Học Máy
(IT 4862)

Nguyễn Nhật Quang

quangnn-fit@mail.hut.edu.vn

Trường Đại học Bách Khoa Hà Nội

Viện Công nghệ thông tin và truyền thông

Năm học 2011-2012

Nội dung môn học:

- Giới thiệu chung
- Đánh giá hiệu năng hệ thống học máy
- Các phương pháp học dựa trên xác suất
- **Các phương pháp học có giám sát**
  - **Học quy nạp luật (Rule induction)**
- Các phương pháp học không giám sát
- Lọc cộng tác
- Học tăng cường

Quy nạp luật – Giới thiệu (1)

- Để học một tập các luật (IF-THEN) cho bài toán phân loại
  - Phù hợp khi hàm mục tiêu (phân loại) có thể được biểu diễn bằng một tập các luật (IF-THEN)

Hàm mục tiêu: $h \equiv \{Luật_1, Luật_2, ..., Luật_m\}$

$Luật_j \equiv$ IF ($Điều\text{-}kiện_{j1} \wedge Điều\text{-}kiện_{j2} \wedge ... \wedge Điều\text{-}kiện_{jn}$) THEN $Kết\ luận_j$

- Các luật (IF-THEN)
  - Một phương pháp phổ biến để biểu diễn tri thức
  - ***Phương pháp biểu diễn dễ hiểu nhất đối với người dùng***

Quy nạp luật – Giới thiệu (2)

- Nhắc lại: Học cây quyết định (Decision tree learning) cũng có cho phép học một tập các luật logic định đề
  - Bước 1: Học cây quyết định
  - Bước 2: Biểu diễn mỗi đường đi trong cây (từ nút gốc đến nút lá) thành một luật tương ứng
- Học một tập các luật
  - Học cây quyết định: Tập các luật logic định đề được học đồng thời
  - Học quy nạp luật: Tập các luật logic định đề/vị từ được học tuần tự (từng luật một)
- Các giải thuật khác nhau để học các kiểu luật khác nhau
  - Các luật logic định đề (chỉ sử dụng các *ký hiệu hằng*)
  - Các luật logic vị từ (sử dụng cả các *ký hiệu biến* và các *ký hiệu vị từ*) – khả năng diễn đạt cao hơn

Quy nạp luật – Ví dụ (1)

- Học một tập các luật ***logic định đề***

  Vd: Hàm mục tiêu (phân loại) `Buy_Computer` được biểu diễn bởi:

  IF (`Age=Old` ∧ `Student=No`) THEN `Buy_Computer=No`
  IF (`Student=Yes`) THEN `Buy_Computer=Yes`
  IF (`Age=Medium` ∧ `Income=High`) THEN `Buy_Computer=Yes`

- Học một tập các luật ***logic vị từ***

  Vd: Hàm mục tiêu (khái niệm) `Ancestor` được biểu diễn bởi:

  IF `Parent(x,y)` THEN `Ancestor(x,y)`
  IF `Parent(x,y)` ∧ `Ancestor(y,z)` THEN `Ancestor(x,z)`

  (`Parent(x,y)` là một *vị từ* thể hiện `y` là cha/mẹ của `x`)

Quy nạp luật – Ví dụ (2)

- Luật: IF (`Age=Old` ∧ `Student=No`) THEN `Buy_Computer=No`
- Những ví dụ nào được phân loại chính xác bởi luật trên?

