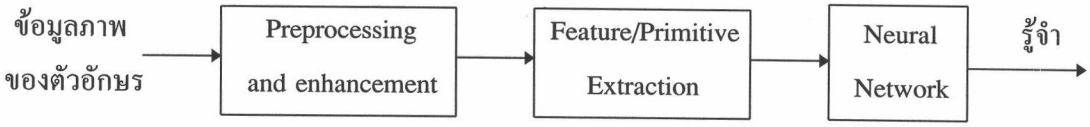
บทที่ 3

กรรมวิธีการการรู้จำ

โครงสร้างของระบบการรู้จำ

จากที่กล่าวมาแล้วข้างต้น ตัวอักษรภาษาไทยมีโครงสร้างของตัวอักษรที่ค่อนข้างซับซ้อน ดังนั้นจึงจำเป็นต้องมีกระบวนการเปลี่ยนแปลงภาพข้อมูลของตัวอักษร ให้มีความเหมาะสมสำหรับการรู้จำ พิจารณาโครงสร้างของระบบการรู้จำดังรูปที่ 3.1 ข้อมูลภาพของตัวอักษรจะถูกนำมาปรับปรุงคุณภาพโดยการจำกัดสัญญาณรบกวน (Noise Reduction) และภาพตัวอักษรจะถูกทำให้บาง (Thinning) โดยให้มีความกว้างของเส้นตัวอักษรเพียง 1 จุดภาพ หลังจากนั้นข้อมูลของภาพตัวอักษรจะถูกเข้ารหัส (Condition Code) เพื่อหาทิศทาง ซึ่งจะนำรหัสต่างๆเหล่านี้มาแปลงเป็นเวกเตอร์โดยการเปรียบเทียบกับรหัสแบบลูกโซ่ของฟรีแมน (Freeman chain code) 8 ทิศทาง (สนธยา เมรินทร์, 2537) และจะนำเอาเฉพาะเวกเตอร์ที่มีทิศทางเดียวกันรวมกันเป็นแบบเปรียบหลัก (Primitive) โดยที่ภาพข้อมูลของตัวอักษรทั้งหมดจะถูกแทนด้วยแบบเปรียบหลักซึ่งจะถูกเชื่อมโยงกันในรูปแบบไวยากรณ์ต้นไม้ (Tree Grammar) หลังจากที่แทนภาพข้อมูลของตัวอักษรด้วยแบบเปรียบหลักใดๆแล้ว ข้อมูลแบบเปรียบเหล่านี้จะถูกป้อนให้กับระดับข้อมูลเข้า (Input Layer) ของนิวรอลเน็ตเวิร์กเพื่อการฝึกและการรู้จำต่อไป



รูปที่ 3.1 โครงสร้างการรู้จำของระบบ

การปรับปรุงคุณภาพของภาพข้อมูล (Preprocessing and enhancement)

เนื่องจากภาพของตัวอักษรที่ได้นั้น จะได้มาจากเครื่องสแกนเนอร์ ซึ่งข้อมูลของภาพตัวอักษรจะถูกจัดเก็บในรูปแบบของอิมเมจไฟล์ เช่น ไฟล์ตระกูล BMP หรือไฟล์ตระกูล PCX และในกระบวนการสแกนภาพนั้น จะพบว่ามีจุดภาพที่ไม่ต้องการปะปนเข้ามาในภาพของข้อมูลตัวอักษร หรือมีจุดภาพของข้อมูลตัวอักษรขาดหายไป ดังนั้นจากรูปโครงสร้างการรู้จำในรูปที่ 3.1 จะมีกระบวนการของการประมวลผลและปรับสัญญาณเบื้องต้น ซึ่งทำหน้าที่กำจัดสัญญาณรบกวน (Noise Reduction), ทำภาพของตัวอักษรให้บาง (Thinning) และการปรับกรอบของภาพตัวอักษร ดังแสดงการทำงานในรูปที่ 3.2

รูปที่ 3.2 แสดงการทำงานของการปรับปรุงคุณภาพของภาพข้อมูล

1. การกำจัดสัญญาณรบกวน (Noise Reduction) จะเป็นกรรมวิธีในการกำจัดจุดภาพที่ไม่ต้องการออกจากภาพข้อมูลของตัวอักษร หรือเพิ่มจุดภาพที่ขาดหายไป โดยที่จะต้องไม่ทำให้ข้อมูลหลักของภาพตัวอักษรมีการเปลี่ยนแปลงรูปร่างไปจากเดิม สัญญาณรบกวนที่พบในภาพข้อมูลตัวอักษรที่สามารถแก้ไขได้ คือ สัญญาณรบกวนที่มีลักษณะเป็นจุดเดี่ยวและรูเดี่ยวในบริเวณตัวอักษร ดังรูปที่ 3.3.ก และ รูปที่ 3.3.ข

รูปที่ 3.3.ก สัญญาณรบกวนแบบรูเดี่ยว

รูปที่ 3.3.ข สัญญาณรบกวนแบบจุดเดี่ยว

วิธีการกำจัดสัญญาณรบกวนทั้งสองชนิดจะใช้ตารางหน้าต่างขนาด 3x3 จุดภาพ ดังแสดงในรูปที่ 3.4 ตรวจสอบกับทุกๆจุดภาพของข้อมูลภาพตัวอักษร โดยที่จะมีเงื่อนไขของการตรวจสอบดังนี้

1.1 ถ้า P เป็นจุดภาพที่กำลังพิจารณา และจุดภาพข้างเคียงของ P ทั้ง 8 ตำแหน่ง (x1-x8) ไม่มีจุดภาพใดๆเลย แสดงว่า P เป็นจุดภาพเดี่ยวให้ทำการลบจุดภาพนี้ออกไป

1.2 ถ้า P เป็นจุดภาพที่กำลังพิจารณา และจุดภาพข้างเคียงของ P ทั้ง 8 ตำแหน่ง (x1-x8) เป็นจุดภาพทั้ง 8 ตำแหน่ง แสดงว่า P เป็นรูจุดภาพเดี่ยวให้ทำการเปลี่ยนจุด P นี้เป็นจุดภาพ

X4	X3	X2
X5	P	X1
X6	X7	X8

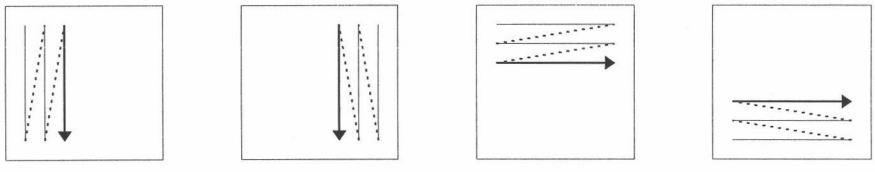
รูปที่ 3.4 แสดงหน้าต่างที่ใช้กำจัดสัญญาณรบกวน และ ภาพที่ได้จากการกำจัดสัญญาณรบกวน

2. การทำตัวอักษรให้บาง (Thinning) เป็นกระบวนการที่เปลี่ยนแปลงภาพข้อมูลให้เหลือเฉพาะเส้นโครงร่างของภาพตัวอักษร (Skeleton) เท่านั้น โดยข้อมูลของภาพที่ผ่านกระบวนการทำตัวอักษรให้บางนั้นจะเหลือความกว้างของเส้นตัวอักษร 1 จุดภาพ วิธีการของการทำตัวอักษรให้บางนั้นมีอยู่ด้วยกันหลายวิธีขึ้นอยู่กับลักษณะของภาพข้อมูล เช่น การทำให้บางแบบเนื่องกัน (Sequential Processing Thinning), การทำให้บางแบบขนาน (Parallel Processing Thinning) ฯลฯ เนื่องจากภาพของตัวอักษรที่ผ่านกระบวนการทำให้บางแล้วจะต้องไม่ทำให้มีการขาดตอนของลายเส้นตัวอักษร ดังนั้นจึงเลือกใช้วิธีของการทำให้บางแบบเนื่องกัน หรือ Classical Thinning ในการทำภาพตัวอักษรให้บาง เนื่องจากวิธีดังกล่าวไม่ทำให้เกิดการขาดตอนของลายเส้นตัวอักษร และมีโครงร่างคล้ายกับตัวอักษรต้นแบบค่อนข้างมาก แต่ข้อเสียของวิธีการทำให้บางแบบเนื่องกันคือจะใช้เวลาในการประมวลผลค่อนข้างมากเพราะจะต้องมีการประมวลผลซ้ำๆ กันถึง 4 ครั้ง (สนธยา เมรินทร์, 2537) ภาพข้อมูลตัวอักษรก่อนการทำตัวอักษรให้บาง และผลการทำตัวอักษรให้บางแสดงไว้ในรูปที่ 3.5.ก และ รูปที่ 3.5.ข

รูปที่ 3.5.ก แสดงภาพก่อนการทำตัวอักษรให้บาง

รูปที่ 3.5.ข แสดงภาพหลังจากทำตัวอักษรให้บาง

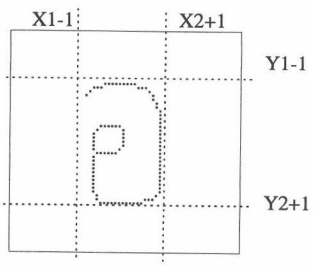
3. การปรับกรอบของตัวอักษร ภาพข้อมูลตัวอักษรที่ผ่านกระบวนการทำให้บางจะมีขนาดของภาพตัวอักษรเล็กลง เนื่องจากความหนาของภาพตัวอักษรจะถูกปรับให้เหลือ 1 จุดภาพ ดังนั้นจึงต้องมีการปรับขอบของตัวอักษรเสียใหม่ โดยมีวิธีการตรวจสอบทั้งกรอบของภาพตัวอักษรทั้งด้านบน ด้านล่างด้านซ้าย และด้านขวา ดังแสดงในรูปที่ 3.6



