theo *y w xi*  ( ⋅  *i*  −*b*) ≥ −1 ξ*i*  ,∀*i*, (8)

*w b*, ,ξ *i*=1

trong đó ξ*i* là các biến slack cho phép lỗi ký quỹ và *C là* một tham số giao dịch với ký quỹ rộng với một số lỗi ký quỹ nhỏ. Khi vấn đề tối ưu hóa mới này được chuyển thành dạng kép, nó chỉ đơn giản là thay đổi ràng buộc (5) thành ràngbuộc box:

0 ≤ ≤ ∀α*i C*, *i*. (9)

Các biến ξ*tôi* hoàn toàn không xuất hiện trong công thức kép.

SVM thậm chí có thể được khái quát hóa hơn nữa đến các phân loại phi tuyến tính [2]. Đầu ra của một SVM phi tuyến tính được tính toán rõ ràng từ hệ số nhân Lagrange:

*N* r r

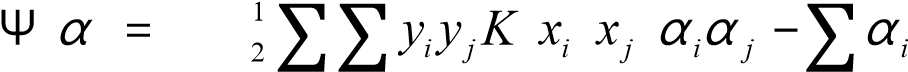
### u = ∑ y jαj K x x( j , ) −b, (10)

*k*=1

trong đó r *K* là một hàm hạt nhân đo lường sự giống nhau hoặc khoảng cách giữa vectơ đầu vàor *x và*  vectơ đào tạo được lưu trữ *x*  *j*  . Ví dụ *về K* bao gồm Gaussians, polynomials, và neural

mạng phi tuyến tính [4]. Nếu *K* là tuyến tính, thì phương trình cho SVM tuyến tính (1) được phục hồi.

Hệ số nhân Lagrange α*tôi vẫn* được tính toán thông qua một chương trình bậc hai. Các phi tuyến tính làm thay đổi dạng bậc hai, nhưng hàm mục tiêu kép Ψ vẫn là bậc hai trong α:

 r *N N* r r *N*

minr ( ) minr  ( , ), α α

*i*=1 *j*=1 *i*=1

0 ≤ ≤ ∀α*i C*, *i*, (11)

*N.*

∑ y *i*α*i*  = 0.

*i*=1

Bài toán QP trong phương trình (11), ở trên, là vấn đề QP mà thuật toán SMO sẽ giải quyết. Để làm cho vấn đề QP ở trên được xác định dương, hàm hạt nhân *K phải* tuân theo các điều kiện của Mercer [4].

Các điều kiện Karush-Kuhn-Tucker (KKT) là cần thiết và đủ conditions cho một điểm tối ưu của một vấn đề QP xác định tích cực. Các điều kiện KKT cho vấn đề QP (11) đặc biệt đơn giản. Vấn đề QP được giải quyết khi, cho tất cả *i*:

α*i*  = 0 ⇔ y *ui i ≥*  1,

0 <α*i*  < *C*  ⇔ y *ui i*  = 1, (12) α*i*  = *C*  ⇔ y *u*i*i ≤*  1.

trong *đó ui* là đầu ra của SVM cho ví dụ đào tạo *thứ*i. Lưu ý rằng các điều kiện KKT có thể được đánh giá trên một ví dụ tại một thời điểm, điều này sẽ hữu ích trong việc xây dựng thuật toán SMO.

## 1.2 Các phương pháp trước đây để đào tạo máy vectơ hỗ trợ

Do kích thước khổng lồ của nó, vấn đề QP (11) phát sinh từ SVM không thể dễ dàng giải quyết thông qua các kỹ thuật QP tiêu chuẩn. Dạng bậc hai trong (11) liên quan đến một ma trận có một số phần tử bằng bình phương của số lượng ví dụ đào tạo. M atrixnày không thể phù hợp với 128 Megabyte nếu có hơn 4000 ví dụ đào tạo.

Vapnik [19] mô tả một phương pháp để giải quyết SVM QP, kể từ đó được gọi là

"chunking." Thuật toán chunking sử dụng thực tế là giá trị của dạng bậc hai giốngnhau nếu bạn loại bỏ các hàng và cột của ma trận tương ứng với hệ số nhân Lagrange bằng không. Do đó, vấn đề QP lớn có thể được chia thành một loạt các vấn đề QP nhỏ hơn, với mục tiêu cuối cùng là xác định tất cả các hệ số nhân Lagrange không bằng không và loại bỏ tất cả các hệ số nhân Lagrange bằng không. Ở mỗi bước, chunking giải quyết một vấn đề QP bao gồm các ví dụ sau: mỗi hệ số nhân Lagrange không bằng không từ bước cuối cùng và *các ví dụ tồi tệ* nhất M vi phạm các ditions KKTcon (12) [4], cho một số giá trị của *M*  (xem hình 2). Nếu có ít hơn các ví dụ *M* vi phạm các điều kiện KKT ở một bước, tất cả các ví dụ vi phạm sẽ được thêm vào. Mỗi vấn đề phụ QP được khởi tạo với kết quả của vấn đề phụ trước đó. Ở bước cuối cùng, toàn bộ hệ số nhân Lagrange không bằng không đã được xác định, do đó bước cuối cùng giải quyết vấn đề QP lớn.

Chunking làm giảm nghiêm trọng kích thước của ma trận từ số lượng ví dụ đào tạo bình phương đến xấp xỉ number của hệ số nhân Lagrange không bằng không bình phương. Tuy nhiên, chunking vẫn không thể xử lý các vấn đề đào tạo quy mô lớn, vì ngay cả ma trận giảm này cũng không thể phù hợp với bộ nhớ.

Phân đoạn

Động vật hải nước

Smo

**Biểu đồ 2.** Ba phương pháp thay thế để đào tạo SVM: Chunking, thuật toán osuna và SMO. Đối với mỗi phương pháp, ba bước được minh họa. Đường mỏng ngang ở mỗi bước đại diện cho bộ đào tạo, trong khi các hộp dày đại diện cho hệ số nhân Lagrangeđược tối ưu hóa ở bước đó. Để phân đoạn, một số ví dụ cố định được thêm vào mỗi bước, trong khi hệ số nhân Lagrange bằng không bị loại bỏ ở mỗi bước. Do đó, số lượng các ví dụ được đào tạo mỗi bước có xu hướng tăng lên. Đối với thuật toán của Osuna, một số cố định of ví dụ được tối ưu hóa mỗi bước: cùng một số ví dụ được thêm vào và loại bỏ khỏi vấn đề ở mỗibước. Đối với SMO, chỉ có hai ví dụ được tối ưu hóa phân tích ở mỗi bước, để mỗi bước đều rất nhanh.

Năm 1997, Osuna, et al. [16]đã chứng minh một định lý cho thấy một bộ thuật toán QP hoàn toàn mới cho SVM. Định lý chứng minh rằng vấn đề QP lớn có thể được chia thành một loạt các vấn đề phụ QP nhỏ hơn. Miễn là ít nhất một ví dụ vi phạm các điều kiện KKT được thêm vào các ví dụ cho vấn đề phụ trước đó, mỗi bước sẽ làm giảm chức năng mục tiêu tổng thể và duy trì một điểm khả thi tuân theo tất cả các ràng buộc. Do đó, một chuỗi các vấn đề phụ QP luôn thêm ít nhất một người vi phạm sẽ được bảo lãnhđể hộitụ. Lưu ý rằng thuật toán chunking tuân theo các điều kiện của định lý, và do đó sẽ hội tụ.

Osuna, et al. đề nghị giữ một ma trận kích thước không đổi cho mọi vấn đề phụ QP, ngụ ý thêm và xóa cùng một số ví dụ ở mỗi bước [16] (xem hình 2). Sử dụng ma trận kích thước không đổi sẽ cho phép đào tạo trên các tập dữ liệu có kích thước tùy ý. Thuật toán được đưa ra trong bài báo của Osuna [16] gợi ý thêm một ví dụ và trừ một ví dụ mỗi bước. Rõ ràng điều này sẽ có hiệuquả, bởi vì nó sẽ sử dụng toàn bộ bước tối ưu hóa QP số để gây ra một ví dụ đào tạo để tuân theo các điều kiện KKT. Trong thực tế, các nhà nghiên cứu cộng và trừ nhiều ví dụ theo heuristics chưa được công bố [17]. Trong mọi trường hợp, cần có bộgiải QP hình chữ số cho tất cả các phương pháp này. QP số nổi tiếng là khó khăn để có được đúng; có nhiều vấn đề chính xác về số cần được giải quyết.

