Nhóm Kiên + Thành + Long (Grid Search)

Ngày 3/12/2010

Mục lục

[1 TÓM TẮT 2](#_Toc279112350)

[2 NỀN TẢNG LÝ THUYẾT 2](#_Toc279112351)

[2.1 Grid Search 2](#_Toc279112352)

[2.2 Pattern Search 3](#_Toc279112353)

[2.2.1 Sơ bộ về ý tưởng 3](#_Toc279112354)

[2.2.2 Thuật toán 3](#_Toc279112355)

[2.2.3 Các ý quan trọng về thuật toán Pattern Search 4](#_Toc279112356)

[2.2.4 Chiến lược đề xuất 4](#_Toc279112357)

[3 CÀI ĐẶT 4](#_Toc279112358)

[3.1.1 Grid Search 5](#_Toc279112359)

[3.1.2 Pattern Search 6](#_Toc279112360)

[4 THÍ NGHIỆM 6](#_Toc279112361)

[4.1 Dữ liệu 6](#_Toc279112362)

[4.2 Thí nghiệm số 1: Pattern Search 6](#_Toc279112363)

[4.3 Thí nghiệm số 2: {SVR Default Value vs Grid Search vs Bag Pattern Search} vs ANN 9](#_Toc279112364)

[4.3.1 Bộ IBM 9](#_Toc279112365)

[4.3.2 Bộ FPT 10](#_Toc279112366)

[5 KẾT LUẬN 12](#_Toc279112367)

[6 CÔNG VIỆC TƯƠNG LAI 12](#_Toc279112368)

[7 TÀI LIỆU THAM KHẢO 13](#_Toc279112369)

# TÓM TẮT

Ở đây, mục tiêu của ta là khảo sát các phương sát các phương pháp lựa chọn tham số cho SVM, cụ thể là Grid Search và Pattern Search.

Grid Search là cách truyền thống để chọn bộ tham số cho SVM. Nó chẳng qua chỉ là vét cạn không gian tìm kiếm và do đó, nó tốn rất nhiều chi phí. Từ đây, người ta mới cải tiến Grid Search bằng cách thực hiện qua 2 bước: (1) Tìm kiếm thưa, (2) Tìm kiếm kỹ xung quanh điểm tốt nhất tìm được bởi bước 1.

Đặc điểm của Pattern Search là chạy nhanh nhưng có thể bị bẫy bởi cực trị cục bộ không tốt (tùy vào điểm khởi tạo ban đầu). Từ đây, ta đưa ra chiến lược là chạy Pattern Search vài lần để có được vài mô hình. Giá trị dự đoán sẽ bằng trung bình của các giá trị dự đoán của các mô hình này.

Đồng thời, ta cũng tiến hành so sánh hiệu quả của SVR so với ANN trong bài toán dự đoán giá chứng khoán.

Ta tiến hành trên 2 bộ dữ liệu: 1 của nước ngoài (IBM) và 1 của Việt Nam (FPT). Kết quả đã một lần nữa khẳng định tính hiệu quả của SVR so với ANN. Với Grid Search cải tiến và Pattern Search, ta đều thu được kết quả tốt. Với số lần cross-validation ít hơn những hiệu quả vẫn không thưa kém so với Grid Search, Pattern Search đáng để ta nghiên cứu và cải tiến thêm.

# NỀN TẢNG LÝ THUYẾT

## Grid Search

Grid Search chẳng qua là vét cạn không gian tìm kiếm. Đầu tiên, ta cần phải có miền giá trị của các biến cần tìm kiếm. Chẳng hạn, trong không gian hai chiều, đã biết . Kế đến, ta cần phải chọn một hệ số Delta nào đó để rời rạc hóa các miền giá trị này. Chẳng hạn, Delta = 2, ta sẽ xét x = 22, 24, 26, …210 và y = 2-5, 2-3, 2-1, …, 25. Sau đó, ta sẽ xét hết tất cả các trường hợp có thể (ở đây là 5\*6 trường hợp) để chọn ra cặp biến tốt nhất.