ก. ด้านซ้าย ข. ด้านขวา ค. ด้านบน ง. ด้านล่าง

รูปที่ 3.6 แสดงทิศทางการตรวจสอบกรอบของภาพตัวอักษร

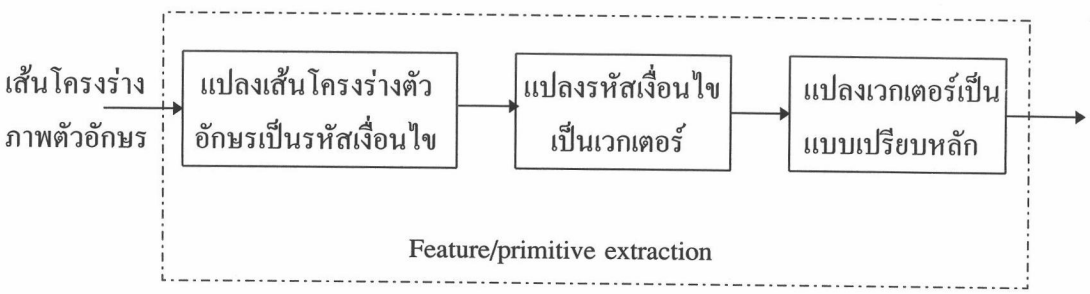
เนื่องจากวิธีการตรวจสอบกรอบของภาพตัวอักษรนั้น จะใช้ตารางหน้าต่าง 3x3 ในการตรวจสอบ ดังนั้นเพื่อความสะดวกจึงจำเป็นต้องเว้นขอบภาพนอกสุดทั้ง 4 ด้านให้ว่างไว้ด้านละ 1 จุดภาพ ด้วยวิธีการนี้ขนาดของกรอบภาพใหม่จะมีตำแหน่งดังนี้ ขนาดของขอบด้านซ้ายที่ตำแหน่ง X_1 - 1 ขอบด้านขวาที่ตำแหน่ง X_2 + 1 ขอบด้านบนที่ตำแหน่ง Y_1 - 1 ขอบด้านล่างที่ตำแหน่ง Y_2 + 1 โดยที่ X_1, X_2, Y_1 และ Y_2 เป็นตำแหน่งของภาพตัวอักษรด้านซ้าย, ด้านขวา, ด้านบน และด้านล่างตามลำดับ ดังแสดงในรูปที่ 3.7



รูปที่ 3.7 แสดงตำแหน่งการปรับกรอบของภาพตัวอักษร

การวิเคราะห์หาลักษณะสำคัญของตัวอักษร (Feature/primitive extraction)

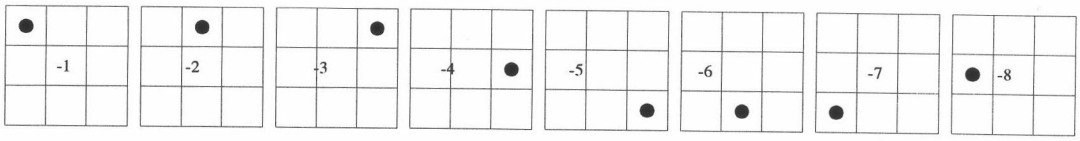
หลังจากที่ข้อมูลภาพของตัวอักษรผ่านกระบวนการปรับปรุงคุณภาพของภาพตัวอักษรแล้ว จะได้ข้อมูลของภาพตัวอักษรที่เหลือเฉพาะเส้นโครงร่างของภาพตัวอักษร (Skeleton) ข้อมูลของเส้นโครงร่างของภาพตัวอักษรนี้จะถูกนำมาหาลักษณะเด่นหรือลักษณะสำคัญ(Feature/primitive extraction) เพื่อใช้เป็นข้อมูลแทนภาพของตัวอักษรแต่ละตัวตัว และใช้เป็นข้อมูลที่ใช้ในการฝึกหัดระบบ และการรู้จำตัวอักษรต่อไป โดยวิธีของการหาลักษณะเด่นของภาพตัวอักษรนั้น จะนำภาพเส้นโครงร่างของภาพตัวอักษรที่มีความกว้างของลายเส้นตัวอักษร 1 จุดภาพ มาผ่านกระบวนการเข้ารหัสเงื่อนไข (Condition code) เมื่อได้จุดภาพที่เข้ารหัสเงื่อนไขแล้ว ก็จะนำรหัสเงื่อนไขเหล่านี้มาคำนวณแปลงเป็นเวกเตอร์ (Vector) โดยพิจารณาจากรหัสเงื่อนไขที่มีทิศทางเดียวกัน หลังจากที่แปลงข้อมูลของภาพตัวอักษรเป็นเวกเตอร์แล้ว จะนำเวกเตอร์เหล่านี้มาเปรียบเทียบกับรหัสแบบลูกโซ่ของฟรีแมน (Freeman chain code) เพื่อที่จะเปลี่ยนข้อมูลเวกเตอร์ให้อยู่ในรูปของแบบเปรียบหลัก และในขั้นตอนสุดท้ายของการหาลักษณะสำคัญของภาพตัวอักษรจะได้แบบเปรียบใดๆ ซึ่งใช้แทนข้อมูลภาพของตัวอักษรนั้น ดังแสดงโครงสร้างการทำงานในรูปที่ 3.8



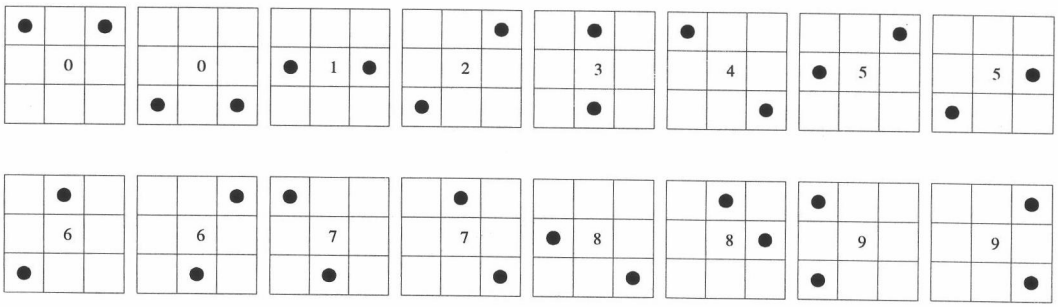
รูปที่ 3.8 แสดงการหาลักษณะสำคัญของข้อมูลภาพเส้นโครงร่างตัวอักษร

การแปลงเส้นโครงร่าง (Skeleton) ให้เป็นรหัสเงื่อนไข (Condition code)

การแปลงเส้นโครงร่างของภาพตัวอักษรให้เป็นรหัสเงื่อนไข (Condition code) จะเป็นการนำเส้นโครงร่างของภาพตัวอักษร ที่ได้จากกระบวนการปรับปรุงคุณภาพของตัวอักษรมาทำการเข้ารหัสเงื่อนไข โดยรหัสเงื่อนไขต่างๆนั้นจะพิจารณาจากจุดภาพรอบข้าง ซึ่งจะใช้ตารางหน้าต่างขนาด 3X3 จุดภาพ ตรวจสอบกับทุกๆจุดภาพ และทำการเปรียบเทียบกับจุดภาพรอบข้างทั้ง 8 จุดภาพ (สนธยา เมรินทร์, 2537; Lam and Suen, 1988) ดังรูปที่ 3.9 แสดงรหัสเงื่อนไขชนิดต่างๆ



รูปที่ 3.9.ก แสดงรหัสเงื่อนไข (Condition code) ที่เป็นลบ (จุดปลาย)



รูปที่ 3.9.ข แสดงรหัสเงื่อนไข (Condition code) ที่เป็นบวก (จุดต่อเนื่อง)

วิธีการเข้ารหัสเงื่อนไข (Condition code) โดยใช้ตารางขนาด 3X3 ดังรูปที่ 3.9.ก และ 3.9.ข มีหลักการพิจารณาจากจุดภาพรอบข้างดังนี้

- ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างมากกว่า 2 จุดภาพเป็นต้นไป แสดงว่าเป็นจุดทางแยก (junction point) ของภาพตัวอักษร ให้แทนจุดนั้นด้วยรหัสเงื่อนไข -9

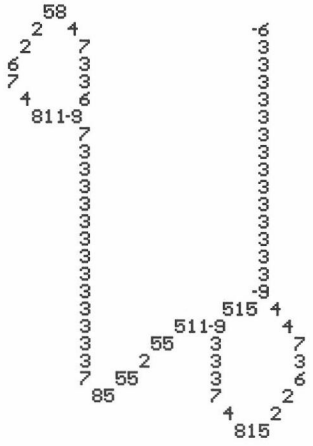
- ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างเพียง 1 จุดภาพ แสดงว่าเป็นจุดปลาย (end point) ของภาพตัวอักษร ให้แทนจุดนั้นด้วยรหัสเงื่อนไขเลขลบ ดังแสดงในรูปที่ 3.9.ก

- ถ้า X เป็นจุดภาพ และมีจุดภาพรอบข้างเท่ากับ 2 จุดภาพ แสดงว่าเป็นจุดต่อเนื่องของภาพตัวอักษร ให้แทนจุดนั้นด้วยรหัสเงื่อนไขเลขบวก ดังแสดงในรูปที่ 3.9.ข

- ถ้า X ไม่ใช่จุดภาพ หรือเป็นจุดภาพที่มีข้อกำหนดนอกเหนือจากที่ได้ระบุไว้ข้างต้น ให้แทนจุดภาพนั้นด้วยรหัสเงื่อนไข -128