# 2. TỐI ƯU HÓA TỐI THIỂU TUẦN TỰ

Tối ưu hóa tối thiểu tuần tự (SMO) là một thuật toán đơn giản có thể nhanh chóng giải quyết vấn đề SVM QP mà không cần thêm bất kỳ bộ nhớ ma trận nào và không cần sử dụng các bước tối ưu hóa QP số. SMO phân hủy vấn đề QP tổng thể thành các vấn đề phụ QP,chúng tôi ing định lý Osuna để đảm bảo hội tụ.

Không giống như các phương pháp trước, SMO chọn giải quyết vấn đề tối ưu hóa nhỏ nhất có thể ở mỗi bước. Đối với vấn đề TIÊU CHUẨN SVM QP, vấn đề tối ưu hóa nhỏ nhất có thể liên quan đến hai liers multip Lagrange,bởi vì hệ số nhân Lagrange phải tuân theo một ràng buộc bình đẳng tuyến tính. Ở mỗi bước, SMO chọn hai hệ số nhân Lagrange để cùng tối ưu hóa, tìm các giá trị tối ưu cho các hệ số nhân này và cập nhật SVM để phản ánh các giá trị tối ưu mới (xem hình 2).

Ưu điểm của SMO nằm ở chỗ việc giải quyết hai hệ số Nhân Lagrange có thể được thực hiện một cách phân tích. Do đó, tối ưu hóa QP số được tránh hoàn toàn. Vòng lặp bên trong của thuật toán có thể được thể hiện trong một lượng ngắn mã C, rather hơn là gọi toàn bộ thói quen thư viện QP. Mặc dù nhiều vấn đề phụ tối ưu hóa hơn được giải quyết trong quá trình thuật toán, mỗi vấn đề phụ nhanh đến mức vấn đề QP tổng thể được giải quyết nhanh chóng.

Ngoài ra, SMO không yêu cầu thêm ma trận stocơn thịnh nộ ở tất cả. Do đó, các vấn đề đào tạo SVM rất lớn có thể phù hợp bên trong bộ nhớ của một máy tính cá nhân hoặc máy trạm thông thường. Bởi vì không có thuật toán ma trận nào được sử dụng trong SMO, nó ít nhạy cảm với các vấn đề về độ chính xác số.

Có haicomponen ts để SMO: một phương pháp phân tích để giải quyết cho hai hệ số Nhân Lagrange và một heuristic để chọn hệ số nhân để tối ưu hóa.

A..

2

=

*C..*

A..

2

=

*C..*

A..

2

0

=

A..

1

0

=

A..

1

=

*C..*

*K..*

*y y*

2

1

2

1

¡ Æ - =

a A.

A..

2

0

=

A..

1

0

=

A..

1

=

*C..*

*y y*

*K..*

1

2

1

2

=

Æ

+ =

a A.

**Biểu đồ 1.** Hai hệ số lagrange phải đáp ứng tất cả các ràng buộc của vấn đề đầy đủ. Những ràng buộc bất bình đẳng khiến hệ số nhân Lagrange nằm trong hộp. Ràng buộc bình đẳng tuyến tính khiến chúng nằm trên một đường chéo. Do đó, một bước của SMO phải tìm tối ưu chức năng mục tiêu trên phân đoạn đường chéo.

## 2.1 Giải cho hai hệ số nhân Lagrange

Để giải quyết cho hai hệ số nhân Lagrange, SMO trước tiên tính toán các ràng buộc trên các hệ số nhân này và sau đó giải quyết ở mức tối thiểu constrained. Để thuận tiện, tất cả các số lượng tham chiếu đến hệ số nhân đầu tiên sẽ có chỉ số dưới 1, trong khi tất cả các đại lượng tham chiếu đến hệ số nhân thứ hai sẽ có chỉ số dưới 2. Bởi vì chỉ có hai hệ số nhân, các ràng buộc có thể dễ dàng được hiển thị theo hai chiều (xem hình 3). Các ràng buộc ràng buộc (9) khiến hệ số nhân Lagrange nằm trong một hộp, trong khi hạn chế bình đẳng tuyến tính (6) khiến hệ số nhân Lagrange nằm trên một đường chéo. Do đó, giới hạn tốithiểu của hàm mục tiêu phải nằm trên một đoạn đường chéo (như trong hình 3). Hạn chế này giải thích lý do tại sao hai là số nhân Lagrange tối thiểu có thể được tối ưu hóa: nếu SMO chỉ tối ưu hóa một hệ số nhân, nó không thể fulfill ràng buộc bình đẳng tuyến tính ở mỗi bước.

Các đầu của đoạn đường chéo có thể được thể hiện khá đơn giản. Không mất tính tổng quát, thuật toán đầu tiên tính hệ số nhân Lagrange thứ hai α2 và tính toán các đầu của đường chéo segment về α2. Nếu mục tiêu *y*1 không bằng mục tiêu *y*2, thì các giới hạn sau áp dụng cho α2:

|  |  |
| --- | --- |
| *L* = tối đa( ,0 α2  −α1), *H*  = min*(C,C*  + −α2  α1).  Nếu mục tiêu *y*1 bằng mục tiêu *y*2, thì các giới hạn sau áp dụng cho α2: | (13) |
| *L* = tối đa(0,α2  + −α1 *C*), *H*  = min*(C,*α2  +α1). | (14) |

Đạo hàm thứ hai của hàm mục tiêu dọc theo đường chéo có thể được biểu thị dưới dạng:

r r r r r r

### η= K x x( 1, 1) + K x( 2 , 2 ) − 2Kx( 1, 2 ). (15)

Trong trường hợp bình thường, hàm khách quan sẽ được xác định dương, sẽ có mức tối thiểu dọc theo hướng hạn chế bình đẳng tuyến tính và η sẽ lớn hơn 0. Trong trường hợp này, SMO tính toán mức tối thiểu theo hướng của ràng buộc:

y 2 mới (*E*1 −  *E*2 )

α α2 = 2 + , (16)

η

trong *đó Ei* = -*ui yi* là lỗi trong ví dụ đào tạo *thứ*i. Bước tiếp theo, mức tối thiểu hạn chế được tìm thấy bằng cách cắt tối thiểu không bị giới hạn đến cuối phân đoạn dòng:

|  |  |  |  |
| --- | --- | --- | --- |
| α2 mới,cắt bớt  =&Kα2new | nếu | *L*<α2new <*H;* | (17) |

%K  *H*  nếu *H*;

' L nếu. 

Bây giờ, *hãy*  = *y y*1 2  . Giá trị của α1 được tính từ mới, cắt, α2:

α1new =α1 + *s*(α2 −α2new, cắt bớt ). (18)

Trong những trường hợp bất thường, η sẽ không tích cực. Một kết η sẽ xảy ra nếu hạt nhân *K* không tuân theo tình trạng của Mercer, điều này có thể khiến hàm khách quan trở nên vô thời hạn. Một dấu hiệu η có thể xảy ra ngay cả với một hạt nhân chính xác, nếu nhiều hơn một ví dụ đào tạo có cùng vectơ đầu vào *x*. Trong mọi trường hợp, SMO sẽ hoạt động ngay cả khi η không tích cực, trong trường hợp đó hàm mục tiêu Ψ nên được đánh giá ở mỗi đầu của phân đoạn dòng:

r r r r

*f*1  = y *E*1( 1 + −*b*) α1K*x*x( 1, 1) −  *s*α2  *K*(*x*1, 2  ), r r r

*f*2  = *y*2  (*E*2 + −*b*) *s*α1K *x*( 1, 2  ) −α2 K *x*( 2  , 2  ),

*L*1  = +α α1 *s*( 2 −  *L*),

(19)

*H*1  = +α α1 *s*( 2 −  *H*),

1 2 r r 1 2 rr r

Ψ L = *L f*1 1 + *Lf*2 + 2L K*x*1 ( 1, 1) + 2 L K *x*x( 2 , 2 ) + *sLL K x x*1 ( 1, 2 ),

1 2 r r1 2 rr r

Ψ H = *H f*1 1 + *Hf*2 + 2 H K *x*1 ( 1, 1) + 2 H K *x*x( 2 , 2 ) + *sHH K x*x1 ( 1, 2 ).