	Rec. ID	Age	Income	Student	Credit_Rating	Buy_Computer
	1	Young	High	No	Fair	No
	2	Medium	High	No	Fair	Yes
X	3	Old	Medium	No	Fair	Yes
	4	Old	Low	Yes	Excellent	No
	5	Medium	Low	Yes	Excellent	Yes
	6	Young	Medium	No	Fair	No
	7	Old	Medium	Yes	Fair	Yes
	8	Medium	High	Yes	Fair	Yes
√	9	Old	Medium	No	Excellent	No

Quy nạp luật định đề – Huấn luyện

- Học một tập các luật theo chiến lược bao phủ gia tăng (incremental covering strategy)
  - Bước 1: Học một luật
  - Bước 2: Loại khỏi tập huấn luyện những ví dụ học nào được phân loại chính xác bởi luật vừa học được

  → Lặp lại 2 bước này để học luật khác (sử dụng tập huấn luyện sau loại bỏ)
- Quá trình học
  - Học các luật tuần tự (bao phủ gia tăng đối với tập huấn luyện)
  - Quá trình học tiếp tục (lặp) tùy theo mong muốn tập luật học được bao phủ đến mức độ nào (hoặc toàn bộ) tập huấn luyện
- Tập luật học được sẽ được sắp thứ tự theo một tiêu chí đánh giá hiệu năng (vd: độ chính xác phân loại)

  →Các luật sẽ được kiểm tra theo đúng trật tự này, khi thực hiện phân loại một ví dụ trong tương lai

Sequential-Covering(TargetAttribute, Attributes, TrainingSet, Threshold)

LearnedRules ← {}

Rule ← `LEARN-ONE-RULE`(TargetAttribute, Attributes, TrainingSet)

while `PERFORMANCE`(Rule, TrainingSet) > Threshold

LearnedRules ← LearnedRules ∪ Rule

TrainingSet ← TrainingSet \ {các ví dụ được phân loại chính xác bởi Rule}

Rule ← `LEARN-ONE-RULE`(TargetAttribute, Attributes, TrainingSet)

end while

Sắp xếp LearnedRules theo đánh giá `PERFORMANCE` đối với tập TrainingSet

return {LearnedRules, DefaultRule}

- DefaultRule: IF <*null*> THEN (Giá trị phổ biến nhất của `TargetAttribute` trong tập `TrainingSet`)
- `LEARN-ONE-RULE`: Hàm thực hiện học một luật đối với tập `TrainingSet`
- `PERFORMANCE`: Hàm đánh giá chất lượng (hiệu quả) của một luật học được

Quy nạp luật định đề – Phân loại

- Đối với một ví dụ cần phân loại:
  - Các luật đã học được sẽ được kiểm tra (khai thác) tuần tự theo đúng trật tự thu được trong giai đoạn huấn luyện
  - Luật tìm được đầu tiên phù hợp với ví dụ (các điều kiện trong mệnh đề *IF* của luật phù hợp với ví dụ) sẽ được sử dụng để phân loại ví dụ này
    → Ví dụ được phân loại dựa trên kết luận (nhãn lớp) trong mệnh đề *THEN* của luật
  - Nếu không có bất kỳ luật nào phù hợp với ví dụ, thì ví dụ này được phân loại bởi luật mặc định (DefaultRule)
    → DefaultRule: Ví dụ được phân vào lớp chiếm số đông trong tập huấn luyện

Chiến lược bao phủ gia tăng – Các vấn đề

- Chuyển bài toán (phức tạp hơn) học một tập các luật thành một chuỗi các bài toán (đơn giản hơn), mỗi bài toán học một luật
  - Sau khi học được một luật, thì tất cả các ví dụ học bị bao phủ (được phân loại chính xác) bởi luật đó sẽ được loại khỏi tập huấn luyện
  - Mỗi luật được học một cách độc lập với các luật khác – Vấn đề: Nếu các luật có sự phụ thuộc (tác động) lẫn nhau?
- Để tìm một chuỗi các luật, thực hiện chiến lược tìm kiếm tham lam (greedy search) mà không có quay lui xét lại (without backtracking)
  - Không đảm bảo tìm được một *tập nhỏ nhất* các luật
  - Không đảm bảo tìm được một *tập tối ưu (vd: về khía cạnh phân loại chính xác)* các luật