หลังจากที่ผ่านกระบวนการเข้ารหัสเงื่อนไข (Condition code) แล้วจะได้ข้อมูลของภาพตัวอักษรดังรูปที่ 3.10 ซึ่งแสดงรหัสเงื่อนไขของจุดภาพตัวอักษรในลักษณะต่างๆ เช่น ทางแยกของตัวอักษร, จุดปลายของตัวอักษร และจุดต่อเนื่องของตัวอักษร

ก. ภาพข้อมูลที่ถูกทำให้บาง



ข. ภาพข้อมูลที่ผ่านการเข้ารหัสเงื่อนไข

รูปที่ 3.10 แสดงภาพข้อมูลตัวอักษรที่ผ่านกระบวนการเข้ารหัสเงื่อนไข

การแปลงรหัสเงื่อนไข (Condition code) ให้เป็นเวกเตอร์

จากรหัสเงื่อนไขชนิดต่างๆดังกล่าว สามารถที่จะจำแนกตำแหน่งต่างๆของจุดภาพนั้นว่าทำมุมในทิศทางใดตามตารางที่ 3.1 โดยรหัสเงื่อนไข 1 ถึง 4 เรียกว่ารหัสเงื่อนไขหลัก ซึ่งจะทำมุมเป็นทวีคูณของ 45 องศา รหัสเงื่อนไข 5 ถึง 8 เรียกว่ารหัสเงื่อนไขรอง ซึ่งจะทำมุมเป็นทวีคูณของ 30 องศา รหัสเงื่อนไข 0 และ 9 เป็นจุดหักเหของเส้นโครงร่างของภาพ

ตารางที่ 3.1 แสดงมุมของรหัสเงื่อนไข

รหัสเงื่อนไข	มุม (องศา)
1	0
2	45
3	90
4	135
5	30
6	60
7	120
8	150

ในกรณีของรหัสเงื่อนไขที่มีการเรียงของรหัสเงื่อนไขหลักและรหัสเงื่อนไขรองคละกันไป จะได้มุมของเส้นโครงร่างของภาพดังที่แสดงในตารางที่ 3.2

ตารางที่ 3.2 แสดงมุมของรหัสเงื่อนไขหลักและรหัสเงื่อนไขรองคละกัน

รหัสเงื่อนไขเรียงต่อเนื่อง	มุม (องศา)
1 - 5 - 1	$0 < \theta < 30$
2 - 5 - 2	$30 < \theta < 45$
2 - 6 - 2	$46 < \theta < 60$
3 - 6 - 3	$60 < \theta < 90$
3 - 7 - 3	$90 < \theta < 120$
4 - 7 - 4	$120 < \theta < 135$
4 - 8 - 4	$135 < \theta < 150$
1 - 8 - 1	$150 < \theta < 180$

ด้วยรหัสเงื่อนไข -1 ถึง -8 (negative condition code) และรหัสเงื่อนไข 1 ถึง 9 (non-negative condition code) ดังแสดงในรูปที่ 3.9.ก และ รูปที่ 3.9.ข จะสามารถคำนวณตำแหน่งของจุดภาพจุดถัดไปโดยใช้สูตรดังตารางที่ 3.3 ในกรณีที่เป็นจุดภาพต่อเนื่อง และใช้สูตรดังตารางที่ 3.4 ในกรณีที่เป็นจุดปลาย

ตารางที่ 3.3 แสดงสูตรการคำนวณจุดภาพตำแหน่งถัดไปในกรณีของจุดต่อเนื่อง

รหัสเงื่อนไขของตำแหน่งปัจจุบัน	ตำแหน่งของจุดถัดไป (n_i , n_j)
0	(pi , 2j - pj)
1	(i , 2j - pj)
2 , 4	(2i - pj , 2j - pj)
3	(2i - pj , j)
5	(i - j + pj , 2j - pj) ; i = pi (i , 2j - pj) ; otherwise
6	(2i - pj , j -i + pj) ; j = pj (2i - pi , j) ; otherwise
7	(2i - pj , i + 1 - pj) ; j = pj (2i - pi , j) ; otherwise
8	(i + j - pj , 2j - pj) ; i = pi (i , 2j - pj) ; otherwise
9	(2i - pi , pj)

(i , j) = ตำแหน่งของจุดภาพปัจจุบัน

(pi , pj) = ตำแหน่งของจุดภาพก่อนหน้า

(n_i , n_j) = ตำแหน่งของจุดภาพถัดไป

จากตาราง 3.3 แสดงการคำนวณจุดภาพที่ตำแหน่งถัดไปในกรณีของจุดต่อเนื่องนั้นสามารถได้สรุปเป็นสูตรได้เพื่อใช้ในการคำนวณได้ดังนี้ (Lam and Seuen, 1988)

$$(ni,nj)=\begin{matrix}(i-j+pj,2*j-pj) & \text{if } i = pi\\ (i,2*j-pj) & \text{otherwise}\end{matrix}$$

ตารางที่ 3.4 แสดงสูตรการคำนวณจุดภาพตำแหน่งถัดไปในกรณีของจุดปลาย

รหัสเงื่อนไขของตำแหน่งปัจจุบัน	ตำแหน่งของจุดถัดไป (n_i , n_j)
-1	(i - 1 , j - 1)
-2	(i - 1 , j)
-3	(i - 1 , j + 1)
-4	(i , j + 1)
-5	(i + 1, j + 1)
-6	(i + 1 , j)
-7	(i + 1 , j - 1)
-8	(i , j - 1)

(i , j) = ตำแหน่งของจุดภาพปัจจุบัน

(pi , pj) = ตำแหน่งของจุดภาพก่อนหน้า

(n_i , n_j) = ตำแหน่งของจุดภาพถัดไป

เมื่อสามารถคำนวณหาทิศทางของเส้นโครงร่างของตัวอักษรได้แล้ว ก็จะนำทิศทางของเส้นโครงร่างตัวอักษรซึ่งคำนวณได้จากการรวมเอาจุดภาพที่มีรหัสเงื่อนไขเหมือนกัน หรือเป็นเงื่อนไขที่ต่อเนื่องกันรวมเข้าด้วยกันเป็นเวกเตอร์ โดยที่เวกเตอร์ที่ใช้งานนั้นมีอยู่ 2 ชนิด คือ

- เวกเตอร์เส้นตรง ใช้ในการแทนจุดภาพตัวอักษรที่มีทิศทางเป็นเส้นตรง โดยที่จะพิจารณารวมเอาทิศทางของจุดภาพตัวอักษรที่มีทิศทางลักษณะเดียวกัน รวมกันเข้าเป็นเวกเตอร์เส้นตรงเดียวกัน การเก็บข้อมูลของเวกเตอร์เส้นตรง จะเก็บตำแหน่งต้นทางและปลายทางของเวกเตอร์เส้นตรงนั้นๆ

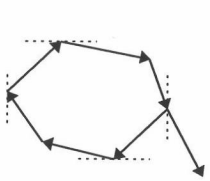
- เวกเตอร์วงกลม ใช้ในการแทนจุดภาพที่มีตำแหน่งเริ่มต้น และตำแหน่งสุดท้ายที่วนมาบรรจบในตำแหน่งเดียวกัน ซึ่งอาจเป็นได้ทั้งเวกเตอร์วงกลมอิสระ หรือเวกเตอร์วงกลมที่มีเวกเตอร์อื่นมาต่อเชื่อมด้วย โดยจุดประสงค์หลักในการใช้เวกเตอร์วงกลมเพื่อที่จะใช้แทนส่วนหัวของตัวอักษร ดังนั้นเวกเตอร์วงกลมจึงถูกกำหนดให้มีจุดเชื่อมต่อเพียงจุดเดียวที่จะสามารถเชื่อมต่อกับเวกเตอร์เส้นตรง การเก็บข้อมูลของเวกเตอร์จะเก็บตำแหน่งของเวกเตอร์วงกลม และตำแหน่งของจุดต่อเชื่อมจากวงกลมนั้นไปยังเวกเตอร์อื่นๆ (ถ้ามี)

วิธีการคำนวณหาเวกเตอร์เส้นตรงนั้น จะเป็นการนำทิศทางของเส้นโครงร่างตัวอักษรที่มีลักษณะเป็นเส้นตรง โดยการพิจารณานำรหัสเงื่อนไขที่เหมือนกันและสอดคล้องกัน หรือเป็นรหัสเงื่อนไขที่ต่อเนื่องกันมารวมกันเข้าเป็นเวกเตอร์เส้นตรง 1 เวกเตอร์ ดังตารางที่ 3.5 แสดงรหัสเงื่อนไขที่ต่อเนื่อง และรหัสเงื่อนไขที่ไม่ต่อเนื่องกัน สนธยา เมรินทร์ ได้เสนอวิธีการหาเวกเตอร์เส้นตรงตามระบบการทำงานที่ 1 (Algorithm 1)

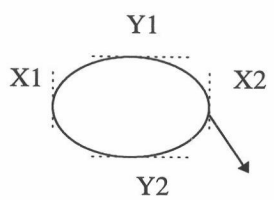
ตารางที่ 3.5 แสดงรหัสต่อเนื่อง และรหัสไม่ต่อเนื่องของรหัสเงื่อนไข

รหัสเงื่อนไข	รหัสที่ต่อเนื่อง	รหัสที่ไม่ต่อเนื่อง
1	1 , 5 , 8	-
2	2 , 5 , 6	9 , 0
3	3 , 6 , 7	-
4	4 , 7 , 8	9 , 0
5	1 , 2 , 5 ,6	0 , 8 , 9
6	2 , 3 , 5 , 5	0 , 7 , 9
7	3 , 4 , 7 , 8	0 , 6 , 9
8	2 , 4 , 7 , 8	0 , 5 , 9
9	2 , 4 , 5 , 6 , 7 , 8	ทุกรหัส
0	2 , 4 , 5 , 6 , 7 , 8	ทุกรหัส

