บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

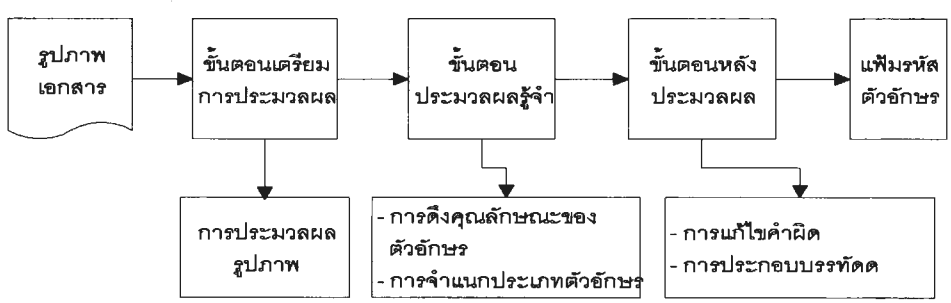
บทนี้แบ่งเนื้อหาออกเป็นสองส่วน โดยเนื้อหาในส่วนแรกจะกล่าวถึงทฤษฎีที่เกี่ยวข้อง และเนื้อหาในส่วนที่สองจะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับงานวิจัยนี้

2.1 ทฤษฎีที่เกี่ยวข้อง

เนื้อหาบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้อง ดังต่อไปนี้ คือ หลักการทำงานของโปรแกรมโอซีอาร์ การวิเคราะห์องค์ประกอบสำคัญ นิวรอลเน็ตเวิร์ก ข่ายงานชั้นเดียว นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า และนิวรอลเน็ตเวิร์กแบบวกกลับ มีรายละเอียดดังนี้

2.1.1 หลักการทำงานของโปรแกรมโอซีอาร์ (Optical Character Recognition หรือ OCR) [2]

โปรแกรมโอซีอาร์ แบ่งการทำงานออกเป็น 3 ส่วน ดังรูปที่ 2.1



รูปที่ 2.1 กระบวนการทำงานของโปรแกรมโอซีอาร์

ในแต่ละขั้นตอนมีหลักการคร่าว ๆ ดังนี้

2.1.1.1 ขั้นตอนเตรียมการประมวลผล (Pre-processing) ในขั้นตอนนี้นำรูปภาพเอกสารมาปรับปรุงคุณลักษณะและคุณสมบัติต่าง ๆ ของรูปภาพตัวอักษร โดยกระบวนการทางการประมวลผลรูปภาพ (Image Processing) เพื่อให้ได้รูปภาพที่เหมาะสมกับขั้นตอนประมวลผลรู้จำ

2.1.1.2 ขั้นตอนประมวลผลรู้จำ (OCR Processing) เป็นขั้นตอนหลักของโปรแกรมโอซีอาร์ ขั้นตอนนี้ทำการรู้จำตัวอักษร ซึ่งประกอบด้วยการทำงาน 2 ส่วนหลัก ๆ ดังนี้

(1) การดึงคุณลักษณะ (Feature Extraction) ขั้นตอนนี้นำรูปภาพตัวอักษรที่ได้จากขั้นตอนเตรียมการประมวลผล มาหาส่วนที่เป็นตัวอักษร และดึงคุณลักษณะต่าง ๆ ของรูปภาพตัวอักษร เพื่อหาคุณลักษณะที่เหมาะสมกับตัวอักษร

(2) การจำแนกประเภท (Classification) ตัวอักษร ขั้นตอนนี้นำรูปภาพตัวอักษรที่ได้จากขั้นตอนการดึงคุณลักษณะ มาใช้ในการเรียนรู้และการรู้จำ

2.1.1.3 ขั้นตอนหลังการประมวลผล (Post Processing) เป็นขั้นตอนที่ช่วยในการปรับปรุงผลลัพธ์ที่ได้จากขั้นตอนประมวลผลรู้จำ โดยการแก้ไขคำที่สะกดผิด (Spelling Corrector) เพื่อให้ข้อมูลมีความถูกต้องมากยิ่งขึ้น

2.1.2 การวิเคราะห์องค์ประกอบสำคัญ (Principal Component Analysis หรือ PCA) [4,10]

การประมวลผลรู้จำ ต้องจัดการกับข้อมูลภาพขนาดใหญ่ มีคุณลักษณะจำนวนมาก ส่งผลกระทบกับเวลาในการประมวลผล ซึ่งใช้เวลาในการทำงานค่อนข้างมาก วิธีการหนึ่งที่มาใช้เพื่อช่วยลดขนาดของข้อมูล และเพิ่มประสิทธิภาพในการทำงาน คือ การวิเคราะห์องค์ประกอบสำคัญ

การวิเคราะห์องค์ประกอบสำคัญ เป็นวิธีการดึงคุณลักษณะ หรือคุณสมบัติสำคัญจากข้อมูลที่สนใจออกมา การทำงานอยู่บนหลักการแปลงเชิงตั้งฉากของพิกัด (Orthogonal Transformation) โดยค่าเส้นพิกัดใหม่ ถูกเรียกว่า องค์ประกอบสำคัญ (Principal Components)

ประโยชน์ขององค์ประกอบสำคัญที่ได้ คือ ลดข้อมูลที่มีมิติมากให้มีขนาดเล็กลง โดยข้อมูลที่ได้มีลักษณะสำคัญภายในไม่แตกต่างจากข้อมูลเดิมมากนัก และทำให้ง่ายต่อการคำนวณทางคณิตศาสตร์มากขึ้น