SMO sẽ di chuyển hệ số nhân Lagrange đến điểm cuối có giá trị thấp nhất của hàm mục tiêu. Nếu hàm mục tiêu giống nhau ở cả hai đầu (trong một ε nhỏ cho lỗi roundoff) và hạt nhân tuân theo các điều kiện của Mercer, thì việc giảm thiểu chung không thể tiến triển. Kịch bản đó được mô tả dưới đây.

## 2.2 Heuristics để chọn hệ số nhân nào để tối ưu hóa

Miễn là SMO luôn tối ưu hóa và thay đổi hai hệ số nhân Lagrange ở mỗi bước và ít nhất một trong các hệ số nhân Lagrangeđã vi phạm các điều kiện KKT trướcbước, thì mỗi bước sẽ giảm chức năng khách quan theo định lý Osuna [16]. Do đó, sự hội tụ được đảm bảo. Để tăng tốc độ hội tụ, SMO sử dụng heuristics để chọn hai hệ số Lagrange nàođể cùng tối ưu hóa.

Có hai heuristics lựa chọn riêng biệt: một cho hệ số Lagrange đầu tiên và một cho lần thứ hai. Sự lựa chọn của heuristic đầu tiên cung cấp vòng lặp bên ngoài của thuật toán SMO. Vòng lặp bên ngoài đầu tiên lặp lại trên toàn bộ bộ đào tạo, xác định xem mỗi ví dụ có vi phạm các điều kiện KKT (12). Nếu một ví dụ vi phạm các điều kiện KKT, ví dụ đó đủ điều kiện để tối ưu hóa. Sau khi một người đi qua toàn bộ bộ đào tạo, vòng lặp bên ngoài lặp lại trên tất cảcác ví dụ hệ số eLagrange không phải là 0 hoặc C (các ví dụ không ràng buộc). Một lần nữa, mỗi ví dụ được kiểm tra dựa trên các điều kiện KKT và các ví dụ vi phạm đủ điều kiện để tối ưu hóa. Vòng lặp bên ngoài làm cho lặp đi lặp lại vượt qua các ví dụ không ràng buộc cho đến khi tất cả các ví dụ không ràng buộc tuân theo các điều kiện KKT trong ε. Vòng lặp bên ngoài sau đó quay trở lại và lặp lại trên toàn bộ bộ đào tạo. Vòng lặp bên ngoài tiếp tục xen kẽ giữa các đường chuyền đơn trong toàn bộ bộ đào tạovà nhiều đường chuyền qua tập con không giới hạn cho đến khi toàn bộ bộ đào tạo tuân theo cácion condit KKT trong vòng ε sau đó thuật toán chấm dứt.

Heuristic lựa chọn đầu tiên tập trung thời gian CPU vào các ví dụ có nhiều khả năng vi phạm các điều kiện KKT: tập con không ràng buộc. Khi thuật toán SMO tiến triển, các ví dụ ở bounds có khả năng ở giới hạn, trong khi các ví dụ không ở giới hạn sẽ di chuyển khi các ví dụ khác được tối ưu hóa. Do đó, thuật toán SMO sẽ hiển thị qua tập con không giới hạn cho đến khi tập con đó tự nhất quán, sau đó SMO sẽ quét toàn bộ tập dữ liệu để tìm kiếm bất kỳ ví dụ ràng buộc nào đã bị KKT vi phạm do tối ưu hóa tập con không ràng buộc.

Lưu ý rằng các điều kiện KKT được kiểm tra trong phạm vi ε thực hiện. Thông thường, ε được đặt thành 10-3. Các hệ thống nhận dạng thường không need để đáp ứng các điều kiện KKT với độ chính xác cao: có thể chấp nhận được đối với các ví dụ về biên độ dương để có đầu ratừ 0,999 đến 1,001. Thuật toán SMO (và các thuật toán SVM khác) sẽ không hội tụ nhanhnhư vậy nếu cần thiết để tạo ra very đầu ra độ chính xác cao.

Khi hệ số nhân Lagrange đầu tiên được chọn, SMO chọn hệ số nhân Lagrange thứ hai để tối đa hóa kích thước của bước được thực hiện trong quá trình tối ưu hóa khớp. Bây giờ, việc đánh giá hàm *hạt nhân K* tốn thời gian, vì vậy SMO xấp xỉ kích thước bước theo giá trị tuyệt đối của tử số trong phương trình (16): | *E*1  −  *E*2| . SMO giữ giá trị lỗi lưu trữ E *cho* mọi ví dụ không ràng buộc trong tập đào tạo và sau đó chọn lỗi để tối đa hóa kích thước bước. Nếu *E*1 là dương, SMO chọn một ví dụ có lỗi tối thiểu *E*2. Nếu *E*1 âm, SMO chọn một ví dụ có lỗi tối đa *E*2.

Trong những trường hợp bất thường, SMO không thể đạt được tiến bộ tích cực bằng cách sử dụng heuristic lựa chọn thứ hai được mô tả ở trên. Ví dụ: tiến độ tích cực cannot được thực hiện nếu các ví dụ đào tạo thứ nhất và thứ hai chia sẻ vectơ đầu vào giống hệt nhau *x,* khiến hàm khách quan trở nên bán xác định. Trong trường hợp này, SMO sử dụng một hệ thống phân cấp của heuristics lựa chọn thứ hai cho đến khi nó tìm thấy một cặp Lagrange nhânrs có thể được thực hiện tiến bộ tích cực. Tiến bộ tích cực có thể được xác định bằng cách thực hiện kích thước bước không bằng không khi tối ưu hóa chung của hai hệ số nhân Lagrange . Hệ thống phân cấp của heuristics lựa chọn thứ hai bao gồm những điều sau đây. Nếu heuristic ở trênkhông đạt được tiến bộ tích cực, thì SMO bắt đầu quay lại thông qua các ví dụ không ràng buộc, tìm kiếm một ví dụ thứ hai có thể đạt được tiến bộ tích cực. Nếu không có ví dụ nào không ràng buộc đạt được tiến bộ tích cực, thì SMO bắt đầu quay lại thông qua tập đào tạo entire cho đến khi một ví dụ được tìm thấy có tiến bộ tích cực. Cả việc lặp lại thông qua các ví dụ không giới hạn và lặp lại thông qua toàn bộ bộ đào tạo được bắt đầu tại các địa điểm ngẫu nhiên, để không thiên vị SMO đối với các ví dụ khi bắt đầu tập huấn luyện. Trong hoàn cảnh cực kỳ thoái hóa, không có ví dụ nào sẽ làm cho một ví dụ thứ hai đầy đủ. Khi điều này xảy ra, ví dụ đầu tiên bị bỏ qua và SMO tiếp tục với một ví dụ đầu tiên được chọn khác.

## 2.3 Ngưỡng điện toán the

Ngưỡng *b được* tính toán lại sau mỗi bước, để các điều kiện KKT được đáp ứng cho cả hai ví dụ được tối ưu hóa. Ngưỡng *b*1 sau đây là hợp lệ khi α1 mới không ở giới hạn, vì nó buộc đầu ra của SVM phải là *y*1 when đầu vàolà *x*1:

mới r r mới, cắt r r

### b1 = E1 + y1(α1 −α1)K xx( 1, 1) + y2 (α2 −α2 )Kx( 1, 2 ) +b. (20)

Ngưỡng *b*2 sau đây là hợp lệ khi α2 mới không ở giới hạn, vì nó buộc đầu ra của SVM phải là *y*2 khi đầu vào *x*2:

mới r r mới, cắt r r

### b2 = E2 + y1(α1 −α1)K xx( 1, 2 ) + y2 (α2 −α2 )Kxx( 2 , 2 ) +b. (21)

Khi cả *b*1 và *b*2 đều hợp lệ, chúng đều bằng nhau. Khi cả hai hệ số Lagrange mới đều bị ràng buộc và nếu *L* không bằng *H*,thì khoảng giữa *b*1 và *b*2 là tất cả cácngưỡng phù hợp với điều kiện KKT. SMO chọn ngưỡng nằm giữa  *b*1 và *b*2.

