5.4 Version Spaces

- Mitchell [1977,1978] เสนอวิธีการที่เรียกว่า version spaces
- เรียน description ที่อธิบายตัวอย่างบวกและไม่อธิบายตัวอย่างลบ
- รูปที่ 5.4.1 คือ ตัวอย่างบวกของการเรียน concept 'Car' ซึ่งแสดงโดย frame

Car023	
origin :	Japan
manufacturer :	Honda
color :	Blue
decade :	1970
type :	Economy

รูปที่ 5.4.1 ตัวอย่างบวกของ concept 'Car'

- สมมุติว่าแต่ละ slot ของ frame เป็นค่าที่แสดงในรูปที่ 5.4.2

origin	∈	{ Japan, USA, Britain, Germany, Italy }
manufacturer	∈	{ Honda, Toyota, Ford, Chrysler, Jaguar, BMW, Fiat }
color	∈	{ Bule, Green, Red, White }
decade	∈	{ 1950, 1960, 1970, 1980, 1990, 2000 }
type	∈	{ Economy, Luxury, Sports }

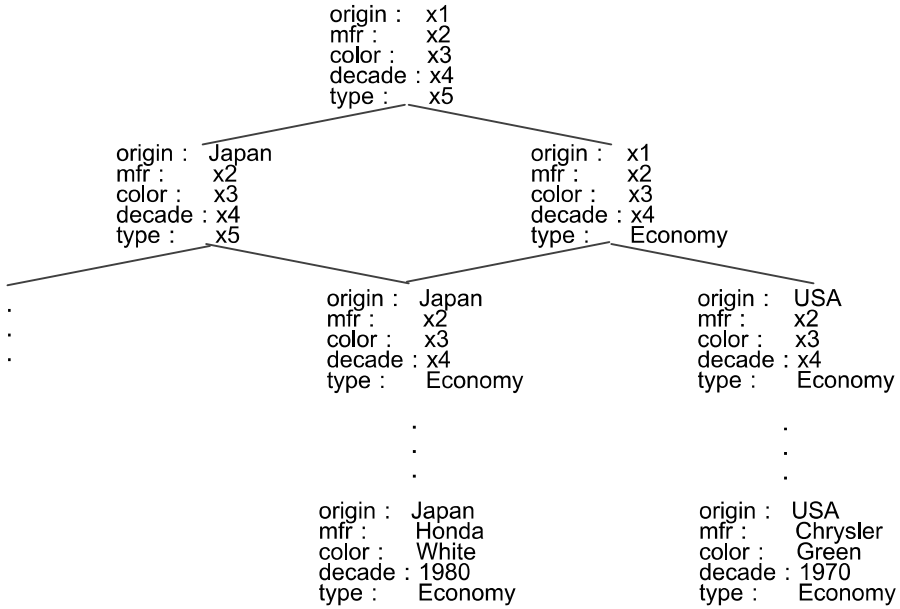
รูปที่ 5.4.2 ภาษาที่ใช้แสดง

- Concept description แสดงในรูปของ slots และ slot values เช่น concept ของ "Japanese economy car" แสดงได้ดังรูปที่ 5.4.3

origin :	Japan
manufacturer :	x1
color :	x2
decade :	x3
type :	Economy

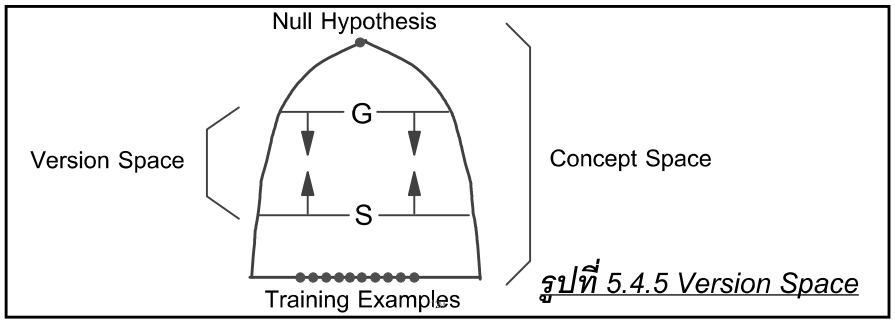
รูปที่ 5.4.3 Concept "Japanese Economy Car"