กระบวนการวิเคราะห์องค์ประกอบสำคัญของข้อมูล เริ่มจากนำข้อมูลรูปภาพตัวอักษรที่ได้ผ่านการแปลงเป็นข้อมูลเวกเตอร์ มาหาเวกเตอร์ค่าเฉลี่ย (Mean Vector) ดังสมการ (2.1)

$$\mathbf{m}_x = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i \qquad (2.1)$$

เมื่อ $\mathbf{m}_x$ แทน เวกเตอร์ค่าเฉลี่ยของเวกเตอร์รูปภาพตัวอักษร (ในการทดลองในบทที่ 3 มีขนาด 1,024 x 1)

$\mathbf{x}_i$ แทน เวกเตอร์รูปภาพตัวอักษร (ในการทดลองในบทที่ 3 มีขนาด 1,024 x 1)

M แทน จำนวนเวกเตอร์ของรูปภาพตัวอักษรทั้งหมดที่ใช้ขณะเรียนรู้

เมื่อหาเวกเตอร์ค่าเฉลี่ยได้แล้ว นำเวกเตอร์ค่าเฉลี่ยมาใช้หาค่าเมตริกซ์โคแวเรียน (Covariance Matrix) ที่แสดงถึงความแปรปรวนของข้อมูลทั้งหมด ดังสมการ (2.2)

$$\mathbf{C}_x = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{m}_x\mathbf{m}_x^T \qquad (2.2)$$

เมื่อ $\mathbf{C}_x$ แทน เมตริกซ์โคแวเรียน (ในการทดลองในบทที่ 3 มีขนาด 1,024 x 1,024)

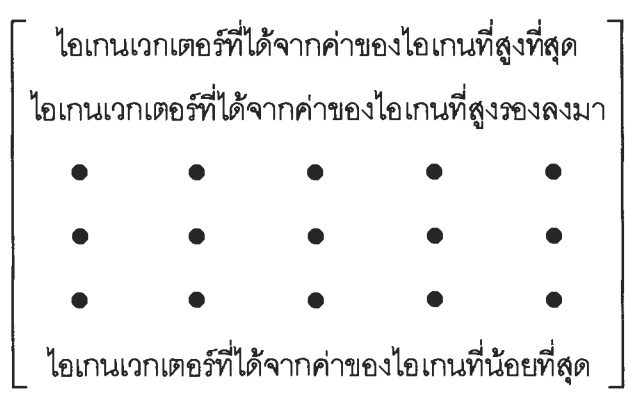
$\mathbf{x}_i^T$ แทน ทรานสโพสเมตริกซ์ของ $\mathbf{x}_i$ (ในการทดลองในบทที่ 3 มีขนาด 1 x 1,024)

$\mathbf{m}_x^T$ แทน ทรานสโพสเมตริกซ์ของ $\mathbf{m}_x$ (ในการทดลองในบทที่ 3 มีขนาด 1 x 1,024)

นำเมตริกซ์โคแวเรียน $\mathbf{C}_x$ ไปใช้ในการหาเมตริกซ์ไอเกนเวกเตอร์ (Eigenvector Matrix) ซึ่งการคำนวณเมตริกซ์ไอเกนเวกเตอร์นั้น ทำได้โดยการหาค่าของไอเกน (Eigenvalue หรือ λ) และไอเกนเวกเตอร์ (Eigenvector หรือ $\mathbf{v}$) ดังสมการ (2.3)

$$\lambda\mathbf{v} = \mathbf{C}_x\mathbf{v} \tag{2.3}$$

นำไอเกนเวกเตอร์ที่ได้จากสมการ (2.3) มาสร้างเมตริกซ์ $\mathbf{A}$ โดยไอเกนเวกเตอร์ที่ได้จากค่าของไอเกนสูงสุดเก็บไว้ที่แถวแรก และไอเกนเวกเตอร์ที่ได้จากค่าของไอเกนรองลงมาเก็บไว้ที่แถวถัดไป ตามลำดับ จนกระทั่งแถวสุดท้าย คือ ไอเกนเวกเตอร์ที่ได้จากค่าของไอเกนน้อยสุด ดังรูปที่ 2.2 และเมตริกซ์ที่ได้ถูกนำไปใช้ในการหาเวกเตอร์รูปแบบของรูปภาพตัวอักษรเพื่อแสดงลักษณะสำคัญของตัวอักษร ดังสมการ (2.4)



รูปที่ 2.2 ลักษณะเมตริกซ์ของไอเกนเวกเตอร์

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m}_x) \tag{2.4}$$

เมื่อ $\mathbf{y}$ แทน เวกเตอร์รูปแบบ

$\mathbf{A}$ แทน เมตริกซ์ของการแปลง

$\mathbf{x}$ แทน ข้อมูลภาพ

$\mathbf{m}_x$ แทน เวกเตอร์ค่าเฉลี่ย

หลักของการวิเคราะห์องค์ประกอบสำคัญ จะพิจารณาว่า ค่าของไอเกนที่มีค่าสูงจะเกี่ยวข้องกับตัวอักษรต้นแบบมากกว่าค่าของไอเกนที่มีค่าต่ำ ดังนั้นเราจะเลือกใช้ไอเกนเวกเตอร์เพียง k ตัวแรก (ในบทที่ 3 ใช้ค่า k=250) ทำให้ได้เมตริกซ์ขนาด (k x N) เมื่อ N คือ จำนวนมิติของเวกเตอร์ภาพ และเวกเตอร์ **y** ที่ได้ มีขนาด k ส่วนแถวที่เหลือ คือ แถวที่ k +1 ถึง N มีค่าเป็นศูนย์ ทำให้ลดเวลาในการคำนวณ เวกเตอร์รูปแบบที่ได้ถูกส่งไปเป็นอินพุตเพื่อใช้ในการจำแนกประเภทตัวอักษรต่อไป