## 2.4 Tối ưu hóa cho SVM tuyến tính r

Để tính toán một SVM tuyến tính, chỉ cần lưu trữ một vectơ *trọng lượng* duy nhất, thay vì tất cả các ví dụ đào tạo tương ứng với hệ số nhân Lagrange không bằng không. Nếu tối ưu hóa chung thành công, vectơ trọng lượng được lưu trữ cần được cập nhật để phản ánh các giá trị hệ số nhân Lagrange mới. Việc cập nhật vectơ trọng lượng rất dễ dàng, do tính tuyến tính của SVM:

r mới r mới r mới, cắt bớt r

*w* = *w*  + *y*1(α1 −α1)*x*1 + *y*2 (α2 −α2 )*x*2. (22)

## 2.5 Chi tiết mã

Mã giả dưới đây mô tả toàn bộ thuật toán SMO:

mục tiêu = điểm vectơ đầu ra mong muốn = ma trận điểm đào tạo

thủ tục takeStep(i1,i2) nếu (i1 == i2) trả về hệ số nhân 0 alph1 = Lagrange cho i1 y1 = target[i1]

E1 = Đầu ra SVM tại điểm [i1] - y1 (kiểm tra bộ nhớ cache lỗi) s = y1 \* y2

Điện toán L, H qua phương trình (13) và (14) nếu (L == H) trả về 0 k11 = kernel(point[i1],point[i1]) k12 = kernel(point[i1],point[i2]) k22 = kernel(point[i2],point[i2]) eta = k11+k22-2\*k12 nếu (eta > 0) { a2 = alph2 + y2\*(E1-E2)/eta nếu (a2 < L) a2 = L khác nếu (a2 > H) a2 = H

} khác

{

Lobj = hàm objective tại a2=L Hobj = hàm objective tại a2=H nếu (Lobj < Hobj-eps) a2 = L else nếu (Lobj > Hobj+eps) a2 = H else a2 = alph2

} nếu (|a2-alph2| < eps\*(a2+alph2+eps)) trả về 0 a1 = alph1+s\*(alph2-a2)

Cập nhật ngưỡng để phản ánh sự thay đổi trong hệ số nhân Lagrange

Cập nhật vectơ trọng lượng để phản ánh sự thay đổi trong a1 &a2, nếu SVM là tuyến tính

Cập nhật bộ nhớ cache lỗi bằng hệ số nhân Lagrange mới

Lưu trữ a1 trong mảng alpha Lưu trữ a2 trong mảng alpha trả về 1 endprocedure

thủ tục kiểm traExample(i2) y2 = mục tiêu [i2]

alph2 = Hệ số nhân Lagrange cho i2

E2 = Đầu ra SVM tại điểm [i2] - y2 (kiểm tra bộ nhớ cache lỗi) r2 = E2 \* y2

nếu ((r2 < -tol && alph2 < C) || (r2 > tol && alph2 > 0))

{ if (số không 0 & không C alpha > 1)

{ i1 = kết quả của heuristic lựa chọn thứ hai (phần 2.2) nếu takeStep(i1,i2) trả về 1

}

vòng lặp trên tất cả các alpha không bằng không và không C, bắt đầu tại một điểm ngẫu nhiên

{ i1 = danh tính của alpha hiện tại nếu takeStep(i1,i2) trả về 1

} lặp lại trên tất cả các i1 có thể, bắtđầuing tại một điểm ngẫu nhiên

{ i1 = biến vòng lặp nếu (takeStep(i1,i2) trả về 1

} } trả về 0 endprocedure

thói quen chính:

numChanged = 0; kiểm traTất cả = 1; trong khi (numChanged > 0 |tất cả)

{ numChanged = 0; nếu (examineAll) vòng lặp I trên tất cả các ví dụ đào tạo numChanged += examineExample(I) vòng lặp khác tôi qua các ví dụ trong đó alpha không phải là 0 & không phải C numChanged += examineExample(I) nếu (examineAll == 1) examineAll = 0 else if (numChanged == 0) examineAll = 1 }

## 2.6 Mối quan hệ với các thuật toán trước đó

Thuật toán SMO có liên quan cả với SVM trước đó và các thuật toán tối ưu hóa. Thuật toán SMO có thể được coi là một trường hợp đặc biệt của thuật toán Osuna, trong đó kích thước của tối ưu hóa là hai và cả hai hệ số nhân Lagrange được thay thế ở mỗi bước hệ số nhân mới đượcchọn thông qua heuristics tốt.

Thuật toán SMO có liên quan chặt chẽ với một loạt các thuật toán tối ưu hóa được gọi là phương pháp Bregman [3] hoặc phương pháp hành động hàng [5]. Các phương pháp này giải quyết các vấn đề lập trình lồi với các ts hạnchế tuyến tính. Chúng là những phương pháp lặp đi lặp lại trong đó mỗi bước chiếu điểm nguyên thủy hiện tại lên từng ràng buộc. Một phương pháp Bregman chưa sửa đổi không thể giải quyết vấn đề QP (11) trực tiếp, bởi vì ngưỡng trong SVM tạo ra một ràng buộc bình đẳng tuyến tính trong vấn đề dual. Nếu chỉ có một ràng buộc được dự kiến cho mỗi bước, ràng buộc bình đẳng tuyến tính sẽ bị vi phạm. Nói một cách kỹ thuật hơn, vấn đề nguyên thủy là giảm thiểu định mức của vectơ rr trọng lượng *w trên* không gian kết hợp của tất cả các trọng lượng có thể vect hoặc *w* với ngưỡng *b tạo* ra một Bregman *D*-projection mà không có một tối thiểu duy nhất [3][6].

Thật thú vị khi xem xét một SVM nơi ngưỡng *b* được cố định ở mức 0, thay vì được giải quyết. Một SVM ngưỡng cố định sẽ không có ràng buộcbình đẳng ar dòng(6). Do đó, chỉ có một hệ số nhân Lagrange sẽ cần được cập nhật tại một thời điểm và một phương pháp hành động hàng có thể được sử dụng. Thật không may, một phương pháp Bregman truyền thống vẫn không áp dụng cho các SVM như vậy, do các biến chùng xuống ξ*i* trong phương trình (8). Sự hiện diện của các biến chùng xuống khiến phép chiếu Bregman r *D*trở nên không độc đáo trong không gian kết hợpcủa vectơ trọng lượng *w* và biến chùng xuống

ξ*i*

May mắn thay, SMO có thể được sửa đổi để giải quyết các SVM ngưỡng cố định. SMO sẽ cập nhật hệ sốnhân Lagrange indi vidual là tối thiểu Ψ theo kích thước tương ứng. Quy tắc cập nhật là

y *E*mới 1 1

α α1 = 1 + ~~r~~   r . (23)

*K x x*( 1, 1)

Phương trình cập nhật này buộc đầu ra của SVM là *y*1 ( tương tựnhư phương pháp Bregman hoặc

Phương pháp QP của Hildreth [10]). Sau khi máy tính α tính toán, nó được cắtvào khoảng [0,*C*] (không giống như các phương pháp trước). Sự lựa chọn hệ số nhân Lagrange để tối ưu hóa giống như lựa chọn đầu tiên được mô tả trong phần 2.2.