- ปัญหาของการเรียนรู้คือ " กำหนด ภาษาที่ใช้แสดงให้ (เช่นในรูปที่ 5.4.2) และตัวอย่างบวก ตัวอย่างลบให้ (เช่นในรูปที่ 5.4.1) จงหา concept description ที่สอดคล้องกับตัวอย่าง (อธิบายตัวอย่างบวก และ ไม่อธิบายตัวอย่างลบ) "
- Concept space คือ space ที่มีสมาชิกเป็น descriptions โดยที่สมาชิกเหล่านั้นมี partial ordering (ตัวที่ general กว่าจะอยู่เหนือตัวที่ specific กว่า) ดังรูปที่ 5.4.4
  - ตัวที่อยู่บนสุด คือ ตัวที่ general มากที่สุด
  - ตัวที่อยู่ล่างสุด คือ ตัวที่ specific มากที่สุด (ตัวอย่างบวก)
  - target concept description จะอยู่ระหว่างบนสุดกับล่างสุด
- ค้นหา target concept description โดยสร้าง hypothesis ที่เป็นเซ็ตย่อยของ concept space และเรียกเซ็ตย่อยนี้ว่า version space



รูปที่ 5.4.4 Concept Space

- Version space เป็น space ที่มีสมาชิกเป็น description ซึ่งสอดคล้องกับตัวอย่าง
- การแสดง version space แสดงโดยเซ็ตย่อย 2 เซ็ต คือ G และ S
  - เซ็ต G ประกอบด้วย most *general* descriptions ที่สอดคล้องกับตัวอย่างที่เคยพบมาทั้งหมด
  - เซ็ต S ประกอบด้วย most *specific* descriptions ที่สอดคล้องกับตัวอย่างที่เคยพบมาทั้งหมด
  - version space จะอยู่ระหว่าง G และ S (ดูรูปที่ 5.4.5)



รูปที่ 5.4.5 Version Space

- หลักการของ version space คือ ทุกครั้งที่เราได้รับตัวอย่างบวกตัวใหม่ เราจะทำให้ S general มากขึ้น และทุกครั้งที่ได้รับตัวอย่างลบ เราจะทำให้ G specific มากขึ้น จนในที่สุด S และ G ลู่เข้าสู่ค่าเดียวกัน ซึ่งเรียกว่า target concept description

algorithm : Candidate Elimination

1. G := {null description}
2. S := {first positive example}
3. Accept a new example E
   IF E is positive THEN
      Remove from G any descriptions that do not cover the example.
      Update S to contain the most specific set of descriptions in the version space that cover the example and the current elements of S.
   END ELSE IF E is negative THEN
      Remove from S any descriptions that cover the example.
      Update G to contain the most genereal set of desciptions in the version space that do not cover the example.
4. IF S and G are both singleton sets and S = G THEN Output the element
   ELSE IF S and G are both singleton sets and S<>G THEN examples were inconsistent.
   ELSE goto 3.

ตัวอย่างของการเรียน concept "Japanese economy car"

origin :	Japan	origin :	Japan
mfr :	Honda	mfr :	Toyota
color :	Blue	color :	Green
decade :	1980	decade :	1970
type :	Economy	type :	Sports
	(+)		(-)

origin :	Japan	origin :	USA
mfr :	Toyota	mfr :	Chrysler
color :	Blue	color :	Red
decade :	1990	decade :	1980
type :	Econamy	type :	Economy
	(+)		(-)

origin :	Japan
mfr :	Honda
color :	White
decade :	1980
type :	Economy
	(+)

- จากตัวอย่างบวก 3 ตัว และ ตัวอย่างลบ 2 ตัวด้านบน เราเริ่มด้วยการสร้าง G และ S จากตัวอย่างแรก

  G = {(x1,x2,x3,x4,x5)}

  S = {(Japan,Honda,Blue,1980,Economy)}

  โดยที่ (x1,x2,x3,x4,x5) เป็นค่าของ slot ที่ 1, 2, 3, 4, 5 ตามลำดับ

- ตัวอย่างที่ 2 เป็นตัวอย่างลบ ดังนั้นเรา specialize G เพื่อไม่ให้ version space อธิบายตัวอย่างลบนี้ โดยการเปลี่ยนตัวแปรให้เป็นค่าคงที่