Học một luật

- Các yêu cầu đối với hàm `LEARN-ONE-RULE`
  - Trả về một luật bao phủ (phân loại được) một số lượng lớn các ví dụ học
    → Các ví dụ học này phù hợp với các điều kiện của luật học được
  - Độ chính xác cao
    → Các phân loại bởi luật học được cần phải chính xác
  - Không cần thiết phải có độ bao phủ quá cao
    → Không cần thiết phải bao phủ (phân loại được) tất cả các ví dụ học
- Giải pháp: Tìm kiếm (học) luật từ-tổng-quát-đến-cụ-thể
  - Bắt đầu với luật tổng quát nhất (không có điều kiện nào)
  - Bổ sung vào luật một điều kiện (đối với một thuộc tính), ưu tiên điều kiện giúp cải thiện tối đa hiệu năng của luật đối với các ví dụ học
  - Lặp lại bước trên để bổ sung thêm một điều kiện khác vào luật

LEARN-ONE-RULE_1(TargetAttribute, Attributes, TrainingSet)

Best_Pre-cond ← Ø

while (Attributes ≠ Ø) and (TrainingSet ≠ Ø)

// 1. Sinh ra tập ứng cử của các điều kiện có thể bổ sung (thêm vào) mệnh đề IF của luật

All_constraints ← Tập các điều kiện có dạng (A=v), trong đó A ∈ Attributes và v là một giá trị của A xảy ra trong TrainingSet

Candidate_Pre-conds ← for each c ∈ All_constraints, Tạo một mệnh đề điều kiện (Best_Pre-cond ∧ c)

// 2. Cập nhật mệnh đề điều kiện tốt nhất ***Best_Pre-cond***

Best_Pre-cond ← $\text{argmax}_{PC \in \text{Candidate\_Pre-conds}}$ {PERFORMANCE(PC, TrainingSet, TargetAttribute)}

Attributes ← Attributes \ {Thuộc tính A^+ tương ứng với điều kiện c^+ vừa mới được bổ sung vào Best_Pre-cond}

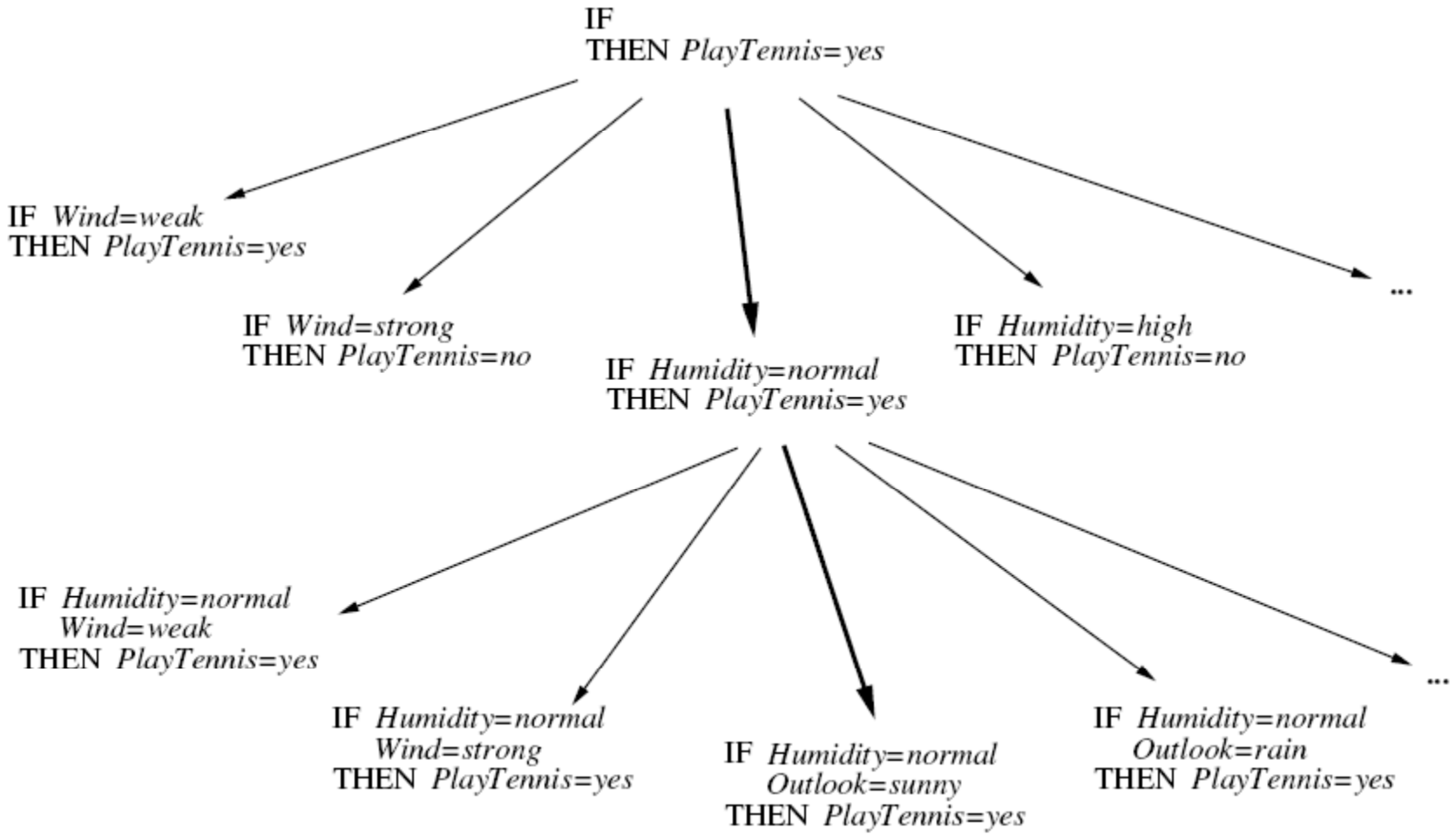
(*) TrainingSet ← {Các ví dụ học phù hợp với Best_Pre-cond}