SMO ngưỡng cố định cho SVM tuyến tính tương tự về khái niệm với quy tắc thư giãn nhận thức [8], trong đó đầu ra của nhận thức được điều chỉnh bất cứ khi nào có lỗi, do đó đầu ra chính xác nằm ở lề. Tuy nhiên, thuật toán SMO ngưỡng cố định đôi khi sẽ làm giảm tỷ lệ đầu vào đàotạo trong vectơ trọng lượng để m tiên đềlề. Quy tắc thư giãn liên tục làm tăng số lượng đầu vào đào tạo trong vectơ trọng lượng và do đó, không phải là lề tối đa. SMO ngưỡng cố định cho hạt nhân Gaussian cũng liên quan đến thuật toán mạng phân bổ tài nguyên (RAN) [18]. Khi RAN phát hiện một số loại lỗi nhất định, nó sẽ phân bổ hạt nhân để sửa lỗi chính xác. SMO sẽ thực hiện tương tự. Tuy nhiên, SMO / SVM sẽ điều chỉnh chiều cao của hạt nhân để tối đa hóa lề trong một không gian tính năng, trong khi RAN sẽ chỉ sử dụng LMS to điều chỉnh chiều cao và trọng lượng của hạtnhân.

# 3 SMO ĐIỂM CHUẨN

Thuật toán SMO đã được thử nghiệm dựa trên thuật toán học tập SVM chunking tiêu chuẩn trên một loạt các điểm chuẩn. Cả hai thuật toán đều được viết bằng C++, sử dụng trình biên dịch Visual C++ 5.0 của Microsoft. Cả hai thuật toán đều được chạy trên bộ xử lý Pentium II 266 MHz không tải chạy Windows NT 4.

Cả hai thuật toán đều được viết để khai thác sự thưa thớt của vectơ đầu vào. Cụ thể hơn, các chức năng hạt nhân dựa vào các sản phẩm chấm trong vòng lặp bên trong. Nếu tanh ta nhập vào là một vectơ thưa thớt, thì một đầu vào có thể được lưu trữ dưới dạng một mảng thưa thớt và sản phẩm dấu chấm sẽ chỉ đơn thuần là lần đầu vào không bằng không, tích lũy các đầu vào không bằng không nhân với trọng lượng tương ứng. Nếu đầu vào là một vectơ nhị phân thưa thớt, thì vị trí của "1" trong đầu vào có thể được lưu trữ và sản phẩm dấu chấm sẽ tính tổng các trọng lượng tương ứng với vị trí của "1" trong đầu vào.

Thuật toán chunking sử dụng thuật toán gradient liên hợp dự kiến [11] làm bộ giải QP của nó, như suggested bởi Burges [4]. Để đảm bảo rằng thuật toán chunking là một điểm chuẩn công bằng, Burges đã so sánh tốc độ của mã chunking của mình trên 200 MHz Pentium II chạy Solaris với tốc độ của mã chunking điểm chuẩn (với cá chóng sản phẩm chấm thưa thớte tắt). Tốc độ được tìm thấy là tương đương, điều này cho thấy mã chunking điểm chuẩn là điểm chuẩn hợp lý.

Đảm bảo rằng mã chunking và mã SMO đạt được độ chính xác tương tự sẽ được chăm sóc. Mã SMO và mã chunking sẽ xác định một ví dụ là vi phạm điều kiện KKT nếu đầu ra cách xa giá trịchính xác hoặc nửa khoảng trống của nó hơn 10-3. Ngưỡng 10-3 đã được chọn là một lỗi không đáng kể trong các tác vụ phân loại. Mã gradient liên hợp dựkiến có ngưỡng dừng, mô tả sự cải thiện tương đối tối thiểu trong hàm mục tiêu ở mỗi bước [4]. Nếu gradient liên hợp dự kiến thực hiện một bước trong đó cải tiến tương đối nhỏ hơn mức tối thiểu này, mãient chấm điểm liên hợp chấm dứt và một bước chunking khác được thực hiện. Burges [4] khuyên bạn nên sử dụng hằng số 10-10 cho mức tối thiểu này.

Trong các thí nghiệm dưới đây, việc dừng gradient liên hợp dự kiến ở độ chính xác 10-10 thường để lại vi phạm KKT lớn hơn 10-3, đặc biệt là đối với các vấn đề quy mô rất lớn. Do đó, thuật toán khối điểm chuẩn đã sử dụng heuristic sau để đặt ngưỡng dừng gradient liên hợp. Ngưỡng bắt đầu từ 3x10-10. Sau mỗi bước chunking, đầu ra được tính cho một vídụ ll có hệ số nhânLagrange không bị ràng buộc. Các đầu ra này được tính toán để tính toán giá trị cho ngưỡng (xem [4]). Mỗi ví dụ cho thấy một ngưỡng được đề xuất. Nếu ngưỡng đề xuất lớn nhất cao hơn 2x10-3 so với ngưỡng đề xuất smallest, thì các điều kiện KKT không thể được thực hiện trong vòng 10-3. Do đó, bắt đầu từ đoạn tiếp theo, ngưỡng gradient liên hợp giảm hệ số 3. Heuristic này sẽ tối ưu hóa tốc độ của gradient liên hợp: nó sẽ chỉ sử dụng độ chính xác cao trên các vấn đề khó khăn nhất. Đối với hầu hết các bài kiểm tra được mô tả dưới đây, ngưỡng vẫn ở mức 3x10-10. Ngưỡng nhỏ nhất được sử dụng là 3,7x10-12, xảy ra ở cuối đoạn cho vấn đề phân loại trang web lớn nhất.

Thuật toán SMO đã được thử nghiệm trên một nhiệm vụ dự đoán thu nhập, một nhiệm vụ phân loại trang web và hai bộ dữ liệu nhân tạo khác nhau. Tất cả thời gian được liệt kê trong tất cả các bảng đều ở trong giây CPU.

## 3.1 Dự đoán thu nhập

Tập dữ liệu đầu tiên được sử dụng để kiểm tra tốc độ của SMO là bộ dữ liệu "người lớn" UCI, có sẵn ở mức ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult. SVM đã được đưa ra 14 thuộc tính của một hình thức điều tra dân số của một hộ gia đình. Nhiệm vụ của SVM là dự đoán liệu household đó có thu nhập lớn hơn 50.000 đô la hay không. Trong số 14 thuộc tính, tám thuộc tính được phân loại và sáu thuộc tính là liên tục. Để dễ thử nghiệm, sáu thuộc tính liên tục đã được phân chia thành các quintiles, mang lại tổng cộng 123 thuộc tính nhị phân, trong đó 14 thuộc tính là đúng. Có 32562 ví dụ trong bộ đào tạo "người lớn". Hai SVM khác nhau đã được đào tạo về vấn đề này: một SVM tuyến tính và một hàm cơ sở xuyên tâm SVM sử dụng hạt nhân Gaussian với phương sai 10. Phương sai này được chọn để giảm thiểutốc độ lỗi trên một bộ xácthực. Giá trị giới hạn *của C* được chọn là 0,05 cho SVM tuyến tính và 1 cho RBF / SVM. Một lần nữa, giá trị giới hạn này đã được chọn để giảm thiểu lỗi trên một tập hợp xác thực.

Hiệu suất thời gian của thuật toán SMO versu s thuật toán chunking choSVM tuyến tính trên tập dữ liệu người lớn được hiển thị trong bảng dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 1605 | 0.4 | 37.1 | 42 | 633 |
| 2265 | 0.9 | 228.3 | 47 | 930 |
| 3185 | 1.8 | 596.2 | 57 | 1210 |
| 4781 | 3.6 | 1954.2 | 63 | 1791 |
| 6414 | 5.5 | 3684.6 | 61 | 2370 |
| 11221 | 17.0 | 20711.3 | 79 | 4079 |
| 16101 | 35.3 | N/a | 67 | 5854 |
| 22697 | 85.7 | N/a | 88 | 8209 |
| 32562 | 163.6 | N/a | 149 | 11558 |

Kích thước bộ đào tạo được thay đổi bằng cách lấy các tập con ngẫu nhiên của bộ đào tạo đầy đủ. Các tập hợp con này được lồng nhau. Các mục "N /A" trong cột thời gian chunking có ma trận quá lớn để phù hợp với 128 Megabyte, do đó không thể được tính thời gian do bộ nhớ đập. Số lượng vectơ hỗ trợ không giới hạn và số lượng vectơ hỗ trợ ràng buộc được xác định từ SMO: kết quả khối thay đổi theo một lượng nhỏ, do dung sai của sự không chính xác xung quanh các điều kiện KKT.