2.1.3 นิวรอลเน็ตเวิร์ก (Neural Network) [11-14]

นิวรอลเน็ตเวิร์กเป็นวิธีการเรียนรู้ของเครื่องแบบหนึ่ง ที่เลียนแบบมาจากการทำงานของเซลล์สมองมนุษย์ และพัฒนาขึ้นโดยมีสมมติฐานที่ว่า

(1) การประมวลผลข้อมูลต่าง ๆ เกิดขึ้นที่ส่วนประกอบเล็ก ๆ จำนวนมาก เรียกว่า นิวรอน (Neuron)

(2) การทำงานของนิวรอนจะส่งสัญญาณผ่านทางเส้นเชื่อมต่อระหว่างนิวรอน ที่เรียกว่า คอนเนกชันลิงค์ (Connection Link)

(3) เส้นเชื่อมแต่ละเส้นจะมีค่าน้ำหนัก (Weight) ที่แตกต่างกัน

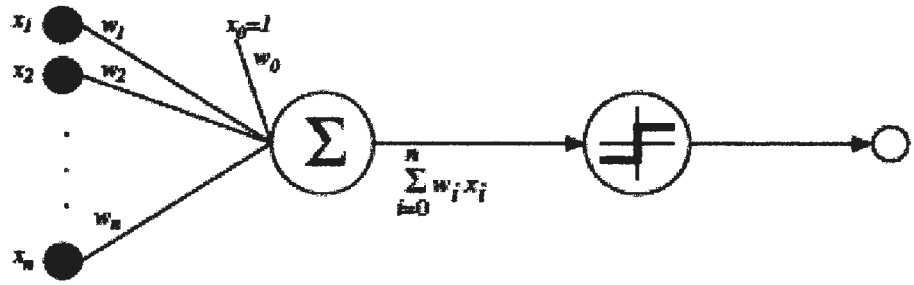
(4) นิวรอนแต่ละตัวได้รับข้อมูลจากนิวรอนอื่น ๆ ผ่านทางฟังก์ชันกระตุ้น (Active Function)

การทำงานของนิวรอลเน็ตเวิร์กประกอบด้วยหน่วยประมวลผลข้อมูล หรือนิวรอน นิวรอนแต่ละตัวเชื่อมต่อกันผ่านทางเส้นเชื่อม เส้นเชื่อมแต่ละเส้นมีค่าน้ำหนักที่แตกต่างกัน ในการทำงานของนิวรอลเน็ตเวิร์กแบ่งการทำงานออกเป็นชั้น ๆ (Layer) โดยทำการประมวลผลในชั้นตัวเองก่อน จากนั้นส่งผลลัพธ์ไปประมวลผลในชั้นถัด ๆ ไป จนครบทุกชั้น เมื่อเสร็จแล้วจะได้ผลลัพธ์ของนิวรอลเน็ตเวิร์ก สำหรับชั้นของเน็ตเวิร์กแบ่งออกเป็น 3 ชั้น คือ ชั้นอินพุต (Input Layer) ชั้นซ่อน (Hidden Layer) ซึ่งอาจมีมากกว่า 1 ชั้น และชั้นเอาท์พุต (Output Layer) ทุก ๆ เน็ตเวิร์กต้องประกอบด้วยชั้นอินพุตและชั้นเอาท์พุต ส่วนชั้นซ่อนจะมีหรือไม่มีก็ได้

การแบ่งประเภทของนิวรอลเน็ตเวิร์กตามจำนวนชั้นของการทำงานแบ่งได้ 2 แบบ คือ แบบข่ายงานชั้นเดียว (Single Layer Network) และแบบข่ายงานหลายชั้น (Multilayer Network) ข่ายงานหลายชั้นที่ใช้ในงานวิจัยนี้จะใช้นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้าและนิวรอลเน็ตแบบวกกลับ การนับจำนวนชั้นของนิวรอลเน็ตเวิร์ก จะนับชั้นซ่อนและชั้นเอาท์พุต ส่วนชั้นอินพุตนั้นบางทีก็นับบางทีก็ไม่นับ สำหรับในงานวิจัยนี้จะไม่นับชั้นอินพุต

2.1.3.1 ข่ายงานชั้นเดียว (Single Layer Network) [13]

นิวรอลเน็ตเวิร์กที่มีลักษณะข่ายงานชั้นเดียวจะมีการรับสัญญาณและหาผลลัพธ์ภายในชั้น ตัวอย่างของนิวรอลเน็ตเวิร์กที่มีลักษณะเช่นนี้ เช่น เพอร์เซปตรอนอย่างง่าย (Simple Perceptron) ชั้นอินพุตและชั้นเอาท์พุต มีการเชื่อมโยงกันโดยตรง ดังรูปที่ 2.3



รูปที่ 2.3 เพอร์เซปตรอนอย่างง่าย

การทำงานของเพอร์เซปตรอน เริ่มตั้งแต่รับอินพุตเข้ามาเป็นเวกเตอร์จำนวนจริง แล้วหาผลรวมเชิงเส้น (Linear Combination) ของอินพุต นำผลรวมที่ได้ไปเปรียบเทียบกับค่าขีดแบ่ง (Threshold) ที่กำหนด เพื่อหาเอาท์พุตออกมา โดยเอาท์พุตจะเป็น 1 ถ้าผลรวมที่ได้มีค่าเกินค่าขีดแบ่ง และเอาท์พุตเป็น -1 ถ้าผลรวมที่ได้ไม่เกินค่าขีดแบ่ง ดังสมการ (2.5)