  G = {(x1,Honda,x3,x4,x5), (x1,x2,Blue,x4,x5),
  (x1,x2,x3,1980,x5), (x1,x2,x3,x4,Economy)}

  ส่วน S ไม่เปลี่ยนแปลง = {(Japan,Honda,Blue,1980,Economy)}
- ตัวอย่างที่ 3 เป็นบวก = (Japan,Toyota,Blue,1990,Economy)

  เรากำจัด description ใน G ที่ไม่สอดคล้องกับตัวอย่างนี้

  G = {(x1,x2,Blue,x4,x5), (x1,x2,x3,x4,Economy)}

  และ generalize S ให้รวมตัวอย่างนี้

  S = {(Japan,x2,Blue,x4,Economy)}

  ที่จุดนี้เราได้ version space ที่แสดง

  "Japanese blue economy car", "blue car" หรือ "Economy car"

- ตัวอย่างที่ 4 เป็นลบ = (USA,Chrysler,Red,1980,Economy)

  G = {(x1,x2,Blue,x4,x5), (x1,x2,Blue,x4,Economy),
  (Japan,x2,x3,x4,Economy)}

  S = {(Japan,x2,Blue,x4,Economy)}
- ตัวอย่างที่ 5 เป็นบวก = (Japan,Honda,White,1980,Economy)

  G = {(Japan,x2,x3,x4,Economy)}

  S = {(Japan,x2,x3,x4,Economy)}

ที่จุดนี้ ได้คำตอบ S=G แสดง "Japanese economy car"

<u>สิ่งสำคัญเกี่ยวกับ version space</u>

- algorithm เป็น least-commitment algorithm -- ในแต่ละขั้นตอน version space จะถูก pruned ให้เป็น space ที่เล็กลงน้อยที่สุดเท่าที่เป็นไปได้ ดังนั้น ถึงแม้ว่าตัวอย่างบวกทุกตัวเป็น Japanese cars ก็ตาม algorithm ก็จะไม่ตัดความน่าจะเป็นที่ concept อาจจะรวม car อื่นๆทิ้ง จนกระทั่งพบตัวอย่างลบ

- กระบวนการค้นหาเป็นแบบ exhaustive breadth-first search : เห็นได้จากการ update เซ็ต G ดังนั้นทำให้ algorithm มีประสิทธิ-ภาพต่ำในกรณีที่ space ใหญ่มากๆ ซึ่งอาจทำให้ดีขึ้นโดยใช้ heuristic เข้าช่วยในการค้นหา
- เซ็ต S ประกอบด้วยสมาชิกเพียงตัวเดียว เพราะว่า ตัวอย่างบวก 2 ตัวใดๆมี generalization เพียงหนึ่งเดียว ดังนั้น version space จึงไม่สามารถเรียน disjunctive concept (concept ที่อยู่ในรูปของ or เช่น "Japanese econamy car or Japanese sport car"
- จุดอ่อนอีกอย่างของ version space คือ ไม่สามารถจัดการกับ noisy example (example ที่มีข้อมูลบางส่วนผิดพลาด) เช่น ถ้า example ตัวที่ 3 (Japan Toyota Blue 1990 Economy) เราให้ class ผิดเป็น (-) algorithm จะไม่สามารถเรียน concept "Japanese economy car" ได้ถูกต้อง

5.5 Identification Tree Learning

- Identification-tree learning เป็นวิธีการเรียนรู้ที่ใช้มากที่สุดใน machine learning
- ระบบเรียนรู้ประเภทนี้ เรียน identification tree จากตัวอย่างของหลายๆ class ซึ่ง tree ที่ได้ใช้ classify ตัวอย่าง
- ตัวอย่างของปัญหา : ต้องการหาว่าอะไรคือปัจจัยที่ทำให้คนที่ไปผึ่งแดดตามชายทะเล บางคนก็จะมีผิวเปลี่ยนเป็นสี tan แต่บางคนต้องได้รับความทรมานจากผิวไหม้ โดยข้อมูลที่สังเกตุได้มี ความแตกต่างของ สีผม ส่วนสูง น้ำหนัก ของผู้ที่ไปผึ่งแดด และบางคนก็ใช้โลชั่น บางคนก็ไม่ใช้