การสร้างเวกเตอร์วงกลม จะสามารถสร้างจากเวกเตอร์เส้นตรง โดยพิจารณาจากจุดทางแยกที่มี 3 ทางแยก และมี 2 ทางแยกที่สามารถเชื่อมต่อถึงกันและกัน โดยที่ไม่มีจุดทางแยกอื่นๆอีกในระหว่างเส้นทางที่เชื่อมต่อถึงกัน แสดงว่าทางแยกทั้ง 2 นั้นมีเส้นทางการเชื่อมต่อเป็นวงกลม และทางแยกที่เหลืออีก 1 เส้นทางจะเป็นจุดเชื่อมต่อของวงกลมกับเวกเตอร์อื่น ดังแสดงในรูปที่ 3.11



ก. ก่อนการทำเป็นเวกเตอร์วงกลม



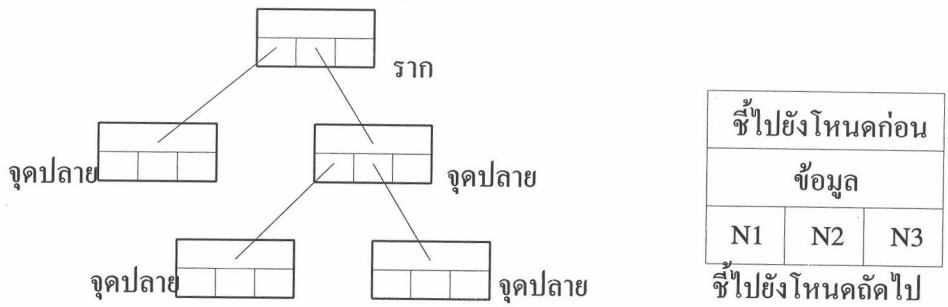
ข. หลังจากการแปลงเป็นเวกเตอร์วงกลม

รูปที่ 3.11 แสดงการแปลงเวกเตอร์เส้นตรงเป็นเวกเตอร์วงกลม

ระบบการทำงานที่ 1. การแปลงจุดภาพที่เข้ารหัสเงื่อนไขให้เป็นเวกเตอร์

1. if พบจุดปลาย หรือจุดทางแยกของภาพ (จุดภาพที่มีรหัสเงื่อนไข -1 ถึง -9)
   then
2. ไปยังจุดปลายอันแรกสุด ถ้าไม่มีให้ไปที่จุดทางแยกอันแรกสุด
3. Do จนกว่าไม่สามารถไปยังจุดปลาย หรือจุดทางแยกได้อีก
   begin
   Trace ไปยังจุดถัดไป
4. เก็บค่ารหัสเงื่อนไขของจุดภาพปัจจุบัน เป็นรหัสเงื่อนไขเริ่มต้น
5. Trace ไปยังจุดถัดไป, เก็บค่ารหัสเงื่อนไขของจุดภาพ
6. if รหัสเงื่อนไขของจุดภาพปัจจุบันเป็นรหัสต่อเนื่องกับรหัสเงื่อนไขเริ่มต้น
   then ทำ step 5
   end if
7. สร้างเวกเตอร์เส้นตรง โดย
   ตำแหน่งเริ่มต้นของเวกเตอร์ได้จากจุดภาพที่ step 4
   ตำแหน่งปลายของเวกเตอร์ ได้จากจุดภาพปัจจุบัน
   คำนวณตำแหน่งของเวกเตอร์ให้สัมพันธ์กับขนาดกรอบเวกเตอร์
8. if จุดปัจจุบันยังไม่เป็นจุดปลายหรือจุดทางแยก
   then ทำ step 4
   else ไปยังจุดปลายหรือจุดทางแยกถัดไป
   end if
   end do
9. else
   สร้างเวกเตอร์วงกลมที่ไม่มีจุดต่อเชื่อม โดยใช้ขนาดของภาพเป็นขนาดของเวกเตอร์
10. end if

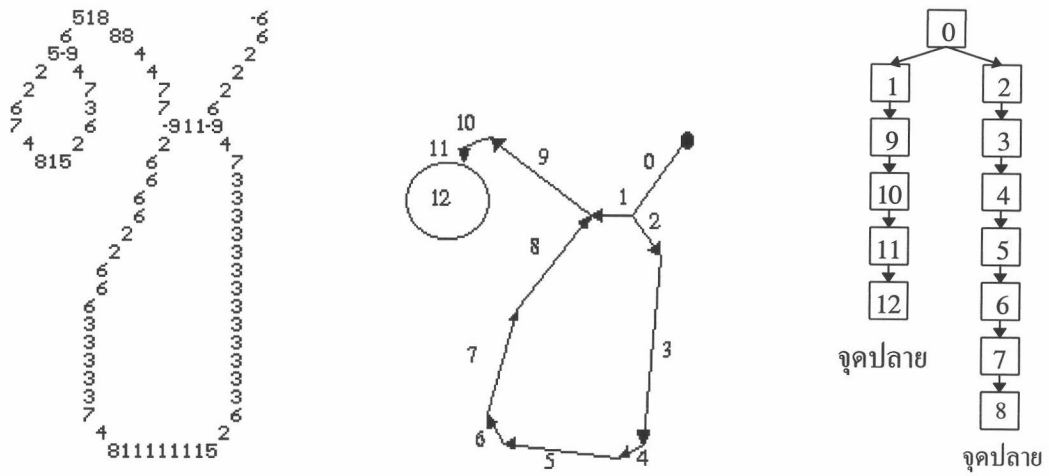
หลังจากที่แปลงข้อมูลของภาพตัวอักษรให้อยู่ในรูปของเวกเตอร์เส้นตรง และเวกเตอร์วงกลมได้แล้ว จะต้องนำเวกเตอร์ที่ใช้แทนภาพข้อมูลตัวอักษรต่างๆเหล่านี้ มาเชื่อมโยงเข้าด้วยกันให้มีลักษณะเหมือนกับโครงร่างตัวอักษรโดยใช้รูปแบบต้นไม้ แบบต้นไม้ทวิภาคอนุกรม 3 (Binary-Tree of Order 3) ที่มีการเชื่อมโยงศูนย์รวม (node) แบบรายการเชื่อมโยงเชิงคู่ (doubly link list) (สนธยา เมรินทร์, 2537) ดังแสดงในรูปที่ 3.12



ก.แสดงการเชื่อมโยงแบบ doubly link list ข. แสดงโครงสร้างของโหนด

รูปที่ 3.12 แสดงโครงสร้างต้นไม้

หลังจากภาพข้อมูลตัวอักษรผ่านกระบวนการของการแปลงรหัสเงื่อนไขเป็นเวกเตอร์แล้ว จะได้เวกเตอร์ของข้อมูลตัวอักษร และต้นไม้ของเวกเตอร์ ดังแสดงในรูปที่ 3.13 นอกจากต้นไม้เวกเตอร์ที่ได้จากโครงร่างตัวอักษรแล้ว ยังมีข้อมูลของตัวอักษรที่สามารถใช้ในการจำแนกตัวอักษรได้ เช่น จำนวนทางแยกของตัวอักษร (branch), จำนวนจุดปลายของตัวอักษร (tail), จำนวนส่วนย่อยของตัวอักษร (Session) ซึ่งประกอบด้วย 2 ส่วน เช่น ฐ และ ญ, จำนวนวงกลม (loop), ลักษณะปลายของตัวอักษรที่ยาวเกินเส้นระดับบน เช่น ป, ฟ, ฝ หรือปลายตัวอักษรที่ยาวเกินเส้นระดับล่าง เช่น ฤ, ฦ ซึ่งข้อมูลดังกล่าวสามารถที่จะจำแนกตัวอักษรที่มีโครงร่างของตัวอักษร หรือข้อมูลของเวกเตอร์ตัวอักษรที่คล้ายคลึงกันได้ เช่น บ กับ ป, ผ กับ ฝ, พ กับ ฟ, ถ กับ ฤ, ภ กับ ฦ เป็นต้น



ก. ข้อมูลภาพที่เข้ารหัสเงื่อนไข ข. เวกเตอร์ของภาพตัวอักษร ค. ต้นไม้ของเวกเตอร์

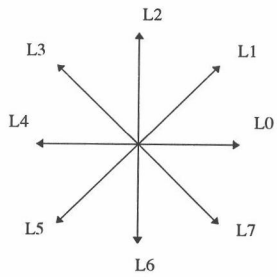
รูปที่ 3.13 แสดงการแปลงข้อมูลภาพตัวอักษรที่เข้ารหัสเงื่อนไขให้เป็นต้นไม้ของเวกเตอร์

การเปลี่ยนเวกเตอร์ให้เป็นแบบเปรียบ (Primitive)

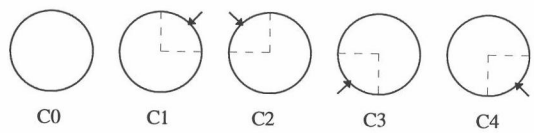
เนื่องจากข้อมูลเวกเตอร์ที่ใช้แทนข้อมูลภาพตัวอักษรนั้น จะเป็นข้อมูลของตำแหน่งจุดเริ่มต้น และจุดปลายของเวกเตอร์ ดังนั้นเพื่อเป็นการลดขนาดข้อมูลที่ใช้แทนภาพตัวอักษร จึงนำข้อมูลเวกเตอร์ของภาพข้อมูลตัวอักษรมาเปลี่ยนให้อยู่ในรูปของแบบเปรียบหลัก โดยลักษณะของแบบเปรียบที่ใช้แทนข้อมูลเวกเตอร์ของภาพตัวอักษรมีดังนี้