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0 \\ -1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n < 0 \end{cases} \quad (2.5)$$

เมื่อ $-w_0$ แทน ค่าขีดแบ่ง

w_i แทน ค่าน้ำหนักของอินพุต i มีค่าเป็นจำนวนจริง

x_i แทน ค่าอินพุต i

o แทน ค่าเอาท์พุตของเพอร์เซปตรอน

เราอาจมองได้ว่าระนาบตัดสินใจ (Decision Surface) ของเพอร์เซปตรอน คือ ระนาบใน n มิติ โดยเพอร์เซปตรอนจะให้เอาท์พุตของตัวอย่างบวกอยู่ด้านหนึ่งของระนาบ และเอาท์พุตของตัวอย่างลบอยู่อีกด้านหนึ่งของระนาบ ในกรณีที่สามารถใช้ระนาบในการแบ่งเซตของตัวอย่างออกเป็นตัวอย่างบวกและตัวอย่างลบได้ เรียกเซตของตัวอย่างที่มีลักษณะนี้ว่า เซตของตัวอย่างที่สามารถแบ่งแยกได้แบบเชิงเส้น (Linearly Separable)

เราสามารถใช้เพอร์เซปตรอนเดี่ยว (Single Perceptron) ในการแสดงฟังก์ชันบูลีน (Boolean Function) บางชนิดได้ เช่น AND OR NAND (¬AND) และ NOR (¬OR) กำหนดให้

ค่าของบูลีนที่เป็นจริงแทน 1 และค่าของบูลีนที่เป็นเท็จแทน -1 แต่มีฟังก์ชันบูลีนบางฟังก์ชันที่ไม่สามารถแสดงด้วยเพอร์เซปตรอนเชิงเดี่ยวได้ เช่น ฟังก์ชัน XOR ดังรูปที่ 2.4

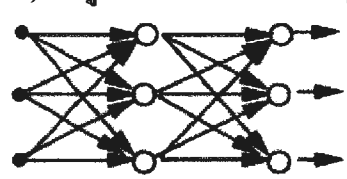
(a) (b)

รูปที่ 2.4 ระนาบตัดสินใจ (a) แบ่งแยกได้แบบเชิงเส้น (b) ไม่สามารถแบ่งแยกได้แบบเชิงเส้น

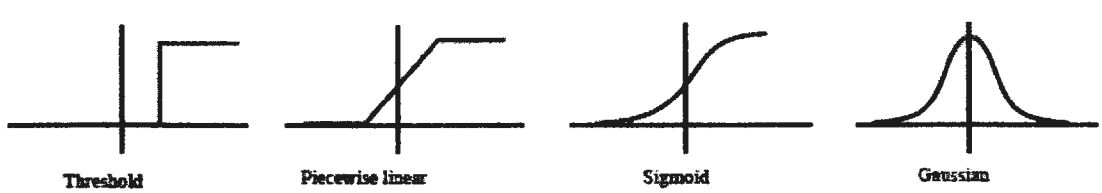
2.1.3.2 นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า (Feed-Forward Neural Network) [12-14]

เนื่องจากข้อจำกัดของข่ายงานชั้นเดียว ที่ไม่สามารถนำมาใช้ในการแบ่งตัวอย่างที่มีลักษณะแบบไม่สามารถแบ่งได้แบบเชิงเส้น วิธีแก้คือนำข่ายงานชั้นเดียวมาต่อกัน เรียกข่ายงานใหม่นี้ว่า ข่ายงานหลายชั้น วิธีการนี้สามารถแก้ปัญหาที่ไม่สามารถแก้ได้ด้วยข่ายงานชั้นเดียว ทำให้เห็นถึงความสามารถของข่ายงานหลายชั้น จึงถูกนำมาใช้กับงานหลายชนิด เช่น การรู้จำเสียงพูด (Speed Recognition) เป็นต้น

นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า เป็นเน็ตเวิร์กแบบหลายชั้นซึ่งเส้นเชื่อมจะโยงจากชั้นอินพุตไปยังชั้นซ่อนแล้วไปยังชั้นเอาท์พุตโดยไม่มีการย้อนกลับ ดังรูปที่ 2.5 และเป็นเน็ตเวิร์กที่มีความสามารถในการสร้างระนาบตัดสินใจแบบไม่เป็นเชิงเส้น (Nonlinear Decision Surface) ซึ่งสามารถแบ่งแยกตัวอย่างได้ดีกว่าระนาบตัดสินใจแบบเชิงเส้น (Linear Decision Surface) ทั้งนี้เนื่องมาจากชนิดของฟังก์ชันกระตุ้น (Active Function) ที่สามารถหาอนุพันธ์ได้ เช่น ฟังก์ชันซิกมอยด์ (Sigmoid Function) ดังรูปที่ 2.6 ที่ให้ค่าเอาท์พุตอยู่ระหว่าง 0 และ 1