Bằng cách lắp một dòng vào cốt truyện nhật ký của thời gian làm việc so với kích thước tập luyện, có thể bắt nguồn tỷ lệthực nghiệm cho SMO và chunking. Thời gian đào tạo SMO có thang đo là ~*N*  1,9, trong khi vảy khối là ~ *N*  3,1. Do đó, SMO cải thiện quy mô thực nghiệm cho vấn đề này bằng nhiều đơn đặt hàng.

Hiệu suất thời gian của SMO và chunking bằng svm Gaussian được hiển thị dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 1605 | 15.8 | 34.8 | 106 | 585 |
| 2265 | 32.1 | 144.7 | 165 | 845 |
| 3185 | 66.2 | 380.5 | 181 | 1115 |
| 4781 | 146.6 | 1137.2 | 238 | 1650 |
| 6414 | 258.8 | 2530.6 | 298 | 2181 |
| 11221 | 781.4 | 11910.6 | 460 | 3746 |
| 16101 | 1784.4 | N/a | 567 | 5371 |
| 22697 | 4126.4 | N/a | 813 | 7526 |
| 32562 | 7749.6 | N/a | 1011 | 10663 |

Thuật toán SMO chậm hơn đối với SVM phi tuyến tính so với SVM tuyến tính, vì thời gian bị chi phối bởi việc đánh giá SVM. Ở đây, thời gian đào tạo SMO quy mô *là ~ N*  2.1, trong khi chunking quy mô là ~ *N*  2.9. Một lần nữa, tỷ lệ của SMO nhanh hơn khoảng một đơn hàng so với chunking. Thử nghiệm dự đoán thu nhập chỉ ra rằng đối với các vấn đề thưa thớt trong thế giới thực với nhiều vectơ hỗ trợ bị ràng buộc, SMO nhanh hơn nhiều so với chunking.

## 3.2 Phân loại trang web

Thử nghiệm thứ hai của SMO là phân loại văn bản: phân loại xem một trang web có thuộc danh mục hay không. Bộ đào tạo bao gồm 49749 trang web, với 300 thuộc tính từ khóa nhị phân thưa thớt được trích xuất từ mỗi trang web. Hai SVM khác nhau đã được thử về vấn đề này: một SVM tuyến tính và một SVM Gaussian phi tuyến tính, sử dụng phương sai 10. Giá *trị C* cho SVM tuyến tính được chọn là 1, trong khi giá trị *C* cho SVM phi tuyến tính được chọn là 5. Một lần nữa, các tham số này đã được chọn để tối đa hóa hiệu suất trên một bộ xác thực.

Thời gian cho SMO so với chunking cho một SVM tuyến tính được hiển thị trong bảng, dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 2477 | 2.2 | 13.1 | 123 | 47 |
| 3470 | 4.9 | 16.1 | 147 | 72 |
| 4912 | 8.1 | 40.6 | 169 | 107 |
| 7366 | 12.7 | 140.7 | 194 | 166 |
| 9888 | 24.7 | 239.3 | 214 | 245 |
| 17188 | 65.4 | 1633.3 | 252 | 480 |
| 24692 | 104.9 | 3369.7 | 273 | 698 |
| 49749 | 268.3 | 17164.7 | 315 | 1408 |

Đối với SVM tuyến tính trên tập dữ liệu này, thời gian đào tạo SMO quy mô *là ~ N*  1,6, trong khi chunking thang âm là ~ *N*  2,5. Thí nghiệm này là một tình huống khác trong đó SMO vượt trội hơn chunking trong thời gian tính toán.

Thời gian cho SMO so với chunking cho một máy ảo S phituyến tính được hiển thị trong bảng, dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 2477 | 26.3 | 64.9 | 439 | 43 |
| 3470 | 44.1 | 110.4 | 544 | 66 |
| 4912 | 83.6 | 372.5 | 616 | 90 |
| 7366 | 156.7 | 545.4 | 914 | 125 |
| 9888 | 248.1 | 907.6 | 1118 | 172 |
| 17188 | 581.0 | 3317.9 | 1780 | 316 |
| 24692 | 1214.0 | 6659.7 | 2300 | 419 |
| 49749 | 3863.5 | 23877.6 | 3720 | 764 |

Đối với SVM phi tuyến tính trên tập dữ liệu này, thời gian đào tạo SMO quy mô *là ~ N*  1,7, trong khi chunking thang đo là ~ *N*  2.0. Trong trường hợp này, tỷ lệ cho SMO có phần tốt hơn chunking: SMO là một yếu tố nhanh hơn từ hai đến sáu lần so với chunking. Kiểm tra ntrên tuyến tính cho thấy SMO vẫn nhanh hơnchunking khi số lượng vectơ hỗ trợ không ràng buộc lớn và tập dữ liệu đầu vào thưa thớt.

## 3.3 Bộ dữ liệu nhân tạo

SMO cũng đã được thử nghiệm trên các bộ dữ liệu được tạo ra nhân tạo để khám phá ce thực hiệncủa SMO trong các tình huống cực đoan. Tập dữ liệu nhân tạo đầu tiên là một tập dữ liệu hoàn toàn phân tách tuyến tính. Dữ liệu đầu vào là vectơ nhị phân ngẫu nhiên 300 chiều, với tỷ lệ 10% của đầu vào "1". Một vectơ trọng lượng 300 chiều được tạo ra ngẫunhiên y trong [-1,1]. Nếu dấu chấm của trọng lượng có điểm đầu vào lớn hơn 1, thì nhãn dương được gán cho điểm đầu vào. Nếu sản phẩm dấu chấm nhỏ hơn –1, nhãn âm sẽ được gán. Nếu sản phẩm chấm nằm giữa -1 và 1, t anh tatrỏ sẽ bị loại bỏ. Một SVM tuyến tính phù hợp với tập dữ liệu này.

Tập dữ liệu phân tách tuyến tính là vấn đề đơn giản nhất có thể xảy ra đối với một SVM tuyến tính. Không có gì đáng ngạc nhiên, việc mở rộng quy mô với kích thước bộ đào tạo là tuyệt vời cho cả SMO và chunking. Thời gian chạy được hiển thị trong bảng dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 1000 | 15.3 | 10.4 | 275 | 0 |
| 2000 | 33.4 | 33.0 | 286 | 0 |
| 5000 | 103.0 | 108.3 | 299 | 0 |
| 10000 | 186.8 | 226.0 | 309 | 0 |
| 20000 | 280.0 | 374.1 | 329 | 0 |

Ở đây, thời gian chạy SMO quy mô *là ~ N,* tốt hơn một chút so với tỷ lệ để phân đoạn, đó *là ~ N*  1,2. Đối với vấn đề thưa thớt dễ dàng này, do đó, chunking và SMO thường có thể so sánh được. Cả hai thuật toán đều được đào tạo với *C* được đặt thành 100. Kích thước khối cho chunking được đặt thành 500.

Khả năng tăng tốc của cả thuật toán SMO và thuật toán chunking due đến mã sản phẩm dấu chấm thưa thớt có thể được đo trên tập dữ liệu dễ dàng này. Cùng một bộ dữ liệu đã được thử nghiệm có và không có mã sản phẩm chấm thưa thớt. Trong trường hợp thí nghiệm không thưa thớt, mỗi điểm đầu vào được lưu trữ dưới dạng vectơ 300 chiều của phao. Kết quả của thí nghiệm thưa thớt/không thưa thớt được hiển thị trong bảng dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO  (thưa thớt) | Thời gian SMO  (không thưa thớt) | Thời gian chunking (thưa thớt) | Thời gian chunking (không thưa thớt) |
| 1000 | 15.3 | 145.1 | 10.4 | 11.7 |
| 2000 | 33.4 | 345.4 | 33.0 | 36.8 |
| 5000 | 103.0 | 1118.1 | 108.3 | 117.9 |
| 10000 | 186.8 | 2163.7 | 226.0 | 241.6 |
| 20000 | 280.0 | 3293.9 | 374.1 | 397.0 |

Đối với SMO, việc sử dụng cấu trúc dữ liệu thưa thớt tăng tốc mã lên hơn hệ số 10, điều này cho thấy thời gian đánh giá của SVM hoàn toàn chiếm ưu thế trong thời gian tính toán SMO. Mã sản phẩm chấm thưa thớt chỉ tăng tốc độ phân đoạntheo hệ số xấp xỉ 1,1, cho thấy việc đánh giá các bước QP số chiếm ưu thế trong tính toán khối. Đối với trường hợp phân tách tuyến tính, hoàn toàn không có hệ số nhân Lagrange ở giới hạn, đây là trường hợp tồi tệ nhất đối với SMO. Do đó, hiệu suất kém của SMO không thưa thớt so với chunking không thưa thớt trong thí nghiệm này nên được coi là một trường hợp tồi tệ nhất.