end while

return (Luật: IF Best_Pre-cond THEN prediction)

(prediction giá trị (nhãn lớp) phổ biến nhất của TargetAttribute trong số các ví dụ học, TrainingSet (trước Bước (*)), phù hợp với Best_Pre-cond

LEARN-ONE-RULE_1



[Mitchell, 1997]

LEARN-ONE-RULE_1 – Các vấn đề

- LEARN-ONE-RULE_1 có chiến lược tìm kiếm giống như ID3
  - Học (phát triển) luật/cây bằng cách bổ sung dần dần các điều kiện đối với các thuộc tính
  - Dừng, khi luật/cây học được đạt tới mức hiệu năng chấp nhận được
- Nhưng có sự khác nhau:
  - Tại mỗi bước tìm kiếm, LEARN-ONE-RULE_1 chỉ đi theo một hướng cụ thể hóa điều kiện (A=v*) giúp đem lại hiệu năng cao nhất
  - ID3 phát triển một cây con gồm tất cả các giá trị có thể v_i của A
- LEARN-ONE-RULE_1 thực hiện tìm kiếm theo chiều sâu tham lam (greedy depth-first), không xét lại (without backtracking)

  → Tại mỗi bước tìm kiếm, điều kiện được bổ sung (A=v*) có thể không tối ưu
- Giải pháp khắc phục: Thực hiện tìm kiếm chùm (beam search)

LEARN-ONE-RULE_2 – Beam search

- Tại mỗi bước tìm kiếm, lưu giữ một tập gồm *k* (thay vì chỉ 1) mệnh đề điều kiện (IF) tốt nhất
- Tại một bước tìm kiếm:
  - Các cụ thể hóa (bổ sung thêm 1 điều kiện) được sinh ra cho mỗi trong số *k* mệnh đề điều kiện tốt nhất
  - Chỉ giữ lại *k* các cụ thể hóa có hiệu năng cao nhất
- Quá trình tìm kiếm học các luật theo hướng gia tăng cụ thể hóa (bổ sung dần các điều kiện)
  - Cho đến khi thu được luật cụ thể hóa tối đa (phần mệnh đề điều kiện liên quan đến tất cả các thuộc tính)
- Mặc dù việc sử dụng Beam search (trong việc học một luật) giúp giảm nguy cơ học được một luật không tối ưu; Nhưng việc sử dụng Greedy search (trong việc học một tập các luật) vẫn có thể dẫn đến học được một tập không tối ưu của các luật