class

attribute →

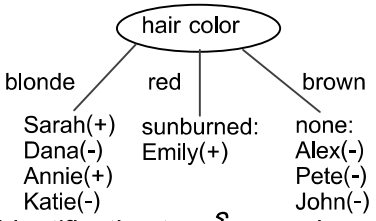
value

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

<u>ตารางแสดงข้อมูลที่สังเกตุได้</u>

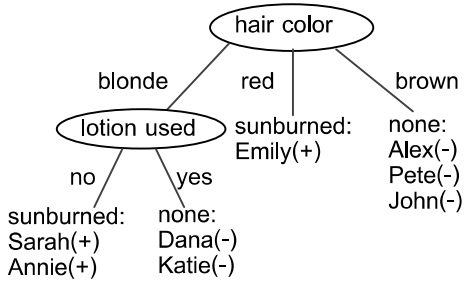
- ถ้าเราต้องการรู้ว่าใครบางคนถ้าไปผึ่งแดดแล้ว ผิวจะไหม้ (sunburned) หรือไม่ ก็อาจจะทำได้โดยเทียบข้อมูลของคนนั้นกับตารางด้านบน ซึ่งความน่าจะเป็นที่ข้อมูลจะอยู่ในตารางเป็น 8/(3*3*3*2) = 15% จะเห็นว่าวิธีนี้ไม่ใช่วิธีที่ดี

- identification tree คือ decision tree ซึ่ง
  - แต่ละ node ใน tree แสดง attribute
  - link ที่ต่อกับ node แสดง value
  - leaf ของ tree แสดง class
- การสร้าง tree ทำโดยสร้าง node ขึ้นทีละ node เพื่อ test คุณสมบัติของตัวอย่าง (ข้อมูล) แล้วแยกตัวอย่างตาม value ของตัวอย่าง ทำจนกระทั่งตัวอย่างในแต่ละ leaf เป็นตัวอย่างของ class เดียวกัน
- รูปที่ 5.5.1 แสดงการเลือก attribule hair color เป็น node แรก ที่จุดนี้ tree ที่สร้างขึ้นแยกตัวอย่างได้ในกรณีที่ hair color เป็น red (class เป็น sunburned) และ brown (class เป็น none) แต่ในกรณีที่ hair color เป็น blonde ยังแยกตัวอย่างไม่ได้ (มีตัวอย่างที่เป็นทั้ง sunburned และ none ปะปนกันอยู่)



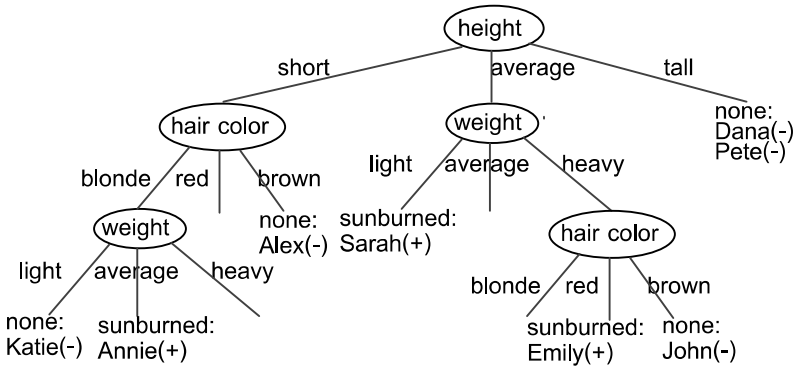
รูปที่ 5.5.1 สร้าง identification tree โดย node แรกเป็น hair color

- ในกรณีที่ hair color เป็น blonde เราสร้าง node ใหม่ตามรูป



รูปที่ 5.5.2 สร้าง node ที่ 2 เป็น lotion used

- identification tree ที่สอดคล้องกับตัวอย่าง อาจมีได้มากกว่า 1 เช่น เราอาจสร้าง tree ได้ดังรูป