Ta nhận thấy hệ số rời rạc hóa Delta càng lớn thì sẽ làm giảm thời gian tính toán, nhưng ngược lại, độ chính xác của nó cũng sẽ giảm đi.

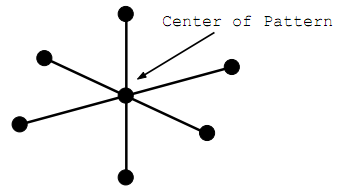
Grid Search là phương pháp truyền thống dùng để chọn bộ tham số cho SVM. Từ nhược điểm cố hữu của nó là quá tốn chi phí, người ta mới đề ra cải tiến như sau:

* Đầu tiên, ta sẽ dùng một grid thưa (Delta lớn) để chọn ra bộ tham số tốt nhất.
* Sau đó, ta sẽ làm kỹ hơn: ta sẽ dùng một grid dày hơn (Delta nhỏ hơn) xung quanh bộ tham số vừa tìm được ở trên.

## Pattern Search

### Sơ bộ về ý tưởng

Đầu tiên, ta cần hiểu thế nào là pattern? Pattern đơn giản chỉ là một nhóm các điểm kề cận nhau trong không gian, trong đó có một điểm ở giữa được gọi là trung tâm của pattern.



Ta sẽ bắt đầu thuật toán từ một điểm bất kỳ trong không gian tìm kiếm. Điểm này được coi là trung tâm của pattern và là điểm “tốt nhất” cho đến thời điểm hiện tại.

Từ điểm trung tâm này, ta sẽ tính giá trị của hàm mục tiêu tại các điềm kề cận nhằm tìm ra được một điểm tốt hơn điểm hiện tại. Các điểm kề cận được với điểm trung tâm được định bởi **hướng** (luôn không đổi) và một thành phần để xác định **khoảng cách** gọi là bước tìm kiếm (search step.)

Nếu tìm được, ta sẽ gán lại trung tâm của pattern là điểm tốt hơn đó. Ngược lại, ta sẽ giảm bước tìm kiếm và lặp lại quá trình trên. Thuật toán sẽ ngừng khi bước tìm kiếm nhỏ hơn một ngưỡng nào đó.

### Thuật toán

* Bước 1: Khởi tạo
  + k = 1
  + Chọn bước tìm kiếm ban đầu và ngưỡng của bước tìm kiếm để dừng thuật toán .
  + Chọn ngẫu nhiên điểm trung tâm xk của pattern, tính f(xk) và gán min = f(xk).
  + BestPoint = xk.
* Bước 2: Nếu , dừng thuật toán!
* Bước 3: For i = 1, …, n //n là số chiều của không gian tìm kiếm
  + và . Tính f() // là dòng thứ i của ma trận đơn vị In
  + Nếu f() < min:
    - min = f()
    - BestPoint =
  + Ngược lại:// Ta đổi ngược hướng tìm kiếm với hy vọng sẽ tìm được điểm tốt hơn
    - và . Tính f()
    - Nếu f() < min:
      * min = f()
      * BestPoint =
* Bước 4:
  + Nếu BestPoint == //Các điểm kế cận không tốt hơn điểm trung tâm 🡪giảm bước tìm kiếm
  + Ngược lại: //Tìm được điểm tốt hơn điểm trung tâm 🡪Gán lại trung tâm của pattern là điểm tốt hơn đó
    - = BestPoint
  + Quay lại bước 2

### Các ý quan trọng về thuật toán Pattern Search

* Thuật toán đơn giản, dễ hiểu, dễ cài đặt, chi phí thấp.
* Khác với cách truyền thống là lấy đạo hàm của hàm mục tiêu (dở khi hàm mục tiêu không “convex”, có nhiều điểm cực trị địa phương; không làm được khi hàm mục tiêu không khả vi), ở đây ta không cần biết hàm mục tiêu có liên tục và khả vi hay không.
* Ngươi ta đã chứng minh được rằng thuật toán hội tụ về cực trị đia phương (dễ hiểu vì ta thấy rằng sau mỗi vòng lặp giá trị hàm mục tiêu tại điểm trung tâm pattern đều giảm hoặc không đổi) và hơn nữa, điểm cực trị này thường khá tốt.