LEARN-ONE-RULE_2(`TargetAttribute`, `Attributes`, `TrainingSet`)

`Best_Pre-cond` ← Ø // mệnh đề điều kiện tổng quát nhất

`Candidate_Pre-conds` ← {`Best_Pre-cond`}

while (`Attributes` ≠ Ø)

// 1. Sinh ra tập ứng cử của các điều kiện có thể bổ sung (thêm vào) mệnh đề IF của các luật

`All_constraints` ← Tập các điều kiện có dạng (`A`=`v`), trong đó

`A` ∈ `Attributes` và `v` là một giá trị của `A`

xảy ra trong `TrainingSet`

`New_candidate_Pre-conds` ←

for each `pc` ∈ `Candidate_Pre-conds`,

for each `c` ∈ `All_constraints`,

Tạo một mệnh đề điều kiện `(pc ∧ c)`

Loại khỏi tập `New_candidate_Pre-conds` bất kỳ mệnh đề điều kiện nào trùng lặp hoặc mâu thuẫn // vd: $(A_i=v_j) \Lambda (A_i=v_k)$

. . .

LEARN-ONE-RULE_2(`TargetAttribute`, `Attributes`, `TrainingSet`)

. . .

// 2. Xác định lại mệnh đề điều kiện tốt nhất

for each `pc` ∈ `New_candidate_Pre-conds`

if (`PERFORMANCE(pc, TrainingSet, TargetAttribute)` >

`PERFORMANCE(Best_Pre-cond, TrainingSet, TargetAttribute)`)

then `Best_Pre-cond` ← `pc`

// 3. Xác định lại tập các mệnh đề điều kiện hiện tại (Giữ lại tối đa k phần tử!)

`Candidate_Pre-conds` ← Tập gồm `k` phần tử tốt nhất trong tập `New_candidate_Pre-conds`, dựa trên đánh giá `PERFORMANCE`

end while

return (Luật: IF `Best_Pre-cond` THEN `prediction`)

(`prediction` giá trị (nhãn lớp) phổ biến nhất của `TargetAttribute` trong số các ví dụ học (`TrainingSet`) phù hợp với `Best_Pre-cond`

Đánh giá hiệu quả của một luật (1)

- Hàm `PERFORMANCE(.)` được sử dụng trong các giải thuật nêu trên
- Đánh giá dựa trên tỷ lệ phân loại chính xác
  - `D_train`^{R}: Tập các ví dụ học phù hợp với mệnh đề điều kiện (*IF*) của luật `R`
  - `n`: Kích thước của tập `D_train`^{R}
  - n_c: Số lượng các ví dụ trong tập `D_train`^{R} được phân loại chính xác bởi `R`

$$PERFORMANCE(R, D\_train^R) = \frac{n_c}{n}$$

Đánh giá hiệu quả của một luật (2)

- Đánh giá dựa trên ước lượng (*m*-estimate) về độ chính xác
  - p: Xác suất trước (tiên nghiệm) của việc một ví dụ, được lấy ngẫu nhiên từ tập dữ liệu, phân lớp được bằng luật R
    - →p là độ chính xác được giả định trước
  - m: Giá trị trọng số chỉ định mức độ ảnh hưởng của xác suất trước p đối với đánh giá hiệu năng của luật
    - Nếu m=0, thì phương pháp *m*-estimate trở thành phương pháp đánh giá dựa trên tỷ lệ phân loại chính xác

$$PERFORMANCE(R, D\_train^R) = \frac{n_c + mp}{n + m}$$

Đánh giá hiệu quả của một luật (3)