รูปที่ 2.5 นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า



รูปที่ 2.6 องค์ประกอบของฟังก์ชันกระตุ้นแบบต่าง ๆ

ฟังก์ชันซิกมอยด์จะหาผลรวมเชิงเส้นของอินพุต และนำผลที่ได้ไปเปรียบเทียบกับค่าขีดแบ่ง โดยเอาท์พุตเป็นฟังก์ชันต่อเนื่อง (Continuous Function) ของอินพุต ดังสมการ (2.6)

$$o = \sigma(\vec{w} \cdot \vec{x}) \tag{2.6}$$

เมื่อ $$\sigma(y) = \frac{1}{1 + e^{-y}}$$

เมื่อ o แทน เอาท์พุต

$\vec{x}$ แทน อินพุต

$\vec{w}$ แทน ค่าน้ำหนักของอินพุต

σ แทน ฟังก์ชันซิกมอยด์ ที่ให้ค่าเอาต์พุตอยู่ระหว่าง 0 และ 1

คุณสมบัติที่ดีอย่างหนึ่งของฟังก์ชันซิกมอยด์ คือ สามารถหาอนุพันธ์ของฟังก์ชันได้ง่าย มีฟังก์ชันบางประเภทที่มีลักษณะเช่นเดียวกันกับฟังก์ชันซิกมอยด์ เช่น ฟังก์ชันไฮเปอร์โบลิกแทนเจนต์ (Hyperbolic Tangent Function) ที่ให้ค่าเอาต์พุตอยู่ระหว่าง -1 และ 1 เป็นต้น

อัลกอริทึมการแพร่กระจายย้อนกลับ (Backpropagation Algorithm) เป็นอัลกอริทึมหนึ่งที่ได้รับความนิยม อัลกอริทึมนี้จะเรียนรู้ค่าเวกเตอร์น้ำหนัก โดยการปรับค่าเกรเดียนเดสเซนต์ (Gradient Descent) เพื่อลดค่าผิดพลาดกำลังสอง (Square Error) ระหว่างเอาท์พุตของเน็ตเวิร์กกับเอาท์พุตเป้าหมาย ถ้าค่าความผิดพลาดที่ได้ยังไม่ถึงจุดต่ำสุด หรือยังไม่อยู่ในเกณฑ์ที่พอยอมรับได้ จะปรับค่าน้ำหนักของตัวอย่างสอนและคำนวณค่าน้ำหนักใหม่ในรอบถัดไป โดยฟังก์ชันกระตุ้นที่ใช้ คือ ฟังก์ชันซิกมอยด์ และการปรับค่าน้ำหนักของตัวอย่างสอนจะปรับทีละ 1 ตัวอย่าง ซึ่งมีขั้นตอนวิธีในการปรับค่าน้ำหนัก ดังนี้

กำหนดให้ตัวอย่างแต่ละตัวที่ใช้ในการเรียนรู้อยู่ในรูป $\langle \vec{x}, \vec{t} \rangle$

เมื่อ $\vec{x}$ แทน อินพุตเวกเตอร์ของนิวรอลเน็ตเวิร์ก

$\vec{t}$ แทน เอาท์พุตเวกเตอร์ของนิวรอลเน็ตเวิร์ก

η แทน อัตราการเรียนรู้ (Learning Rate)

x_{ji} แทน อินพุตขององค์ประกอบ j ซึ่งมาจาก i

w_{ji} แทน ค่าน้ำหนักขององค์ประกอบ j ซึ่งมาจาก i

1. สร้างนิวรอลเน็ตเวิร์กตามโครงสร้างที่ต้องการ กำหนดจำนวนโหนดในแต่ละชั้น
2. กำหนดค่าน้ำหนักเริ่มต้นของนิวรอลเน็ตเวิร์ก โดยวิธีการสุ่มค่า
3. ปรับค่าน้ำหนักตามขั้นตอนต่อไปนี้

สำหรับตัวอย่าง $\langle \vec{x}, \vec{t} \rangle$ ที่ใช้ในการเรียนรู้แต่ละตัวอย่าง ทำตามขั้นตอนดังนี้

(1) รับตัวอย่าง $\vec{x}$ เข้าไปในนิวรอลเน็ตเวิร์ก แล้วคำนวณค่าเอาท์พุต o_u ของทุก ๆ นิวรอนในนิวรอลเน็ตเวิร์ก

(2) คำนวณค่าความคลาดเคลื่อน δ_k สำหรับเอาท์พุต k

$$\delta_k \leftarrow o_k(1-o_k)(t_k - o_k)$$

(3) คำนวณค่าความคลาดเคลื่อน δ_h สำหรับชั้นซ่อน h

$$\delta_h \leftarrow o_h(1-o_h) \sum_{k \in outputs} w_{kh}\delta_k$$

(4) ปรับค่าน้ำหนัก w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

เมื่อ $\Delta w_{ji} = \eta \delta_i x_{ji}$

2.1.3.3 นิวรอลเน็ตเวิร์กแบบวกกลับ (Recurrent Neural Network)[13-14]