Thí nghiệm thưa thớt so với không thưa thớt cho thấy một phần của sự vượt trội của SMO so với chunking đến từ việc khai thác mã sản phẩm chấm thưa thớt. Đối vớitunately, nhiều vấn đề trong thế giới thực có đầu vào thưa thớt. Ngoài các bộ dữ liệu từ thực được mô tả trong phần 3.1 và phần 3.2, mọi vấn đề được định lượng hoặc mã hóa thành viên mờ sẽ thưa thớt. Ngoài ra, nhận dạng ký tự quang học [12], nhận dạng ký tự handwritten [1], và hệ số biến đổi sóng của hình ảnh tự nhiên [13] [14] có xu hướng được thể hiện tự nhiên dưới dạng dữ liệu thưa thớt.

Tập dữ liệu nhân tạo thứ hai trái ngược hoàn toàn với tập dữ liệu dễ dàng đầu tiên. Tập thứ hai được tạo ra với các điểm đầu vào nhị phân 300 chiều ngẫu nhiên (10% "1") và nhãn đầu ra ngẫu nhiên. Do đó, các SVM phù hợp với tiếng ồn tinh khiết. Giá *trị C* được đặt thành 0,1, vì vấn đề về cơ bản là không thể giải quyết được. Kết quả cho SMO và chunking áp dụng cho một SVM tuyến tính được hiển thị dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO | Thời gian phân đoạn | Số vectơ hỗ trợ không ràng buộc | Số lượng vectơ hỗ trợ ràng buộc |
| 500 | 1.0 | 6.4 | 162 | 263 |
| 1000 | 3.5 | 57.9 | 220 | 632 |
| 2000 | 15.7 | 593.8 | 264 | 1476 |
| 5000 | 67.6 | 10353.3 | 283 | 4201 |
| 10000 | 187.1 | N/a | 293 | 9034 |

Tỷ lệ cho SMO và chunking cao hơn nhiều trên tập dữ liệu thứ hai. Điều này phản ánh độ khó của vấn đề. Thời gian tính toán SMO có thang đo *là ~ N*  1,8, trong khi thời gian tính toán khối có quy mô là *~ N*  3,2. Tập dữ liệu thứ hai cho thấy SMO vượt trội khi hầu hết các vectơ hỗ trợ bị ràng buộc. Do đó, để xác định sự gia tăng tốc độ gây ra bởi mã sản phẩm chấm thưa thớt, cả SMO và chunking đều được kiểm tra mà không có mã sản phẩm chấm thưa thớt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | Thời gian SMO  (thưa thớt) | Thời gian SMO  (không thưa thớt) | Thời gian chunking (thưa thớt) | Thời gian chunking (không thưa thớt) |
| 500 | 1.0 | 6.0 | 6.4 | 6.8 |
| 1000 | 3.5 | 21.7 | 57.9 | 62.1 |
| 2000 | 15.7 | 99.3 | 593.8 | 614.0 |
| 5000 | 67.6 | 400.0 | 10353.3 | 10597.7 |
| 10000 | 187.1 | 1007.6 | N/a | N/a |

Trong trường hợp SVM tuyến tính, mã sản phẩm chấm thưa thớt tăng SMO khoảng 6, trong khi chunking chỉ tăng lên tối thiểu. Trong thử nghiệm này, SMO nhanh hơn chunking ngay cả đối với dữ liệu không phân biệt.

Tập dữ liệu thứ hai cũng được thử nghiệm bằng cách sử dụng SVM Gaussiantha t có phương sai là 10. Giá *trị C* vẫn được đặt thành 0,1. Kết quả cho các SVM Gaus được trình bày trong hai bảng dưới đây:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kích thước bộ đào tạo | | Thời gian SMO | | Thời gian phân đoạn | | Số vectơ hỗ trợ không ràng buộc | | | Số lượng vectơ hỗ trợ ràng buộc | |
| 500 | | 5.6 | | 5.8 | | 22 | | | 476 | |
| 1000 | | 21.1 | | 41.9 | | 82 | | | 888 | |
| 2000 | | 131.4 | | 635.7 | | 75 | | | 1905 | |
| 5000 | | 986.5 | | 13532.2 | | 30 | | | 4942 | |
| 10000 | | 4226.7 | | N/a | | 48 | | | 9897 | |
| Kích thước bộ đào tạo | | Thời gian SMO  (thưa thớt) | | Thời gian SMO  (không thưa thớt) | | Thời gian chunking (thưa thớt) | Thời gian chunking (không thưa thớt) | |
| 500 | | 5.6 | | 19.8 | | 5.8 | 6.8 | |
| 1000 | | 21.1 | | 87.8 | | 41.9 | 53.0 | |
| 2000 | | 131.4 | | 554.6 | | 635.7 | 729.3 | |
| 5000 | | 986.5 | | 3957.2 | | 13532.2 | 14418.2 | |
| 10000 | | 4226.7 | | 15743.8 | | N/a | N/a | |

Đối với svm Gaussian phù hợp với tiếng ồn tinh khiết, thời gian tính toán SMO quy mô *là ~ N*  2,2, trong khi thời gian tính toán khối có quy mô *là ~ N*  3,4. Vỏ tiếng ồn tinh khiết mang lại tỷ lệ tồi tệ nhất cho đến nay, nhưng SMO vượt trội hơn so với khối lượng theo nhiều thứ tự trong scaling. Tổng thời gian chạy của SMO vẫn vượt trội so với chunking, ngay cả khi áp dụng cho dữ liệu không thưa thớt. Sự co thắt của dữ liệu đầu vào mang lại tốc độ xấp xỉ hệ số 4 cho SMO cho trường hợp phi tuyến tính, điều này chothấy sản phẩm chấm speed vẫn đang thống trị thời gian tính toán SMO cho phi tuyến tính

SVM

# 4 KẾT LUẬN

SMO là một thuật toán đào tạo cải tiến cho SVM. Giống như các thuật toán đào tạo SVM khác, SMO chia nhỏ một vấn đề QP lớn thành một loạt các vấn đề QP nhỏ hơn. Không giống như cáclgorithms khác, SMO sử dụng các vấn đề QP nhỏ nhất có thể, được giải quyết nhanh chóng và phân tích, thường cải thiện đáng kể thời gian mở rộng và tính toán của nó.

SMO đã được thử nghiệm trên cả các vấn đề trong thế giới thực và các vấn đề nhân tạo. Từ các thử nghiệm này, những điều sau đây có thể được suy ra:

* SMO có thể được sử dụng khi người dùng không dễ dàng truy cập vào gói lập trình bậc hai và / hoặc không muốn điều chỉnh gói QP đó.
* SMO hoạt động rất tốt trên SVM, nơi nhiều hệ số nhân Lagrange bị ràng buộc.
* SMO hoạt động tốt cho SVM tuyến tính beca sử dụng thời gian tính toáncủa SMO bị chi phối bởi đánh giá SVM và việc đánh giá SVM tuyến tính có thể được thể hiện dưới dạng một sản phẩm chấm duy nhất, thay vì tổng hợp các hạt nhân tuyến tính.
* SMO hoạt động tốt cho SVM với đầu vào thưa thớt, ngay cả đối với SVM phi tuyến tính, bởi vì thời gian tính toán hạt nhân điện tử thứcó thể được giảm, tăng tốc trực tiếp SMO. Bởi vì chunking dành phần lớn thời gian của nó trong mã QP, nó không thể khai thác tuyến tính của SVM hoặc sự thưa thớt của dữ liệu đầu vào.
* SMO sẽ hoạt động tốt cho các ems probllớn, bởi vì quy mô của nó với kích thước tập huấn luyện là tốt hơn so với chunking cho tất cả các vấn đề thử nghiệm đã thử cho đến nay.