- Đánh giá dựa trên giá trị Entropy
  - `c`: Số lượng các giá trị của thuộc tính phân loại (= Số lượng nhãn lớp)
  - p_i: Tỷ lệ số lượng các ví dụ trong tập D_train^R được phân (gán) vào lớp thứ *i*

$$PERFORMANCE(R, D\_train^R) = -Entropy(D\_train^R)$$

$$= \sum_{i=1}^{c} p_i . \log_2 p_i$$

Các luật logic vị từ

- Các định nghĩa hình thức trong logic vị từ
  - Hằng (như trong logic định đề) – Vd: `John`
  - Biến – Vd: `x`
  - Vị từ – Vd : `Male`(John)
  - Hàm – Vd: `age`(John)
  - Term: là một hằng, hoặc một biến, hoặc là một hàm đối với term khác – Vd: `age`(`x`)
  - Literal: là một vị từ (hoặc phủ định của vị từ) đối với một tập các terms – Vd: `Greater_Than`(`age`(`John`),`20`), ¬`Male`(`x`)
- Một luật logic vị từ là một mệnh đề dạng chuẩn Horn
  - `H` và L_i(`i`=1..`n`) là các literals
  - Luật logic vị từ: IF $L_1 \wedge ... \wedge L_n$ THEN `H`
  - Dạng chuẩn Horn tương ứng: $H \vee \neg L_1 \vee ... \vee \neg L_n$

Học các luật logic vị từ – Giải thuật FOIL

- Để học một tập các luật logic vị từ (có chứa các biến)
  - Các luật logic vị từ *có khả năng diễn đạt cao hơn nhiều* so với các luật logic định đề
- Giải thuật FOIL ("first-order inductive logic")
  - Bắt đầu với một tập rỗng (các luật học được)
  - Học một luật mới, và sau đó bổ sung vào tập các luật học được
    - Tuần tự bổ sung các literals *kết hợp (conjunctive)* vào trong luật mới, cho đến khi không có ví dụ sai nào (negative instance) được phân loại (phù hợp) với luật mới
      → Một ví dụ sai là ví dụ thỏa mãn mệnh đề điều kiện (IF) của luật, nhưng có giá trị sai đối với vị từ (trong mệnh đềTHEN)
    - Khi xét các literals ứng cử viên, cần lựa chọn literal có giá trị đánh giá `Foil_Gain` lớn nhất
  - Loại bỏ các ví dụ đúng (positive instances) đối với luật mới
  - Lặp lại để học một luật khác...Cho đến khi không còn ví dụ đúng (positive instances) nào nữa

FOIL(`TargetPredicate`, `Predicates`, `TrainingSet`)

`PosSet` ← The instances in `TrainingSet` for which `TargetPredicate` is true

`NegSet` ← The instances in `TrainigSet` for which `TargetPredicate` is false

`Learned_rules` ← Ø

while (`PosSet` ≠ Ø)

// Learn a new rule

`R` ← The most general rule (i.e., the one that predicts `TargetPredicate` with no precondition)

`NegSet_R` ← `NegSet`

while (`NegSet_R` ≠ Ø)

// Add a new literal to specialize R

`Candidate_literals` ← Generate candidate new literals for `R`, based on `Predicates`

. . .

FOIL(`Target_predicate`, `Predicates`, `Examples`)

...

`Best_literal` ← argmax$_{L \in \texttt{Candidate_literals}}$ `Foil_Gain`(`L`,`R`)

Add `Best_literal` to the preconditions of `R`

`NegSet_R` ← {instances in `NegSet_R` that match the precondtions of `R`}

end while

`Learned_rules` ← {`Learned_rules`, `R`}

`PosSet` ← `PosSet` \ {instances in `PosSet` covered by `R`}

end while

return `Learned_rules`

Học luật logic vị từ – Phân loại

Đối với 1 ví dụ cần phân loại:

- Xét (kiểm tra) tuần tự các luật đã học được theo đúng thứ tự của chúng thu được sau quá trình huấn luyện
- Luật đầu tiên tìm được thỏa mãn ví dụ (là luật có mệnh đề điều kiện *IF* thỏa mãn/phù hợp với ví dụ) sẽ được dùng để phân loại
  - Ví dụ được phân loại bởi mệnh đề *THEN* của luật đó
- Nếu không có luật nào phù hợp với ví dụ, thì ví dụ được phân loại bởi luật mặc định (default rule)
  - Ví dụ được gán nhãn lớp bởi giá trị (nhãn lớp) phổ biến nhất trong tập huấn luyện

Sản sinh các điều kiện cụ thể hóa trong FOIL

- Luật cần được cụ thể hóa: $L_1 \wedge ... \wedge L_n \rightarrow P(x_1,...,x_k)$
  - $L_1,...,L_n$: Các literals tạo nên mệnh đề điều kiện của luật
  - $P(x_1,...,x_k)$: Literal tạo nên mệnh đề kết luận của luật
- Sinh ra các điều kiện cụ thể hóa của luật bằng cách bổ sung một literal mới L_{n+1} , có thể là:
  - $x_i=x_j$, $x_i=c$, $x_i>x_j$, $x_i \geq x_j$; trong đó x_i và x_j là các biến đã tồn tại trong luật
  - $Q(y_1,...,y_m)$; trong đó Q là một vị từ thuộc tập $Predicates$, và y_i là các biến và ít nhất một trong số các biến này (y_i) phải đã tồn tại trong luật
  - Dạng phủ định (negation) của 1 trong 2 dạng literals nêu trên

FOIL –Định hướng quá trình tìm kiếm

- Literal nào (trong số các điều kiện cụ thể hóa - candidate specialization literals) nên được bổ sung thêm vào luật?
- Giải thuật FOIL sử dụng hàm `Foil_Gain` để đánh giá hiệu quả của việc bổ sung thêm một literal vào luật
- `Foil_Gain` ưu tiên việc bổ sung một literal cho phép mang lại một luật mới có *nhiều ràng buộc đúng (positive bindings) và ít ràng buộc sai (negative bindings) hơn* luật ban đầu (trước khi bổ sung literal)
  - Một *ràng buộc biến (variable binding)* là một phép gán (thay thế) mỗi biến bằng một hằng số (giá trị)
  - Một *ràng buộc đúng (positive binding)* là một ràng buộc biến làm cho (mệnh đề *THEN* của) luật đó là đúng

Hàm đánh giá Foil_Gain

$$Foil\_Gain(L,R) = t.\left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right)$$

- `R`: Luật ban đầu (trước khi bổ sung literal `L`)
- `L`: Literal ứng cử viên để bổ sung vào luật `R`
- p_0: Số lượng các ràng buộc đúng (positive bindings) của luật `R`
- n_0: Số lượng các ràng buộc sai (negative bindings) của luật `R`
- p_1: Số lượng các ràng buộc đúng của luật mới `(R+L)`
- n_1: Số lượng các ràng buộc sai của luật mới `(R+L)`
- `t`: Số lượng các ràng buộc đúng của luật ban đầu `R` cũng là các ràng buộc đúng của luật mới `(R+L)` (Số lượng các ràng buộc biến làm cho cả 2 luật `R` và `(R+L)` cùng đúng)

Giải thuật FOIL – Các vấn đề

- Các luật học được bởi giải thuật FOIL bị hạn chế (giới hạn) hơn là các mệnh đề Horn tổng quát
  - → Các literals không được phép chứa các ký hiệu hàm. (Lý do: Để giảm sự phức tạp của việc tìm kiếm trong không gian các khả năng)
- Các luật học được bởi giải thuật FOIL có khả năng diễn đạt cao hơn (more expressive than) các mệnh đề Horn tổng quát
  - → Các literals xuất hiện trong mệnh đề điều kiện của luật (i.e., $L_1,...,L_n$) có thể (được phép) ở dạng phủ định

Tài liệu tham khảo

- T. M. Mitchell. ***Machine Learning***. McGraw-Hill, 1997.