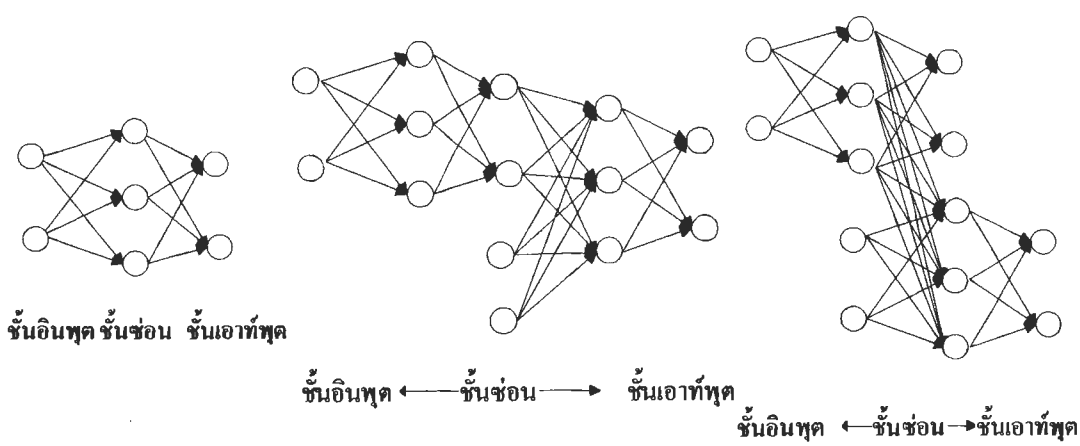
นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้าในหัวข้อ 2.1.3.2 เป็นนิวรอลเน็ตเวิร์กที่ไม่มีการนำข้อมูลในอดีตมาใช้ สำหรับปัญหาที่ต้องใช้ข้อมูลในอดีตช่วยในการพิจารณานั้น เราจึงจำเป็นต้องใช้นิวรอลเน็ตเวิร์กแบบวกกลับ ตัวอย่างเช่น การทำนายราคาหุ้นในตลาดหุ้น ราคาของหุ้นในวันพรุ่งนี้ แทนด้วย y(t+1) ขึ้นอยู่กับข้อมูลข่าวสารวันนี้ แทนด้วย x(t) และข้อมูลของเมื่อวาน แทนด้วย x(t-1) ถ้าใช้นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า ดังรูปที่ 2.7 (a) จะไม่สามารถนำข้อมูลของเมื่อวานนี้มาใช้ในการทำนายราคาหุ้นในวันนี้ ดังนั้นเราจึงเพิ่มโหนด b เข้าไปในชั้นซ่อนและเพิ่มอินพุตโหนด c(t) เข้าไปในรูปที่ 2.7 (a) ผลการเพิ่มโหนดแสดงดังรูป 2.7 (b) จะพบว่า อินพุตโหนด c(t) มาจากโหนด b ที่เวลา t-1 และอินพุตโหนด c(t-1) มาจากโหนด b ที่เวลา t-2 แสดงว่าอินพุตโหนด c(t) ของเน็ตเวิร์กได้มาจากโหนด b ในขั้นตอนก่อนหน้า

รูปที่ 2.7 (a) นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า (b) นิวรอลเน็ตเวิร์กแบบวกกลับ

นิวรอลเน็ตเวิร์กที่ช่วยแก้ปัญหาข้างต้น เรียกว่า นิวรอลเน็ตเวิร์กแบบวกกลับ เป็นนิวรอลเน็ตเวิร์กที่มีการนำข้อมูลในขั้นตอนก่อนหน้าป้อนกลับเข้าไปเน็ตเวิร์กในขั้นตอนถัดไป นิวรอลเน็ตเวิร์กแบบจอร์แดน (Jordan) [15] และเอลแมน (Elman) [16] เป็นนิวรอลเน็ตเวิร์กแบบวกกลับที่ได้รับความนิยม จอร์แดนเป็นเน็ตเวิร์กที่มีการนำข้อมูลจากชั้นเอาท์พุตในขั้นตอนก่อนหน้าป้อนกลับเข้าไปนิวรอลเน็ตเวิร์กในขั้นตอนถัดไป ดังรูปที่ 2.8 (a) ส่วนเอลแมนเป็นเน็ตเวิร์กที่นำข้อมูลจากชั้นซ่อนในขั้นตอนก่อนหน้าป้อนกลับไปในนิวรอลเน็ตเวิร์กในขั้นตอนถัดไป ดังรูป 2.8 (b)

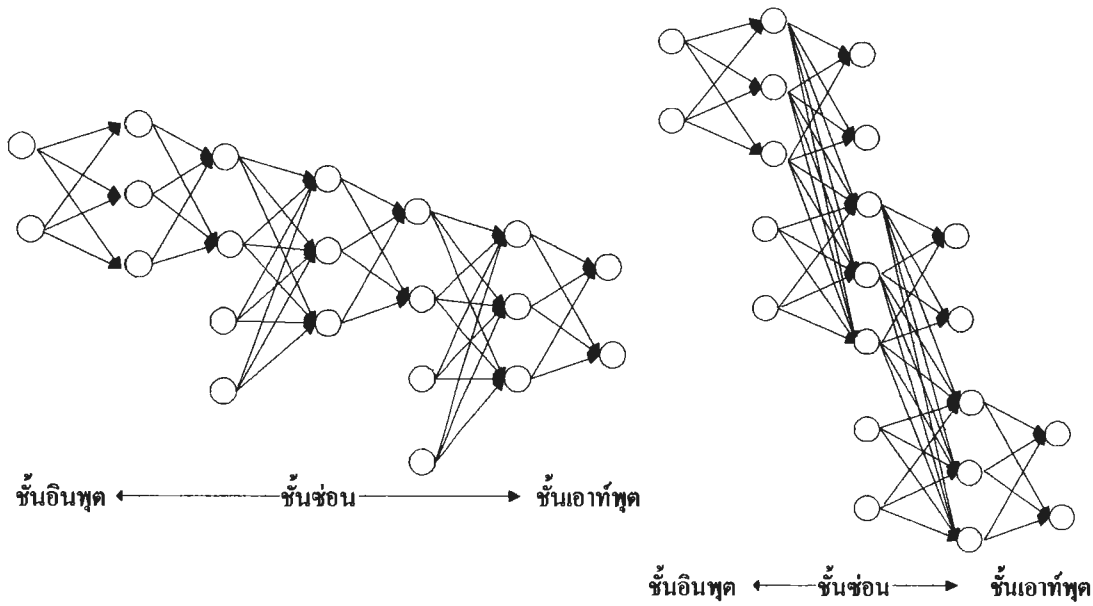
รูปที่ 2.8 (a) จอร์แดน (Jordan) (b) เอลแมน (Elman)