- แบบเปรียบเส้นตรง มีอยู่ด้วยกัน 8 รูปแบบ ดังรูปที่ 3.14.ก ซึ่งมีลักษณะเหมือนกับรหัสลูกโซ่ของฟรีแมน โดยที่แบบเปรียบแต่ละรูปแบบจะใช้แทนเวกเตอร์เส้นตรงที่ทำมุมครอบคลุมพื้นที่ 45 องศา เช่น แบบเปรียบ L1 จะถูกใช้แทนเวกเตอร์เส้นตรงที่ทำมุม 0 ถึง 45 องศา, แบบเปรียบ L2 จะถูกใช้แทนเวกเตอร์เส้นตรงที่ทำมุม 45 ถึง 90 องศา เป็นต้น

- แบบเปรียบวงกลม มีอยู่ด้วยกัน 5 รูปแบบ ดังรูปที่ 3.14.ข โดยแบบเปรียบแรก (C0) เป็นวงกลมที่ไม่มีจุดเชื่อมต่อกับเวกเตอร์อื่นๆ ใช้แทนเวกเตอร์วงกลมที่ไม่มีการเชื่อมต่อกับเวกเตอร์ชุดอื่นเลย ในส่วนของแบบเปรียบวงกลมที่เหลือ (C1, C2, C3, C4) จะใช้แทนเวกเตอร์วงกลมที่มีจุดเชื่อมต่อกับเวกเตอร์อื่นๆ โดยมีพื้นที่ของการเชื่อมต่ออยู่ในบริเวณพื้นที่แต่ละ 45 องศา



รูปที่ 3.14.ก แบบเปรียบเส้นตรง

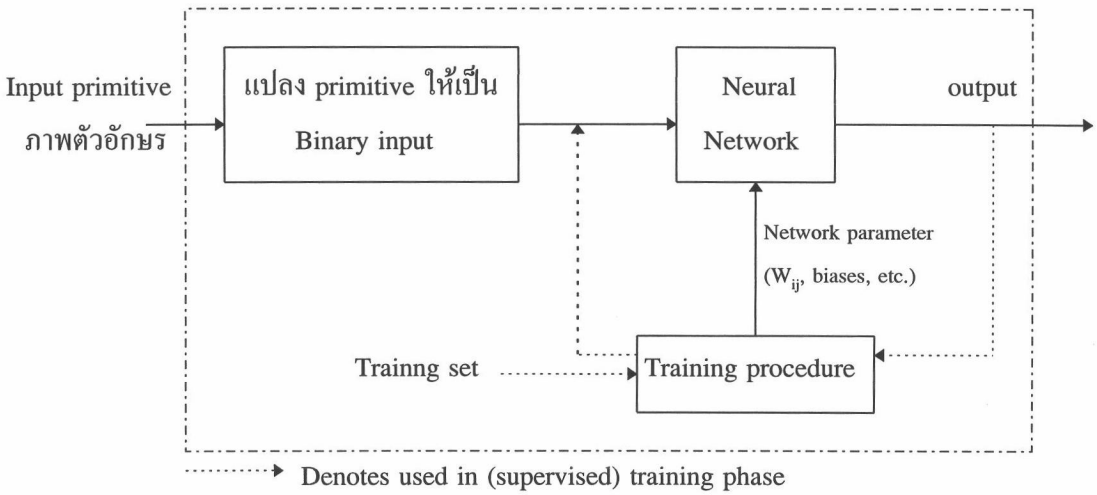


รูปที่ 3.14.ข แบบเปรียบวงกลมวงกลม

รูปที่ 3.14 แสดงรูปแบบของแบบเปรียบหลัก (primitive)

<u>การรู้จำโดยนิวรอลเน็ตเวิร์กแบบ Backpropagation</u>

หลังจากการหาลักษณะสำคัญของภาพตัวอักษร (Feature/primitive extraction) จะได้แบบเปรียบ (Primitive) ซึ่งใช้แทนข้อมูลภาพของตัวอักษรนั้น แบบเปรียบ (Primitive) ของภาพตัวอักษรจะถูกแปลงให้เป็นข้อมูลที่เหมาะสมกับนิวรอลเน็ตเวิร์กในระดับข้อมูลเข้า (Input layer) ซึ่งแบบเปรียบ (Primitive) ของภาพตัวอักษรจะถูกแปลงให้เป็นเลขฐานสอง (Binary) กล่าวคือจะมี 2 สถานะคือ 0 และ 1 เท่านั้น การจำแนกหรือการรู้จำตัวอักษรนั้นเป็นการใช้วิธีการของนิวรอลเน็ตเวิร์กที่ได้อธิบายในบทที่ 2 ซึ่งเนิวรอลเน็ตเวิร์กดังกล่าวเป็นนิวรอลเน็ตเวิร์กแบบ Backpropagation โดยที่นิวรอลเน็ตเวิร์กประเภทนี้จะต้องมีการเรียนรู้ของระบบ (Supervised learning) ก่อนที่ระบบจะสามารถจำแนกและรู้จำตัวอักษรได้ ดังนั้นจึงจะต้องมีการฝึก (Training) ระบบนิวรอลเน็ตเวิร์ก โดยใช้ภาพตัวอักษรที่เราต้องการรู้จำเป็นข้อมูลในการฝึก (Training) เมื่อกระบวนการสอนของระบบสิ้นสุดลงแล้ว ระบบการรู้จำก็จะสามารถบ่งบอกการรู้จำภาพตัวอักษร ตามข้อมูลตัวอักษรที่ป้อนเข้ามากับข้อมูลที่ระบบเรียนรู้จากการฝึก (Training) ได้ ดังรูปที่ 3.15 แสดงการรู้จำโดยวิธีนิวรอลเน็ตเวิร์ก



รูปที่ 3.15 แสดงการรู้จำตัวอักษรโดยวิธีนิวรอลเน็ตเวิร์ก

<u>การแปลงแบบเปรียบ (Primitive) ให้เหมาะสมกับระดับข้อมูลเข้า (Input Layer) ของนิวรอลเน็ตเวิร์ก</u>

เนื่องจากข้อมูลของภาพตัวอักษรที่ถูกแทนด้วยแบบเปรียบ (Primitive) ไม่สามารถป้อนให้กับระดับข้อมูลเข้า (Input Layer) ของนิวรอลเน็ตเวิร์กได้โดยตรง ดังนั้นจะต้องแปลงแบบเปรียบ (Primitive) เหล่านี้ให้เป็นเลขฐานสอง (Binary) ดังตารางที่ 3.6 แสดงวิธีการแปลงแบบเปรียบ (Primitive) ให้เป็นเลขฐานสอง (Binary)

ตารางที่ 3.6.ก แสดงการแปลงแบบเปรียบเส้นตรงให้เป็นเลขฐานสอง

แบบเปรียบเส้นตรง		เลขฐานสอง
L0	→	0000
L1	↗	0001
L2	↑	0010
L3	↖	0011
L3	←	0100
L5	↙	0101
L6	↓	0110
L7	↘	0111

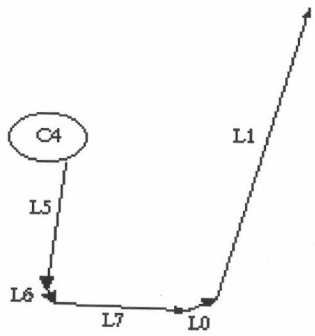
ตารางที่ 3.6.ข แสดงการแปลงแบบเปรียบวงกลมเป็นเลขฐานสอง

แบบเปรียบวงกลม	เลขฐานสอง
C0	1000
C1	1001
C2	1010
C3	1011
C4	1100

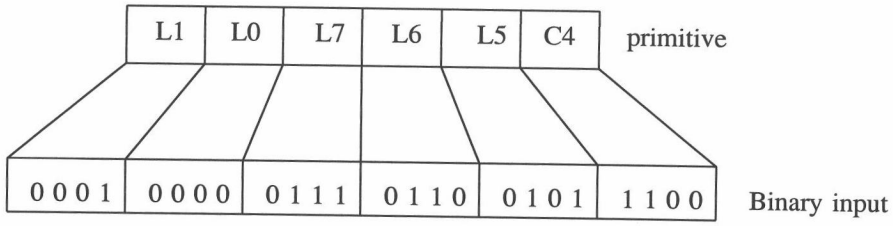
ขั้นตอนการแปลงภาพข้อมูลตัวอักษรให้เป็นเลขฐานสอง (Binary) สามารถกระทำได้ โดยนำภาพข้อมูลตัวอักษรดังแสดงในรูปที่ 3.16.ก มาผ่านกระบวนการปรับปรุงคุณภาพของภาพข้อมูล และทำให้บางเหลือเฉพาะเส้นโครงร่างของภาพตัวแสดงในรูปที่ 3.16.ข หลังจากนั้นนำเส้นโครงร่างของภาพมาเข้ารหัสเงื่อนไขแปลงเป็นเวกเตอร์และแบบเปรียบ ดังแสดงในรูปที่ 3.16.ค ในขั้นตอนต่อไปจะนำแบบเปรียบที่ได้มาเปรียบเทียบกับรูปแบบของแบบเปรียบเส้นตรง และแบบเปรียบวงกลม ดังแสดงในตารางที่ 3.6.ก และ ตารางที่ 3.6.ข ดังนั้นภาพข้อมูลในรูปที่ 3.16.ก จะถูกแปลงเป็นข้อมูลของแบบเปรียบหลักได้เป็น L1, L0, L7, L6, L5 และ C4 ตามลำดับ ซึ่งจะได้ข้อมูลเลขฐานสอง (Binary) แทนภาพตัวอักษรเป็น 0001,0000,0111,0110,0101,1100 ดังรูปที่ 3.16.ง