Đối với các bộ thử nghiệm khác nhau, thời gian đào tạo của SMO theo kinh nghiệm quy mô *giữa ~ N* và *~ N*  2,2. Thời gian đào tạo của vảy khối giữa *~N*  1,2 và ~ *N*  3,4. Việc mở rộng SMO có thể tốt hơn một đơn hàng so với chunking. Đối với các bộ thử nghiệm trong thế giới thực, SMO có thể nhanh hơn 1200 lần đối với SVM tuyến tính và hệ số nhanh hơn 15 lần đối với SVM phi tuyến tính.

Vì dễ sử dụng một tỷ lệtốt hơn với kích thước tập huấn luyện, SMO là một ứng cử viên mạnh mẽ để trở thành thuật toán đào tạo SVM tiêu chuẩn. Các thí nghiệm đánh giá nhiều hơn so với các kỹ thuật QP khác và heuristics Osuna tốt nhất là cần thiết trước khi kết luận cuối cùng có thể được rút ra.

# XÁC NHẬN

Cảm ơn Lisa Heilbron đã hỗ trợ chuẩn bị văn bản. Cảm ơn Chris Burges đã chạy một tập dữ liệu thông qua mã gradient liên hợp dự kiến của mình. Cảm ơn Leonid Gurvits đã chỉ ra sự giống nhau của SMO với các phương pháp Bregman.

# REFERENCES

1. Bengio, Y., LeCun, Y., Henderson, D., "Globally Trained Handwritten Word Recognizer using Spatial Representation, Convolutional Neural Networks and Hidden Markov Models," *Advances in Neural Information Processing Systems,*  5, J. Cowan, G. Tesauro, J. Alspector, chủ biên, 937-944, (1994).
2. Boser, B. E., Guyon, I.M., Vapnik, V., "Thuật toán đào tạo cho phân loại lề tối ưu", Hội thảo thường niên lần thứ *năm về lý thuyết học tập tính toán,* ACM, (1992).
3. Bregman, L.M., "Phương pháp thư giãn để tìm điểm C ommon của bộ lồi và ứng dụng của nó để giảiquyết các vấn đề trong lập trình lồi," Toán tính toán và Vật lý toán học Liên *Xô,* 7:200-217, (1967).
4. Burges, C. J.C., "Hướng dẫn về hỗ trợ máy vectơ để nhận dạngmẫu" được gửi đến Khai thác dữ liệu và Khám phá kiến thức, http://svm.research.belllabs.com/SVMdoc.html,(1998).
5. Censor, Y., "Phương pháp hành động hàng cho các hệ thống khổng lồ và thưa thớt và các ứng dụng của *chúng", Siam Review,* 23(4):444-467, (1981).
6. Censor, Y., Lent, A., "Một phương pháphành động hàng ite cho lập trình lồi khoảng thời gian," *J. Tối ưu hóa lý thuyết và ứng dụng,*  34(3):321-353, (1981).
7. Cortes, C., Vapnik, V., "Hỗ trợ mạng vector" *Machine Learning,*20:273-297, (1995).
8. Duda, R. O., Hart, P. E., *Pattern Classification và Phân tích cảnh,*John Wiley & Sons, (1973).
9. Joachims, T., "Phân loại văn bản với máy vectơ hỗ trợ", Báo cáo kỹ thuật LS VIII, số 23, Đại họcDortmund,ftp://ftp-ai.informatik.unidortmund.de/pub/Reports/report23.ps.Z, (1997).
10. Hildreth, C., "Một quy trình lập trình bậc hai", *Naval Research Logistics Quarterly,*  4:79-85, (1957).
11. ^ Gill, P. E., Murray, W., Wright, M. H., Tối ưu *hóa thực tế,* Báo chí học thuật, (1981).
12. LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller,

U. A., Sackinger, E., Simard, P. và Vapnik, V., "Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition," *Neural Networks: The Statistical Mechanics Perspective*,Oh, J. H., Kwon, C. and Cho, S. (chủ biên), World Scientific, 261-276, (1995).

1. Mallat, S., *A*  *Wavelet Tour of Signal Processing, Academic* Press, (1998).
2. Oren, M., Papageorgious, C., Sinha, P., Osuna, E., Poggio, T., "Phát hiện người đi bộ bằng cách sử dụng các mẫu Wavelet," *Proc. Thị giác máy tính và Nhận dạng mẫu '97,*  193-199, (1997).
3. Osuna, E.,Freund, R., Girosi, F., "Máy vectơ hỗ trợ đào tạo: Một ứng dụng để phát hiện khuôn mặt," Proc. Thị giác máy tính và nhận dạng *mẫu '97,*130-136, (1997).
4. Osuna, E., Freund, R., Girosi, F., "Thuật toán đào tạo được cải tiến để hỗ trợ máy vectơ," *Proc.* *IEEE NNSP '97*, (1997).
5. Osuna, E., Giao tiếp cá nhân.
6. Platt, J.C., "Một mạng phân bổ tài nguyên cho nội suy chức năng," *Tính toán thần kinh,* 3(2):213-225, (1991).
7. Vapnik, V., *Ước tính sự phụ thuộc dựa trên dữ liệu thực nghiệm,*Springer-Verlag, (1982).
8. Vapnik, V., *Bản chất của lý thuyết học thống kê,*  Springer-Verlag, (1995).

# PHỤ LỤC: DẪN XUẤT GIẢM THIỂU HAI VÍ DỤ

Mỗi bước của SMO sẽ tối ưu hóa hai hệ số Lagrange. Không mất tính tổng quát, hãy để hai hệ số nhân này được α1 và α2. Do đó, hàm Ψ mục tiêu từ phương trình (11) có thể được viết là

Ψ = 21 *K*11α12 + 21 *K*22α22 + *sK*12α1α2 + *y*1α1 1*v*  + *y*2α2*v*2 − − +Ψα1 hằng số α2 , (24)

đâu

r r

*Kij*  = K *x*  *x*( *i* , *j*  ),

## *vi* = ∑*N* y *j* *j* *Kij* = *ui* +*b* − *y K* *i* − *y K* *i* , (25)

*j*=3

và các biến có gắn dấu sao cho biết các giá trị ở cuối lần lặp trước đó. Ψ là các thuật ngữ không phụ thuộc vào α1 hoặc α2.

Mỗi bước sẽ tìm thấy mức tối thiểu dọc theo đường được xác định bởi ràng buộc bình đẳng tuyến tính (6). Ràng buộc bình đẳng tuyến tính đó có thể được thể hiện dưới dạng

α1  + *s*α2  =α1\*  + s *α*\*2  = *w*. (26)

Hàm khách quan dọc theo ràng buộc bình đẳng tuyến tính có thể được thể hiện chỉ α2:

Ψ = 21 *K*11(*w*−  *s*α2 )2 + 21 *K*22α22 + *sK*12 (*w*−  *s*αα2 ) 2

(27)

+ *y w*1( − s *α*2 )*v*1 − +*w s*α α α2 + *y*2 2*v*2 − +2 hằng số Ψ.

Cực chi của hàm mục tiêu ở *mức d*Ψ

= −*sK*11(*w*  −  *s*α2 ) + *K*22α α2 −  *K*12 2 + *sK*12 (*w*−  *s*α2 ) − y *v*2 1 + *s*+ y *v*2 2 −1= 0. (28) *d*α2

Nếu dẫn xuất thứ hai là dương, đó là trường hợp thông thường, thì tối thiểu α2 có thể được thể hiện dưới dạng

α2  (*K*11  + *K*22  − 2*K*12  ) = s *K*( 11  −  *K*12  )*w*+ y *v*2  ( 1  − + −*v*2  ) 1 *s*. (29)

Mở rộng phương trình cho *năng suất w* *và* v

α2 (*K*11 + *K*22 − 2*K*12 ) =α\*2 (*K*11 + *K*22 −2*K*12 ) + y *u*2 ( 1 −*u*2 + *y*2 −  *y*1). (30)

Nhiều đại số mang lại phương trình (16).