สมมติว่าถ้าในชั้นอินพุต ชั้นซ่อนและชั้นเอาท์พุตประกอบด้วยโหนด 2 3 และ 2 โหนด ตามลำดับ ถ้านิวรอลเน็ตเวิร์กที่เลือกใช้เป็นนิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้าโครงสร้างที่ได้จะปรากฏ ดังรูปที่ 2.9 (a) แต่ถ้าเป็นนิวรอลเน็ตเวิร์กแบบวกกลับจะต้องพิจารณาว่าวกกลับกี่รอบ เช่น ถ้าวกกลับ 1 รอบ โครงสร้างของจอร์แดนจะปรากฏดังรูปที่ 2.9 (b) และโครงสร้างของเอลแมนจะปรากฏดังรูปที่ 2.9 (c) ถ้าวกกลับ 2 รอบ โครงสร้างของจอร์แดนจะปรากฏดังรูปที่ 2.10 (a) และโครงสร้างของเอลแมนจะปรากฏดังรูปที่ 2.10 (b)



(a) (b) (c)

รูปที่ 2.9 (a) นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า (b) จอร์แดนที่มีการวกกลับ 1 รอบ (c) เอลแมนที่มีการวกกลับ 1 รอบ



รูปที่ 2.10 (a) จอร์แดนที่มีการวกกลับ 2 รอบ (b) เอลแมนที่มีการวกกลับ 2 รอบ

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยทางด้านโปรแกรมโอซีอาร์มีผู้พัฒนากันมาอย่างต่อเนื่อง ดังเช่น งานของชาญฤทธิ์ [2] ได้พัฒนาโปรแกรมโอซีอาร์ภาษาไทยเพื่อใช้ในการรู้จำตัวพิมพ์ภาษาไทย เริ่มตั้งแต่การเตรียมรูปภาพตัวอักษร พร้อมทั้งปรับปรุงคุณลักษณะสำคัญของตัวอักษรให้เหมาะสม จากนั้นจะใช้การวิเคราะห์องค์ประกอบสำคัญเพื่อช่วยในการดึงคุณลักษณะที่สำคัญของตัวอักษรออกมา ขั้นตอนประมวลผลรู้จำใช้นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้าที่มีการปรับค่าน้ำหนักโดยใช้อัลกอริทึมการแพร่กระจายย้อนกลับ นำผลลัพธ์ที่ได้ไปแก้ไขคำผิดแบบไตรแกรมของประเภทคำ ทดสอบกับข้อมูลจำนวน 71,832 ตัวอักษร ซึ่งมี 6 รูปแบบ ๆ ละ 8 ขนาด ผลการทดลองพบว่าผลการรู้จำเมื่อยังไม่ได้แก้ไขคำผิดมีความผิดพลาดเฉลี่ยร้อยละ 1.85 ผลการรู้จำหลังจากแก้ไขคำผิดที่ไม่เป็นคำ มีความผิดพลาดเฉลี่ยร้อยละ 1.47 และผลการรู้จำหลังจากแก้ไขคำผิดที่ไม่เป็นคำและคำผิดที่เป็นคำแล้วมีความผิดพลาดเฉลี่ยร้อยละ 1.50

ในส่วนการประมวลผลรู้จำนั้น วิธีในการดึงคุณลักษณะมีหลายวิธี ตัวอย่างเช่น การแมปคุณลักษณะ (Feature Map) และการใช้ค่าน้ำหนักร่วมกัน (Weight Sharing) เช่น งานวิจัยของ Cun และคณะ [3] การวิเคราะห์องค์ประกอบสำคัญ เช่น งานวิจัยของ Grother [4] ธเนศ [1] ชาญฤทธิ์ [2] ทฤษฎีการวิเคราะห์องค์ประกอบสำคัญของข้อมูลแบบเคอร์เนล (Kernel Principle Component Analysis) เช่น งานวิจัยของพัฒนชัย [7] เป็นต้น

ส่วนวิธีการจำแนกประเภทตัวอักษร ได้แก่ นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า เช่น Cun และคณะ [3] Grother [4] Lee และ Kim [9] และธเนศ [1] ใช้วิธีโปรแกรมตรรกะเชิง

อุปนัย เช่น อภิญญา [8] การเรียนรู้แบบหลายกลุ่มซัพพอร์ตเวกเตอร์แมชชีน เช่น งานวิจัยของพัฒนชัย [7] กลุ่มก้อนนิวรอลเน็ตเวิร์ก เช่น งานวิจัยของสุขวสา [5] นิวรอลเน็ตเวิร์กแบบวกกลับ เช่น งานวิจัยของ Lee และ Kim [9] เป็นต้น

งานวิจัยของสุกรี [6] ได้ประยุกต์โปรแกรมตรรกะเชิงอุปนัยและนิวรอลเน็ตเวิร์กในการรู้จำตัวพิมพ์อักษรภาษาไทย 77 ตัว จำนวน 2 รูปแบบ ๆ ละ 6 ขนาด โดยใช้ชุดข้อมูลทดลองสองชุด คือ ชุดที่หนึ่งใช้จำนวนสัญพจน์ (Literal) ที่ไม่ตรงและจำนวนสัญพจน์ที่ตรงกับตัวอย่างเป็นข้อมูลในการเรียนรู้ ส่วนชุดที่สองใช้ค่าความจริงของสัญพจน์ทุกสัญพจน์ในกฎแต่ละข้อแทน จากการทดลองข้อมูลชุดที่หนึ่งมีความถูกต้องร้อยละ 92.55 และข้อมูลชุดที่สองมีความถูกต้องร้อยละ 94.26