รูปที่ 3.16.ก ภาพตัวอักษร

รูปที่ 3.16.ข เส้นโครงร่างของภาพข้อมูล



รูปที่ 3.16.ค แสดงแบบเปรียบ

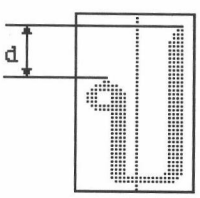


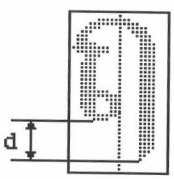
รูปที่ 3.16.ง แสดงข้อมูลแบบเปรียบ และเลขฐานสองที่ใช้แทนภาพตัวอักษร

รูปที่ 3.16 แสดงการแปลงข้อมูลภาพตัวอักษรให้เป็นเลขฐานสอง (Binary)

<u>การฝึกและการรู้จำ</u>

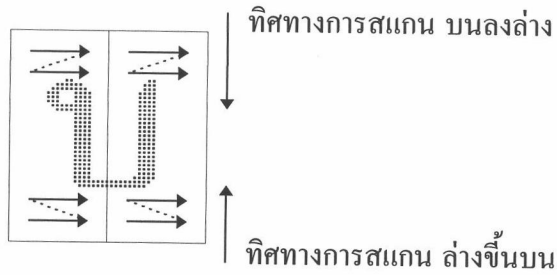
เนื่องจากข้อมูลภาพตัวอักษรที่ได้จากเส้นโครงร่างของภาพ โดยการแปลงเส้นโครงร่างของภาพให้เป็นเวกเตอร์และแบบเปรียบนั้น ยังไม่เพียงพอในการรู้จำตัวอักษรบางตัวที่มีลักษณะของเวกเตอร์และแบบเปรียบที่คล้ายคลึงกัน (บ กับ ป, ผ กับ ฝ, พ กับ ฟ, ถ กับ ฤ, ภ กับ ฦ) ดังนั้นจึงต้องนำรูปลักษณ์ (Feature) ที่มีลักษณะเด่นของแต่ละตัวอักษรมาเป็นข้อมูลให้กับนิวรอลเน็ตเวิร์ก ซึ่งข้อมูลที่ใช้นอกจากจะใช้ข้อมูลของแบบเปรียบที่ได้จากโครงร่างของภาพตัวอักษรแล้วนั้น ยังมีจำนวนวงกลมของภาพตัวอักษร, ลักษณะของปลายตัวอักษรที่อยู่เหนือเส้นระดับบน (ตัวอักษร ป, ฟ, ฝ) และลักษณะของปลายตัวอักษรที่อยู่ต่ำกว่าเส้นระดับล่าง (ตัวอักษร ฦ, ฤ) ดังรูปที่ 3.17 และรูปที่ 3.18 แสดงวิธีการคำนวณหาตำแหน่งของปลายตัวอักษรว่าอยู่เหนือเส้นระดับบน, อยู่ต่ำกว่าเส้นระดับล่าง หรือปลายของตัวอักษรมีตำแหน่งไม่เกินเส้นระดับบนและไม่ต่ำกว่าเส้นระดับล่าง





รูปที่ 3.17.ก ปลายตัวอักษรเกินเส้นระดับบน รูปที่ 3.17.ข ปลายตัวอักษรต่ำกว่าเส้นระดับล่าง

รูปที่ 3.17 แสดงระยะห่างของปลายตัวอักษรที่อยู่เกินเส้นระดับบนและต่ำกว่าเส้นระดับล่าง



รูปที่ 3.18 แสดงทิศทางการสแกนหาตำแหน่งจุดปลายตัวอักษร

เนื่องจากการรู้จำตัวอักษรในงานวิจัยนี้เป็นการรู้จำตัวอักษรทีละ 1 ตัวอักษร จึงไม่สามารถกำหนดเส้นบรรทัดของตัวอักษรได้ ดังนั้นวิธีการหาตำแหน่งจุดปลายของตัวอักษรจึงกระทำได้โดยแบ่งภาพออกเป็น 2 ส่วน จากนั้นจะสแกนหาตำแหน่งจุดภาพที่พบจุดแรกทีละส่วนจากบนลงล่างโดยสแกนภาพจากซ้ายสุดจนถึงกึ่งกลางภาพ และสแกนจากล่างขึ้นบนจากซ้ายสุดจนถึงกึ่งกลางภาพ (ภาพส่วนด้านซ้ายในรูป 3.18) ในส่วนที่เหลือก็ทำเช่นเดียวกัน แต่จะสแกนหาจุดภาพจากบนลงล่างโดยเริ่มต้นการสแกนที่กึ่งกลางภาพไปจนถึงด้านขวาสุดของภาพ และสแกนจากล่างขึ้นบนจากกึ่งกลางภาพไปจนด้านขวาสุด (ภาพส่วนด้านขวามือในรูปที่ 3.18) หลังจากนั้นจะนำตำแหน่งที่พบจุดภาพมาทำการเปรียบเทียบกับค่าที่ตั้งไว้ (ในที่นี้กำหนดไว้ที่ระดับ 1/4 ของความสูงของภาพ) ตัวอย่างเช่น ในรูปที่ 3.17.ก ถ้าระยะ d มีค่ามากกว่า 1/4 ของความสูงของภาพจะถือว่ามีปลายตัวอักษรอยู่เหนือเส้นระดับบน ในทำนองเดียวกันกับรูปที่ 3.17.ข ถ้าระยะ d มีค่ามากกว่า 1/4 ของความสูงของภาพจะถือว่ามีปลายตัวอักษรอยู่ต่ำเส้นระดับล่างเช่นกัน

ดังนั้นข้อมูลทั้งหมดที่ใช้แทนภาพตัวอักษร 1 ภาพตัวอักษรจะประกอบด้วยจำนวนวงกลมในภาพตัวอักษร, ลักษณะของปลายตัวอักษรที่อยู่เกินเส้นระดับบน ต่ำกว่าเส้นระดับล่าง หรือไม่เกินทั้งระดับบนและต่ำกว่าเส้นระดับล่าง และแบบเปรียบที่ได้จากโครงร่างและเวกเตอร์ของภาพข้อมูล ดังรูปที่ 3.19 โดยจำนวนของวงกลมจะถูกแปลงเป็นเลขฐานสอง ปลายของตัวอักษรที่เกินเส้นระดับบนจะถูกแทนด้วยเลขฐานสอง 1001 ปลายของตัวอักษรที่ต่ำกว่าเส้นระดับล่างจะถูกแทนด้วยเลขฐานสอง 0111 ปลายของตัวอักษรที่ไม่อยู่เกินเส้นระดับบนและไม่ต่ำกว่าเส้นระดับล่างจะถูกแทนด้วยเลขฐานสอง 0000

จำนวนวงกลม	ลักษณะปลายตัวอักษร	แบบเปรียบของภาพตัวอักษร
0000 = no loop	0000 = non-above&below	แบบเปรียบเส้นตรง 8 รูปแบบ
0001 = 1 loop	1001 = above	แบบเปรียบวงกลม 5 รูปแบบ
0010 = 2 loop	0111 = below	ที่แปลงเป็นเลขฐานสอง

รูปที่ 3.19 แสดงรูปแบบข้อมูลเลขฐานสองที่ใช้แทนข้อมูลภาพตัวอักษร

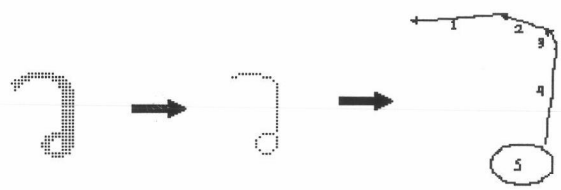
ตัวอย่างการแปลงภาพข้อมูลให้เป็นเลขฐานสอง เช่น จากรูปที่ 3.16 ซึ่งมีจำนวนวงกลม 1 วงกลม คือ หัวของตัวอักษร มีลักษณะปลายตัวอักษรที่เกินเส้นระดับบน และมีแบบเปรียบของภาพตัวอักษรคือ L1, L0, L7, L6, L5 และ C4 ตามลำดับ ดังนั้นข้อมูลเลขฐานสองที่ใช้แทนภาพในรูปที่ 3.16.ก คือ 0001,1001,0001,0000,0111,0110,0101,1100 ดังรูปที่ 3.20 แสดงการแปลงภาพตัวอักษร (รูปที่ 3.16.ก) ให้เป็นเลขฐาน 2

จำนวนวงกลม	ลักษณะปลายตัวอักษร	แบบเปรียบ
1 loop	above	L1, L0, L7, L6, L5, C4

เลขฐาน2 ที่แทนภาพ 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 1 0 0

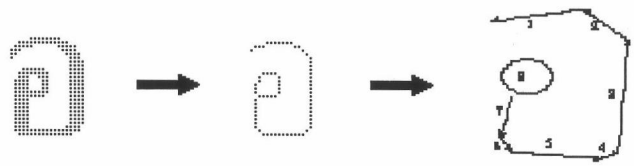
รูปที่ 3.20 แสดงวิธีการแปลงภาพข้อมูลให้เป็นเลขฐานสอง