### Chiến lược đề xuất

Từ ưu điểm của Pattern Search là chạy nhanh và nhược điểm là hội tụ về cực trị cục bộ (ta lo ngại sẽ mắc phải một cực trị cục bộ không tốt), ta đề xuất ra 2 chiến lược như sau:

* Chiến lược số 1: Đầu tiên, ta sẽ chạy Pattern Search một vài lần đề tìm ra một vài bộ tham số. Sau đó, ta sẽ chọn ra bộ tham số mà có MSE Cross-Validation nhỏ nhất để huấn luyện.
* Chiến lược số 2: Đầu tiên, ta sẽ chạy Pattern Search một vài lần có một vài mô hình. Giá trị dự đoán sau cùng sẽ bằng trung bình của các giá trị dự đoán tạo bởi các mô hình này (kỹ thuật “bag”).

# CÀI ĐẶT

Ở đây, ta chỉ xét SVR với hàm lỗi và hàm Gaussian kernel.

Với mô hình này, ta cần tìm 3 tham số: C, gamma (của hàm Gauss), và epsilon.

Căn cứ vào các báo cáo và theo kinh nghiệm (với LibSVM khi C lớn, thời gian thực hiện cross-validation sẽ rất lâu), ta chọn miền giá trị cho ba biến này như sau:

Ở đây, trong phạm vi bài toán dự đoán giá chứng khoán, ta sử dụng hàm mục tiêu là độ lỗi MSE của 5 folds cross-validation (dự đoán xu hướng thì có thể sẽ phải sử dụng một độ lỗi khác.) Mục tiêu của ta là tìm được bộ ba tham số (C, Gamma, Epsilon) sao cho tại đó hàm mục tiêu này đạt được một cực trị cục bộ đủ tốt (tất nhiên, đạt được cực trị toàn cục thì càng tốt nhưng thường khó vì rằng mặt thể hiện hàm này thường phức tạp, nhiều cực trị cục bộ.)

Một điểm đáng để ý với hàm mục tiêu này là trong đó, thông qua cross-validation ta cũng đã ngầm hướng đến cái đích là ngăn overfit.

### Grid Search

Thuật toán

* Bước 1: Khởi tạo
* Giá trị bắt đầu của C, gamma, epsilon.
* Số interval (số này thể hiện số điểm grid), chẳng hạn nếu chọn 7 thì sẽ có 7\*7\*7 điểm cần search.
* Delta
* Bước 2: Search thưa

1. Ứng với từng bộ ba C, gamma, epsilon, thực hiện cross validation để tìm giá trị lỗi
2. Cập nhật bộ ba mới nếu độ lỗi thấp hơn độ lỗi nhỏ nhất hiện tại.
3. Tăng giá trị của từng tham số theo delta
4. Quay lại bước 1 nếu chưa hết interval của cả 3 tham số.

* Bước 3: Thay đổi giá trị khởi tạo

Để chuẩn bị cho việc search kỹ:

* Gán lại giá trị bắt đầu bằng giá trị lân cận của bộ ba có độ lỗi nhỏ nhất
* Giảm delta
* Bước 4: Thực hiện search kỹ như bước 2.

Để bắt đầu, giá trị delta sẽ được gán bằng 2. Sau khi tìm được bộ ba ưng ý, delta sẽ được giảm xuống 0.25. Ở đây, interval được cho bằng 7.

Ví dụ: Giả sử đã tìm được các giá trị ở bước search thưa: C = 24, gamma = 2-2, epsilon = 2-6. Gán lại giá trị bắt đầu lân cận bộ ba này, sao cho từ vị trí đó đến giá trị hiện tại là 7/2 = 3 interval:

C = 23.25, gamma = 2-1.25, epsilon = 2-5.25.