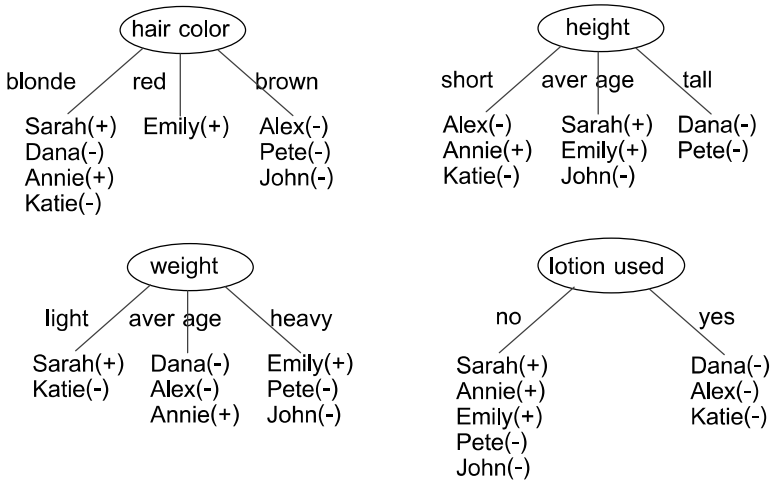


รูปที่ 5.5.3 identification tree ที่แยกตัวอย่างได้ แต่ไม่ถูกต้องตามความรู้สึกของคน

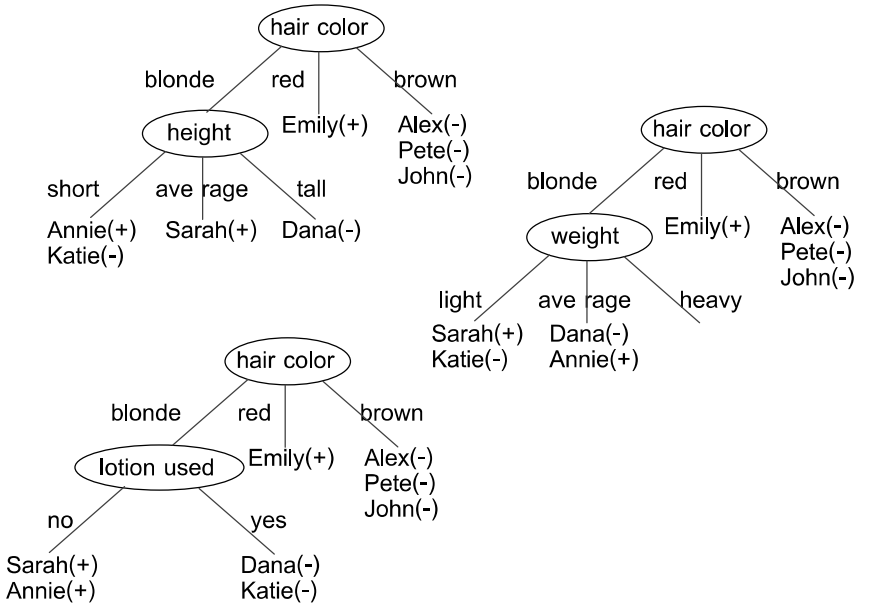
- ระหว่าง trees 2 ต้นในตัวอย่างที่แล้ว เราอยากได้ tree ต้นแรก
- Occam's razor ในการสร้าง identification tree
  - identification tree ที่มีขนาดเล็กที่สุดที่สอดคล้องกับตัวอย่าง เป็น tree ที่ดีที่สุด
- การหา tree ต้นที่เล็กที่สุดจะเสียค่าใช้จ่ายในการคำนวณมาก ดังนั้นจึงไม่สามารถกระทำได้ในทางปฏิบัติ

การเลือกตัว test

- หลักการสร้าง tree แบบหนึ่งคือพยายามเลือก test node ที่แยกตัวอย่างเป็นเซ็ตย่อย โดยที่ ทำให้สมาชิกส่วนใหญ่ในแต่ละเซ็ตย่อยเป็น class เดียวกันมากที่สุด
- ตัวอย่างเช่นในรูปที่ 5.5.4 แสดงผลของแต่ละ test node ในกรณีของ test node เป็น hair color สามารถแยกตัวอย่างเป็น 3 เซตย่อย เซ็ตย่อยแรก (blond) มีตัวอย่างของ 2 classes ปนกันอยู่ ส่วนเซ็ตย่อยที่ 2 (red) และ 3 (brown) มีตัวอย่างของ class sunburned และ none อยู่อย่างเดียวตามลำดับ



รูปที่ 5.5.4 ผลของแต่ละ test node :
hair color แยกตัวอย่างออกเป็น class ได้ดีที่สุด