จากทฤษฎีนิวรอลเน็ตเวิร์กในบทที่ 2 สามารถจำแนกการทำงานของระบบนิวรอลเน็ตเวิร์กได้เป็น 2 ลักษณะ คือ การแพร่ขยายแบบไปข้างหน้า (forward signal propagation) และการแพร่ขยายแบบย้อนหลัง (backward signal propagation) โดยในส่วนแรกจะเป็นการป้อนข้อมูลอินพุต (input pattern) ให้กับนิวรอลเน็ตเวิร์ก จากนั้นนิวรอลเน็ตเวิร์กจะทำการคำนวณจากระดับข้อมูลเข้า (Input layer) ไปยังระดับแสดงผล (Output layer) เพื่อให้ได้ผลที่แท้จริง (actual output) ซึ่งเป็นการคำนวณจากอินพุตไปยังเอาต์พุต และในส่วนที่เหลือจะเป็นการคำนวณจากเอาต์พุตมายังอินพุต โดยจะเป็นการคำนวณหาค่าความผิดพลาด (error signal) ระหว่างผลที่ได้จากการคำนวณของนิวรอลเน็ตเวิร์ก (actual output) กับผลที่ต้องการ (desired output) จากนั้นก็จะทำการปรับปรุงระบบ (ปรับน้ำหนักในการเชื่อมต่อ) เพื่อให้ได้ค่าผลที่ต้องการทั้งระบบ ซึ่งข้อมูลที่ใช้ในการฝึก (Training) ระบบนิวรอลเน็ตเวิร์กนั้นจะประกอบด้วยคู่ของข้อมูลอินพุต (input pattern) และเอาต์พุตที่ต้องการ (desired output)



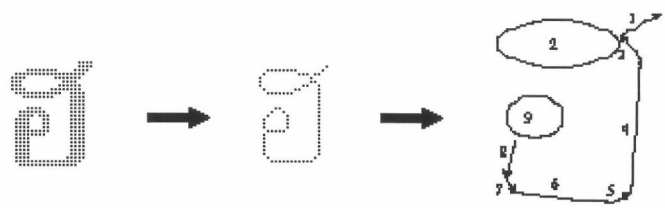
dummy input

Binary input : 0001,1001,0100,0011,0010,0001,1001,**1111,1111,1111,1111**



dummy input

Binary input : 0001,0000,0100,0011,0001,0000,0111,0110,0101,1011,**1111**

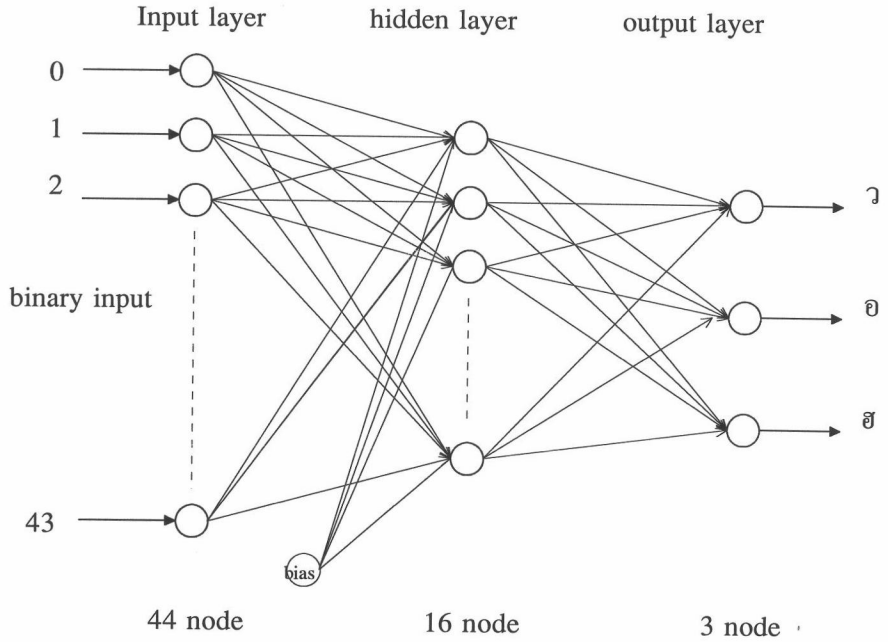


Binary input : 0010,0000,0000,1001,0011,0001,0000,0111,0110,0101,1011

รูปที่ 3.21 แสดงการสร้างข้อมูลชุดฝึก (training set) และข้อมูลจำลอง (dummy input)

ตัวอย่างการออกแบบระบบนิวรอลเน็ตเวิร์กสำหรับการรู้จำตัวอักษร เช่นสมมุติว่าต้องการให้นิวรอลเน็ตเวิร์ก เรียนรู้ภาพตัวอักษร 3 ภาพ คือ ว, อ และ ฮ ในขั้นตอนแรกจะต้องเตรียมข้อมูลชุดฝึก (Training set) เพื่อใช้ในการเรียนรู้ภาพตัวอักษรทั้ง 3 ภาพ (ว, อ, และ ฮ) ของนิวรอลเน็ตเวิร์ก เนื่องจากข้อมูลแบบเปรียบของตัวอักษรแต่ละตัวไม่เท่ากัน ดังนั้นจึงต้องมีการใส่ข้อมูลจำลอง (Dummy input) เพื่อให้ข้อมูลเลขฐานสองที่ใช้แทนตัวอักษรแต่ละตัวมีจำนวนเท่ากัน เพื่อที่จะป้อนให้กับระดับข้อมูลเข้า (Input layer) ของนิวรอลเน็ตเวิร์กต่อไป ซึ่งการเพิ่มจำนวนของข้อมูลจำลองนั้นเป็นการชดเชยข้อมูลของตัวอักษร ที่มีข้อมูลของเลขฐานสองที่ใช้แทนภาพตัวอักษรนั้นๆ น้อยกว่าข้อมูลเลขฐานสองที่ใช้แทนภาพตัวอักษรในชุดฝึก (Training set) ที่มีค่ามากที่สุด และในตัวอย่างนี้ ตัวอักษร ฮ มีข้อมูลของเลขฐานสองที่ใช้แทนภาพมากที่สุด ดังนั้นข้อมูลเลขฐานสองของภาพตัวอักษร ว และ อ จะต้องถูกชดเชยให้มีจำนวน input เท่ากับข้อมูลเลขฐานสองของภาพตัวอักษร ฮ โดยการเพิ่ม '1' ให้ครบตามจำนวนเลขฐานสองที่ใช้แทนภาพตัวอักษร ฮ ดังรูปที่ 3.21 แสดงการสร้างชุดฝึก (Training set) และข้อมูลจำลอง (Dummy input)

ในส่วนของนิวรอลเน็ตเวิร์กนั้นจะประกอบด้วย 3 ระดับ คือ ระดับข้อมูลเข้า (Input layer), ระดับซ่อนตัว (Hidden layer) และระดับแสดงผล (Outpput layer) โดยที่ระดับข้อมูลเข้า (Input layer) จะมีจำนวนโหนดเท่ากับ 44 โหนด เนื่องจากตัวอักษร ฮ ที่ใช้เลขฐานสองในการแทนภาพข้อมูลมีจำนวนมากที่สุดเท่ากับ 44 ตำแหน่ง และจำนวนโหนดในระดับแสดงผล (Output layer) จะมีจำนวนเท่ากับจำนวนตัวอักษรที่ต้องการรู้จำ เนื่องจากตัวอักษรที่ต้องการให้ระบบนิวรอลเน็ตเวิร์กรู้จำมี 3 ตัวอักษร (ว, อ, ฮ) ดังนั้นจำนวนโหนดของระดับแสดงผล (Output layer) จึงเท่ากับ 3 โหนด ในส่วนของจำนวนโหนดในระดับซ่อนตัว (Hidden layer) นั้น ไม่สามารถที่จะกำหนดจำนวนที่แน่นอนได้ เพราะถ้าให้จำนวนโหนดในระดับซ่อนตัว (Hidden layer) มากก็จะเสียเวลาในการประมวลผลมากเนื่องจากจำนวนการเชื่อมต่อของน้ำหนักระหว่างโหนด (Connection weight) จะมากตามไปด้วยแต่การเรียนรู้ของระบบอาจจะเร็วขึ้น ในทางกลับกันถ้ากำหนดจำนวนโหนดในระดับซ่อนตัว (Hidden layer) น้อยเกินไประบบอาจจะไม่สามารถให้ผลที่ถูกต้องได้ (ไม่สามารถเบนเข้าสู่ข้อผิดพลาดที่น้อยที่สุดได้) ในการทดลองหาจำนวนโหนดที่เหมาะสมในระดับซ่อนตัว (Hidden layer) นั้นจะเริ่มต้นจากจำนวนโหนดค่าใดค่าหนึ่งก่อน จากนั้นจะทำการฝึกระบบถ้านิวรอลเน็ตเวิร์กสามารถให้ผลที่ถูกต้องได้ก็จะลดจำนวนโหนดในระดับซ่อนตัว (Hidden layer) ลงไปเรื่อยๆ แต่ถ้าถ้านิวรอลเน็ตเวิร์กไม่สามารถให้ผลที่ถูกต้องได้ หรือใช้เวลาในการฝึกนานจนเกินไปก็จะค่อยๆเพิ่มจำนวนโหนดในระดับซ่อนตัว (Hidden layer) ขึ้นตามลำดับ ในทางปฏิบัติแล้วจะใช้จำนวนโหนดในระดับซ่อนตัว (Hidden layer) น้อยที่สุดเท่าจะเป็นไปได้ สมมุติว่าจำนวนโหนดในระดับซ่อนตัว (Hidden layer) จำนวน 16 โหนด เพียงพอในการให้ระบบนิวรอลเน็ตเวร์กรู้จำตัวอักษร 3 ภาพ ดังนั้นจะได้ระบบนิวรอลเน็ตเวิร์กที่ใช้สำหรับรู้จำตัวอักษร ว, อ, และ ฮ ดังรูปที่ 3.22