### Pattern Search

* Đầu tiên, ta khởi tạo ngẫu nhiên một trung tâm của pattern nằm trong miền đã đặt ra.
* Khởi tạo bước tìm kiếm , ngưỡng
* Còn lại, cài đặt giống như lý thuyết ở trên.

# THÍ NGHIỆM

## Dữ liệu

* Nước ngoài:
  + IBM: từ 2/1/2001 đến 31/12/2002; gồm 500 điểm dữ liệu; bộ train chiếm 74.6%; số node đầu vào = 6.
* Việt Nam:
  + FTP: từ 13/12/2006 đến 9/8/2010; gồm 914 điểm dữ liệu; bộ train chiếm 80%; số node đầu vào = 5.

Bộ dữ liệu được scale về (0, 1) theo công thức:

Trong đó:

* : giá đóng cửa tại thời điểm t
* : giá đóng cửa cao nhất
* : giá đóng cửa thấp nhất

## Thí nghiệm số 1: Pattern Search

Mục tiêu của thí nghiệm này là để giúp ta hiểu hơn về Pattern Search và kiểm tính hiệu quả của 2 chiến lược đã được đề xuất.

Ta chạy Pattern Search trên bộ IBM 10 lần.

Kết quả:

*Tìm bộ tham số*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Số vòng lặp | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Lần 6 | Lần 7 | Lần 8 | Lần 9 | Lần 10 |
| 0 | 0.00059307 | 0.001424958 | 0.00095231 | 0.000607141 | 0.001103794 | 0.000573051 | 0.00060145 | 0.014652945 | 0.000615987 | 0.000775641 |
| 1 | 0.000551341 | 0.000937936 | 0.000567516 | 0.000567305 | 0.000699098 | 0.000547736 | 0.000546354 | 0.002896043 | 0.000562909 | 0.000609825 |
| 2 | 0.000547605 | 0.000674795 | 0.000555552 | 0.000539701 | 0.000576233 | 0.000547271 | 0.000546354 | 0.000824624 | 0.000540295 | 0.000547319 |
| 3 | 0.000538797 | 0.000586498 | 0.000551915 | 0.000539701 | 0.000548479 | 0.000541705 | 0.000546354 | 0.000589917 | 0.000533381 | 0.000540254 |
| 4 | 0.000538797 | 0.000551745 | 0.000551915 | 0.000539701 | 0.000534705 | 0.000538905 | 0.000546354 | 0.000550618 | 0.000533381 | 0.000538784 |
| 5 | 0.000538435 | 0.00054658 | 0.000546328 | 0.000539383 | 0.000534705 | 0.000538905 | 0.000546354 | 0.000539772 | 0.000533381 | 0.000538784 |
| 6 | 0.000538435 | 0.000536524 | 0.000546317 | 0.000539383 | 0.000533591 | 0.000538905 |  | 0.000539772 | 0.000533381 | 0.000538784 |
| 7 | 0.000538435 | 0.000536524 | 0.000533771 | 0.000539383 | 0.000533591 | 0.000536176 |  | 0.000539772 | 0.000533381 | 0.000533873 |
| 8 | 0.000533788 | 0.000536524 | 0.000524729 |  | 0.000533188 | 0.000536176 |  | 0.000539772 |  | 0.000533873 |
| 9 | 0.000533788 | 0.000536524 | 0.000524729 |  | 0.000533188 | 0.000536176 |  | 0.000539772 |  | 0.000533873 |
| 10 |  | 0.000535292 | 0.000524729 |  | 0.00053116 |  |  |  |  |  |
| 11 |  | 0.000535292 | 0.000524729 |  | 0.00053116 |  |  |  |  |  |
| **Số lần coss-validation** | **48** | **56** | **60** | **41** | **64** | **53** | **30** | **49** | **38** | **47** |

Table 1

Từ Table 1 ta có được đồ thị:

Figure 1

Nhận xét:

* Ta thấy rằng Pattern Search hoạt động ổn định: cả 10 lần chạy với những điểm khởi tạo khác nhau, thuật toán đều hội tụ về được các giá trị cực trị cục bộ tốt.
* Số lần thực hiện cross-validation ít: trung bình 47 lần.