นอกจากงานของสุกรี [6] ซึ่งทดลองกับตัวพิมพ์อักษรภาษาไทยแล้ว มีผู้พัฒนางานด้านการรู้จำตัวพิมพ์อักษรภาษาไทย เช่น งานของอภิญญา [8] ทดลองการรู้จำตัวพิมพ์อักษรภาษาไทยกับข้อมูลสองชุด ชุดแรกทดสอบกับข้อมูลที่ไม่เคยเรียนรู้มาก่อน ผลการทดลองมีความถูกต้องร้อยละ 87.38 จากจำนวนตัวอักษร 539 ตัว ชุดที่สองทดสอบกับข้อมูลที่มีสัญญาณรบกวน ผลการทดลองมีความถูกต้องร้อยละ 87.89 จากจำนวนตัวอักษร 2,156 ตัว

งานของธเนศ [1] ทดลองกับตัวอักษรภาษาไทย 6,528 ตัว มีรูปแบบตัวอักษร 6 รูปแบบ ๆ ละ 7 ขนาด ทดลองกับข้อมูลที่มีสัญญาณรบกวนสองชุด คือ มีความเข้มของเอกสารจางลงกว่าข้อมูลสอนและมีความเข้มของเอกสารเพิ่มขึ้นกว่าข้อมูลสอน ผลที่ได้มีความถูกต้องเฉลี่ยร้อยละ 96.84

งานของสุขวสา [5] ทดลองการรู้จำตัวพิมพ์อักษรภาษาไทยจำนวน 6,528 ตัว มี 6 รูปแบบ ๆ ละ 8 ขนาด โดยใช้กลุ่มก้อนนิวรอลเน็ตเวิร์กเพื่อทำหน้าที่จำแนกประเภทของข้อมูล โดยการรวมผลลัพธ์จากตัวจำแนกประเภทหลาย ๆ ตัวเพื่อทำนายผลลัพธ์สุดท้ายและใช้การรวมผลลัพธ์แบบถ่วงน้ำหนัก ผลการทดลองพบว่า ข้อมูลทดสอบชุดแรก (มีความเข้มของเอกสารจางลงกว่าข้อมูลสอน) มีความผิดพลาดต่ำสุดร้อยละ 1.53 และข้อมูลทดสอบชุดที่สอง (มีความเข้มของเอกสารเพิ่มขึ้นกว่าข้อมูลสอน) มีความผิดพลาดต่ำสุดร้อยละ 1.29

งานของพัฒนชัย [7] ทดลองการรู้จำตัวพิมพ์อักษรภาษาไทยจำนวน 1,424 ตัว ซึ่งมีรูปแบบ 6 รูปแบบ ๆ ละ 8 ขนาด ผลที่ได้มีความถูกต้องเพิ่มขึ้นกว่าของโปรแกรมอ่านไทย 2.0 แต่ใช้หน่วยความจำและเวลามากกว่าเดิม

ส่วนการรู้จำตัวเลขอารบิก ดังเช่นในงานวิจัยของ Cun และคณะ [3] ได้ทดลองกับข้อมูลตัวเลขรหัสไปรษณีย์ ประกอบไปด้วยตัวพิมพ์และลายมือเขียน (ลายมือเขียน 2,007 ตัว และตัวพิมพ์ 2,007 ตัว) จากการทดลองพบว่ามีอัตราความผิดพลาด (Error Rate) 1% และการไม่รู้จำ (Reject) 9% งานวิจัยของ Grother [4] ทดลองกับลายมือเขียนตัวเลขอารบิกจำนวน 15,000

ตัวอย่าง พบว่ามีความถูกต้องร้อยละ 96.1 งานวิจัยของ Lee และ Kim [9] ทดลองกับลายมือเขียนตัวเลข โดยใช้นิวรอลเน็ตเวิร์กแบบวกกลับรูปแบบใหม่ ซึ่งแตกต่างจากจอร์แดนและเอลแมน เนื่องจากมีการพัฒนามาจากนิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า แต่ปรับโครงสร้างให้สอดคล้องกับนิวรอลเน็ตเวิร์กแบบวกกลับ คือ โหนดแต่ละโหนดของชั้นซ่อนจะถูกเชื่อมต่อ (Connected) จากทุก ๆ โหนดของชั้นอินพุต และโหนดแต่ละโหนดของชั้นเอาท์พุตถูกเชื่อมต่อจากทุก ๆ โหนดของชั้นซ่อน ตลอดจนเชื่อมต่อโดยตรงกับโหนดของตัวเองในชั้นเอาท์พุต และเชื่อมต่อกับทุก ๆ โหนดในชั้นเอาท์พุตด้วยกันเอง โดยทดลองกับ 4 เน็ตเวิร์ก คือ นิวรอลเน็ตเวิร์กแบบป้อนไปข้างหน้า จอร์แดน เอลแมนและนิวรอลเน็ตเวิร์กแบบวกกลับของ Lee และ Kim ผลการทดลองพบว่าเน็ตเวิร์กแบบที่สี่ให้ความถูกต้องสูงสุด