รูปที่ 3.22 แสดงตัวอย่างนิวรอลเน็ตเวิร์กที่ใช้สำหรับรู้จำตัวอักษร ว,อ, และ ฮ

ในการฝึก (training) นั้นข้อมูลเลขฐานสองของภาพตัวอักษรแต่ละตัวจะถูกป้อนให้กับระดับข้อมูลเข้า (Input layer) ของนิวรอลเน็ตเวิร์กเพื่อคำนวณหาผล (output) จากนั้นจะทำการเปรียบเทียบผลที่ได้ (actual output) กับผลที่ต้องการ (desired output) เพื่อคำนวณหาความผิดพลาดระหว่างผลที่ได้กับผลที่ต้องการ และทำการปรับปรุงน้ำหนักการเชื่อมต่อ (connection weight) ของระบบเพื่อให้ผลที่ถูกต้อง และมีค่าความผิดพลาดของระบบที่อยู่ภายในขอบเขตของความผิดพลาดที่กำหนด (error threshold) โดยที่ระบบนิวรอลเน็ตเวร์กจะทำการซ้ำ (iteration) จนกว่าจะได้ผลที่ต้องการ ดังรูปที่ 3.24 แสดงผังไหลการทำงานของ back-propagation learning จากตัวอย่างนี้เป็นการเรียนรู้ภาพตัวอักษรเพียง 3 ภาพ และภาพละ 1 รูปแบบ (font) เท่านั้น ถ้าต้องการให้ระบบนิวรอลเน็ตเวิร์กนี้เรียนรู้ภาพตัวอักษรในหลายๆรูปแบบ (multi-font) หรือต้องการเพิ่มภาพตัวอักษรให้ระบบการรู้จำ ก็เพียงแต่เพิ่มข้อมูลในชุดฝึก และให้ระบบเรียนรู้ข้อมูลจากชุดฝึกใหม่ เช่นถ้าต้องการให้ระบบสามารถรู้จำภาพตัวอักษรตัวเอียงก็เพียงแต่เพิ่มข้อมูลของภาพตัวอักษรตัวเอียง (*ว, อ* และ *ฮ*) ให้กับชุดฝึก ดังรูปที่ 3.23.ก แสดงตัวอย่างข้อมูลในชุดฝึกสำหรับการรู้จำภาพตัวอักษร ว, อ, และ ฮ และรูปที่ 3.23.ข แสดงตัวอย่างข้อมูลในชุดฝึกสำหรับการรู้จำภาพตัวอักษร ว, อ, ฮ, *ว, อ* และ *ฮ* ตามลำดับ

Training Data

Training Item # 1	000110010100001100100001100111111111111111111	input
ภาพตัวอักษร ว	001	output
Training Item # 2	000100000100001100010000011101100101101111111	input
ภาพตัวอักษร อ	010	output
Training Item # 3	001000000000100100100001000000111011001011011	input
ภาพตัวอักษร ฮ	100	output

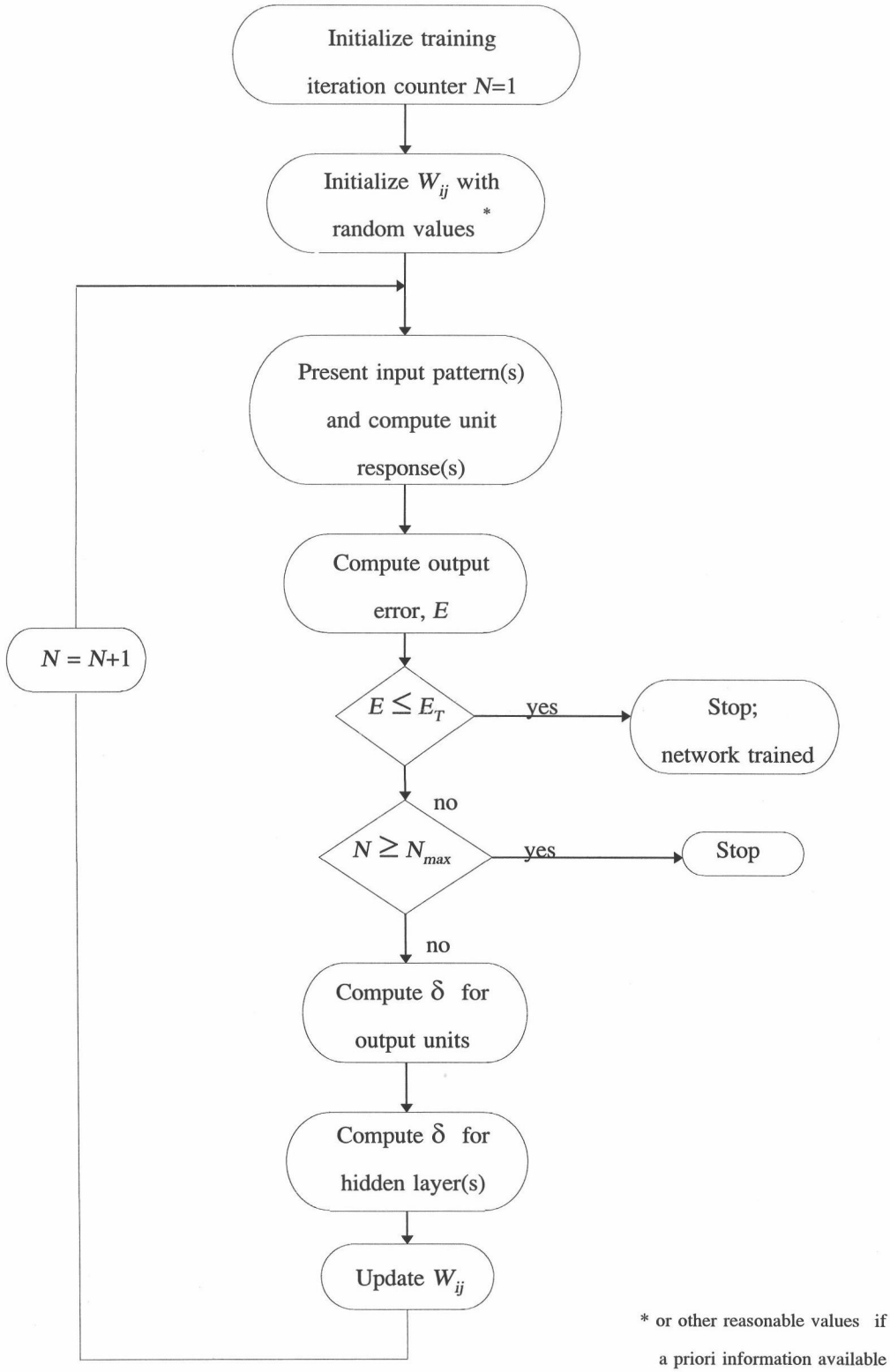
รูปที่ 3.23.ก แสดงข้อมูลชุดฝึกสำหรับการรู้จำภาพตัวอักษร ว, อ, และ ฮ

Training Data

Training Item # 1	000110010100001100100001100111111111111111111	input
ภาพตัวอักษร ว	001	output
Training Item # 2	000100000100001100010000011101100101101111111	input
ภาพตัวอักษร อ	010	output
Training Item # 3	001000000000100100100001000000111011001011011	input
ภาพตัวอักษร ฮ	100	output
Training Item # 4	000100000100001100011001111111111111111111111	input
ภาพตัวอักษร ว	001	output
Training Item # 5	000100000100001100010000011101100101010010011	input
ภาพตัวอักษร อ	010	output
Training Item # 6	001000000000100100100001000000111011001011011	input
ภาพตัวอักษร ฮ	100	output

รูปที่ 3.23.ข แสดงข้อมูลชุดฝึกสำหรับการรู้จำภาพตัวอักษร ว, อ, ฮ,ว, อ และ ฮ

รูปที่ 3.23 แสดงข้อมูลชุดฝึกสำหรับการรู้จำภาพตัวอักษร



รูปที่ 3.24 แสดงผังไหลของการทำงานของ back-propagation learning

ตัวแปรที่สำคัญในการฝึกหัดระบบนิวรอลเน็ตเวิร์กคือ learning rate และ momentum โดยที่ถ้า learning rate มีค่าน้อยๆจะทำให้จำนวนรอบของการทำซ้ำ (iteration) มากขึ้นตามไปด้วย แต่จะสามารถให้ผลที่ถูกต้องมากขึ้น เนื่องจาก learning rate เปรียบเสมือน step ในการทำซ้ำ ในทางกลับกันถ้าค่า learning rate มากจะทำให้ระบบบรรลุผล (convergence) เร็วขึ้นแต่ระบบอาจจะไม่สามารถให้คำตอบที่ถูกต้องได้ (เกิดการ oscillate) ส่วน momentum จะเป็นส่วนช่วยให้ระบบบรรลุผล (convergence) เร็วขึ้นโดยการนำอัตราการเปลี่ยนแปลงของน้ำหนักการเชื่อมต่อ (weight) ในครั้งก่อนมาคิดคำนวณด้วย เพื่อช่วยให้การเปลี่ยนแปลงของน้ำหนักการเชื่อมต่อไปในทิศทางเดียวกัน โดยปกติแล้วการกำหนดจำนวนของ learning rate และ momentum ที่ดีที่สุดจะขึ้นอยู่กับลักษณะของ surface error (E vs. w_{ij}) เช่น ถ้ามีการเปลี่ยนแปลงของ surface error อย่างรวดเร็ว learning rate ควรจะมีค่าน้อยๆ ในทางกลับกันถ้า surface error มีลักษณะราบเรียบจำนวนของ learning rate ควรจะมีค่ามากขึ้นเพื่อให้ระบบบรรลุผลเร็วขึ้น