*Test*

|  |  |  |
| --- | --- | --- |
| **Model** | **MSE cross-validation** | **MSE Test** |
| 1 | 0.000533788 | 2.085338104 |
| 10 | 0.000533873 | 2.129430706 |
| 9 | 0.000533381 | 2.20318881 |
| 5 | 0.00053116 | 2.422241709 |
| 8 | 0.000539772 | 2.553823078 |
| 6 | 0.000536176 | 2.767812765 |
| 4 | 0.000539383 | 2.81204886 |
| 3 | 0.000524729 | 3.088422308 |
| 2 | 0.000535292 | 4.56841591 |
| 7 | 0.000546354 | 7.904750118 |

Table 2

Ghi chú:

* MSE lúc Cross-Validation là làm trên dữ liệu đang scale; còn MSE lúc test là làm trên dữ liệu đã được phục hồi về ban đầu.
* Để dễ theo dõi, cột MSE Test đã được sắp xếp theo thứ tự tăng dần.

Nhận xét:

* Ta nhận thấy MSE Test thấp nhất ứng là 2.085338104 với mô hình số 1, mô hình này có MSE Cross-Validation là 0.000533788. Trong khi đó, con số này lại không phải là con số nhỏ nhất trong MSE Cross-Validation của tất cả các mô hình. Mô hình có MSE Cross-Validation nhỏ nhất là 0.000524729 ứng với mô hình số 3, tuy nhiên lại cho MSE test tương đối cao là 3.088422308.

Ta rút ra một kết luận là:

*Nhìn chung, khi MSE Cross-Validation ứng với các bộ tham số mà có độ chênh lệnh tương đối lớn, thì bộ tham số nào có MSE Cross-Validation càng nhỏ sẽ cho kết quả dự đoán càng tốt. Tuy nhiên, khi các MSE Cross-Validation này không chênh lệch nhau nhiều thì chưa chắc là bộ tham số có MSE nhỏ hơn sẽ cho kết quả dự đoán tốt hơn 🡪* **Chiến lược số 1 thất bại.**

* Ta thấy 7 mô hình có MSE là 2.xx, 1 mô hình là 3.xx, 1 mô hình là 4.xx, 1 mô hình là 7.xx.) Nếu ta chỉ đơn thuần dùng Pattern Search thì rõ ràng ta cảm thấy không ổn định cho lắm (rủi thay, nếu ta chạy Pattern Search mà ra phải mô hình có MSE Test là 7.xx.) Rõ ràng, ở đây áp dụng **chiến lược số 2**, lấy trung bình dự đoán của tất cả mô hình, sẽ ổn định hơn. Để dễ dàng theo dõi, từ đây về sau ta sẽ gọi chiến lược số 2 này là **Bag Pattern Search**. Bad Pattern Search sẽ được cài đặt với 10 lần chạy Pattern Search.

## Thí nghiệm số 2: {SVR Default Value vs Grid Search vs Bag Pattern Search} vs ANN

Mục tiêu của thí nghiệm này là so sánh tính hiệu quả trong bài toán dự đoán giá chứng khoán giữa 2 mô hình: SVR và ANN. Trong SVR ta lại so sánh giữa các phương pháp tìm kiếm tham số: Default Valuse, Grid Search và Bag Pattern Search

Ta làm trên hai bộ dữ liệu là IBM và FPT.