รูปที่ 5.5.5 ผลของแต่ละ test node ที่เพิ่มเข้าใน subtree ของรูปที่ 5.5.4

- หลังจากนี้ เราแบ่งสมาชิกในเซ็ตย่อยที่ 1 ออกเป็น class ต่อไป
- รูปที่ 5.5.5 แสดงการสร้าง tree ต่อจากเดิม โดยให้ test node ที่เพิ่มเข้ามาเป็น height, weight และ lotion used ตามลำดับ
- ผลของ test node ที่เป็น lotion used สามารถแยกตัวอย่างออกเป็น class ได้อย่างสมบูรณ์ ที่จุดนี้เราหยุดการสร้าง tree

<u>วัดความสามารถในการแยกตัวอย่างได้อย่างไร</u>

- ID-3 เป็นโปรแกรมที่สร้าง identification tree โดยใช้ information theory เป็นตัววัดความสามารถในการแยกตัวอย่าง (gain) ของแต่ละ node

$$Gain(node) = \left(\sum_c -\frac{n_{tc}}{n_t}\log_2\frac{n_{tc}}{n_t}\right) - \sum_b\left(\frac{n_b}{n_t}\times\left(\sum_c -\frac{n_{bc}}{n_b}\log_2\frac{n_{bc}}{n_b}\right)\right)$$

โดยที่ n_b คือ จำนวนตัวอย่างของ link b
n_t คือ จำนวนตัวอย่างของทุก link รวมกัน
n_{bc} คือ จำนวนตัวอย่างของ class c ที่ link b

- เช่น node hair color ในรูปที่ 5.5.4 มีค่า gain เป็น

$$\left[-\frac{3}{8}\log_2\frac{3}{8}-\frac{5}{8}\log_2\frac{5}{8}\right]-\left[\frac{4}{8}\left(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4}\right)+\frac{1}{8}\times 0+\frac{3}{8}\times 0\right]=0.45$$

ถ้าเราคำนวณค่า Gain ของทุก node จะได้ดังนี้

Gain(hair color) = 0.45 **Gain(height) = 0.26**

Gain(weight) = 0.01 **Gain(lotion) = 0.34**

ในกรณีนี้เราเลือก node hair color เพราะมีค่า Gain มากที่สุด (แยกตัวอย่างออกเป็น class ได้ดีที่สุด)

- เมื่อคำนวณ Gain ของทุก node ในรูปที่ 5.5.5 จะได้

Gain(height) = 0.5 **Gain(weight) = 0.0** **Gain(lotion) = 1.0**

ซึ่งแสดงว่า node lotion ดีที่สุด

- Gain ได้รับการพิสูจน์ว่าเป็นฟังก์ชั่นที่ดีที่สุดฟังก์ชั่นหนึ่งโดยไม่ขึ้นกับชนิดของข้อมูล

การเปลี่ยนจาก tree เป็น rule

- เมื่อเราสร้าง tree เรียบร้อยแล้ว เราสามารถเปลี่ยน tree ให้อยู่ในรูปของ rule "IF THEN" ได้ โดยแสดงทุก path เริ่มต้นจาก root node ไปยัง leaf node ทุกครั้งที่พบ test node เพิ่ม test node กับค่าของ test ไว้ในส่วนของ IF และเมื่อพบ leaf node ให้ใส่ class ไว้ในส่วนของ THEN
- จาก tree ที่สร้างในรูป เราเปลี่ยนเป็น rules ได้ดังนี้

(1) IF the person's hair color is blonde
the person uses lotion
THEN nothing happens

(2) IF the person's hair color is blonde
the person uses no lotion
THEN the person turns red

(3) IF the person's hair color is red
THEN the person turns red

(4) IF the person's hair color is brown
THEN nothing happens

- ตารางด้านล่างแสดงการเปรียบเทียบการใช้ learning tools (ID-3) และไม่ใช้ ในการพัฒนา expert system

	Application	No. of Rules	Develop (Man Ys)	Maintain (Man Ys)	Learning Tools
MYCIN	Medical Diagnosis	400	100	N/A	N/A
XCON	VAX computer configuration	8,000	180	30	N/A
GASOIL	Hydrocarbon separtation system configuration	2,800	1	0.1	ExpertEase and Extran7
BMT	Configuration of fire-protection equipment in buildings	30,000	9	2.0	1st Class and Rulemaster

- GASOIL และ BMT เป็น expert systems ที่สร้างโดย learning tools ซึ่งพัฒนามาจากโปรแกรม ID-3