Cấu hình số node tầng đầu vào – tầng ẩn – tầng xuất của ANN:

* IBM: 6-3-1
* FPT: 5-4-1

Kết quả:

### Bộ IBM

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MSE** | **NMSE** | **Số lần cross-validation** |
| ANN | 8.868032623 | 0.282134676 |  |
| SVR – Default | 24.432211166 | 0.511679220 |  |
| SVR – Grid | 2.218166440 | 0.077952288 | 650 |
| SVR – Bag | 2.521256619 | 0.088795204 | 486 |

Table 3

Figure 2

Figure 3

### Bộ FPT

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MSE** | **NMSE** | **Số lần Cross-Validation** |
| ANN | 3.212684864 | 0.10248331 |  |
| SVR – Default | 2440.432789740 | 0.98576468 |  |
| SVR – Grid | 3.128828916 | 0.100699622 | 650 |
| SVR – Bag | 2.707575593 | 0.087144611 | 531 |

Figure 4

Figure 5

Nhận xét:

* SVM (ta không xét SVM với bộ tham số mặc định) luôn cho kết quả tốt hơn ANN. Điều này một lần nữa khẳng định tính hiệu quả của SVM so với ANN như nhiều bài báo đã nói.
* Grid Search cải tiến cho phép cải thiện thời gian tìm kiếm đáng kể và luôn ổn định.
* Bag Pattern Search có số lần cross-validation luôn nhỏ hơn Grid Search nhưng kết quả có thể nói là bằng với Grid Search.

# KẾT LUẬN

Như vậy, ta đã thấy được trong bài toán dự đoán chứng khoán, SVM luôn tỏ ra hiệu quả hơn ANN. Ở đây, ta khảo sát 2 phương pháp chọn bộ tham số cho SVM: Grid Search và Pattern Search. Grid Search được cải tiến với hai bước tìm kiếm thưa và tìm kiếm dày đã giúp cải thiện thời gian tìm kiếm đáng kể. Bản chất của Grid Search là vét cạn; nó giúp ta cảm thấy an tâm về tính ổn định tuy nhiên chi phí tìm kiếm lớn. Pattern Search với kỹ thuật “bag” đã chứng tỏ được sức mạnh của mình. Đây đáng là một thuật toán để ta nghiên cứu thêm.

# CÔNG VIỆC TƯƠNG LAI

* Việc chọn tham số:
  + Nghiên cứu thêm về việc chọn miền giá trị cho các tham số.
  + Ở đây, ta có thể cải tiến thêm Pattern Search để tăng tốc hơn nữa.
* Việc tiền xử lý dữ liệu.
* Việc huấn luyện: khảo sát cách huấn luyện cải tiến cho SVM.
  + Cách huấn luyện truyền thống là ta chia thành 2 bộ train, test; sau đó dùng bộ train để huấn luyện mô hình; sau khi có mô hình ta sẽ dùng mô hình đó để dự đoán cho tất cả các điểm trong bộ test. Điều này có nghĩa là ta đã giả sử mối quan hệ giữa giá của ngày X và các ngày trước đó là cố định.
  + Cách huấn luyện cải tiến: để dự đoán giá của một ngày, ta luôn phải tiến hành train. Kích thước bộ train này tương đối nhỏ. Cách làm này rất hứa hẹn bởi những lý do sau:
    - Bên ANN đã có bài báo cho kết quả rất tốt.
    - Với kích thước bộ train nhỏ, ta có thể cải thiện đáng kể tốc độ thực thi của cross-validation và do đó, cải thiện tốc độ thực thi của các thuật toán tìm bộ tham số (ta có thể áp dụng Grid Search.)
    - Cách train này cũng giải quyết luôn vụ online vì rằng để dự đoán giá của một ngày mới, ta luôn phải tiến hành train lại từ đầu.
* Nghiên cứu thêm về việc đưa thêm các input khác ngoài giá đóng cửa.

# TÀI LIỆU THAM KHẢO

[1] Michinari Momma, Kristin P. Bennett, *A Pattern Search Method for Model Selection of Support Vector Regression*, In Proceedings of SDM'2002

[2] J. Dennis, V. Torczon, *Derivative-free pattern search methods for mul-*

*tidisciplinary design problems*, paper AIAA-94-4349 in Proceedings of the 5th

AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and

Optimization,Panama City, FL, Sept. 7-9, (1994), pp. 922-932

[3] http://www.dtreg.com/svm.htm