รายงานโครงการวิจัยประจำปีงบประมาณ 2545

เรื่อง

ศึกษาแนวทางการพัฒนาโคโฮเนนนิวรอลเน็ตเวิร์คให้สามารถรับข้อมูลและประมวลผลข้อมูลประเภทข้อความเพื่อใช้ในงานแบ่งกลุ่มเอกสาร

A TEXT PROCESSING KOHONEN NEURAL NETWORK FOR DOCUMENT CLUSTERINGS

โดย

ชื่อหัวหน้าโครงการวิจัย

ผศ.ดร. วรพจน์ กรีสุระเดช

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ 10520

พ.ศ. 2546

สารบัญ

หน้า

บทที่

1. วัตถุประสงค์

วัตถุประสงค์ของการทำ clustering (การจัดกลุ่ม) คือ การแยกข้อมูลออกเป็นกลุ่มๆ โดยข้อมูลที่อยู่ภายในกลุ่มเดียวกันจะมีค่าความเหมือนกันมากกว่าข้อมูลที่อยู่ต่างกลุ่มกัน ซึ่งเราสามารถนำผลลัพธ์จากการ clustering มาใช้ในลักษณะของการแบ่งกลุ่มของข้อมูลหรืออาจทำไปใช้ในลักษณะของการแยกย่อยข้อมูลเพื่อเลือกเอาข้อมูลเฉพาะกลุ่มที่เราสนใจเอาไปทำการวิเคราะห์ด้วยวิธีการอื่นๆต่อไป การทำเช่นนี้ทำให้สามารถลดปริมาณการใช้ทรัพยากรของระบบลงและยังทำให้ได้ผลลัพธ์ที่เฉพาะเจาะจงมากขึ้น

เทคนิคในการ clustering สำหรับวัตถุที่มีค่าของคุณสมบัติเป็น numerical values(ค่าเชิงตัวเลข) เป็นที่รู้จักกันอย่างแพร่หลาย เช่น Sequential Agglomerative Hierarchical Nonoverlaping (SAHN) [8] เป็นการ clustering ในแบบ Hierarchical clustering โดยจะสร้าง cluster จากการจัดให้กลุ่มที่ใกล้เคียงกันมารวมกันในแต่ละรอบของการทำงาน Partitioning around medoids (PAM) [9] เป็นโมเดลในแบบของ partition clustering ซึ่งมีแนวคิดจากการหาตัวแทนของข้อมูลจำนวน k ตัว (เรียกอีกอย่างว่า mediods) จากข้อมูลทั้งหมดโดยผลรวมของค่าความแตกต่างภายในกลุ่มเดียวกันจะน้อยที่สุด Fuzzy C-Medoids (FCMdd) และ Robust Fuzzy C-Medoids (RFCMdd) [7] เป็นโมเดลการ clustering หนึ่งในแบบของ fuzzy clustering ซึ่งมีความแตกต่างจากการ clustering ในแบบอื่นๆคือข้อมูลข้อมูลหนึ่งอาจเป็นสมาชิกของ cluster 2 cluster ในเวลาเดียวกันได้

แม้ว่าวิธีเหล่านี้สามารถทำการ clustering ข้อมูลที่มีคุณสมบัติเป็นค่าเชิงตัวเลขได้เป็นอย่างดี แต่ในปัจจุบัน ความสำคัญของข้อมูลที่เป็นประเภทข้อความเริ่มมีมากขึ้น ข้อมูลที่เป็นเอกสารข้อความมีจำนวนมากขึ้น แม้ว่าวิธีในการ clustering แบบเดิมๆจะสามารถใช้ในการ clustering เอกสารเหล่านี้ได้ แต่ก็ต้องมีการนำข้อมูลไปผ่านขั้นตอนการแปลงรูปข้อมูลคุณสมบัติของเอกสารเหล่านั้นให้เป็นค่าเชิงตัวเลขเสียก่อน จึงจะนำมาใช้ได้ วิธีการนี้แม้จะให้ผลลัพธ์ในระดับที่ยอมรับได้ แต่การแปลงข้อมูลที่เป็นข้อความให้เป็นค่าเชิงตัวเลขก่อนนั้น อาจทำให้ข้อมูลสูญเสียความหมายในตัวมันเองไป นอกจากนี้ยังทำให้เสียเวลาค่อนข้างมากในการแปลงและกำหนดสัญลักษณ์เพื่อแทนข้อมูลเหล่านั้นในกรณีที่ข้อมูลมีการกระจายตัวกันมากด้วย

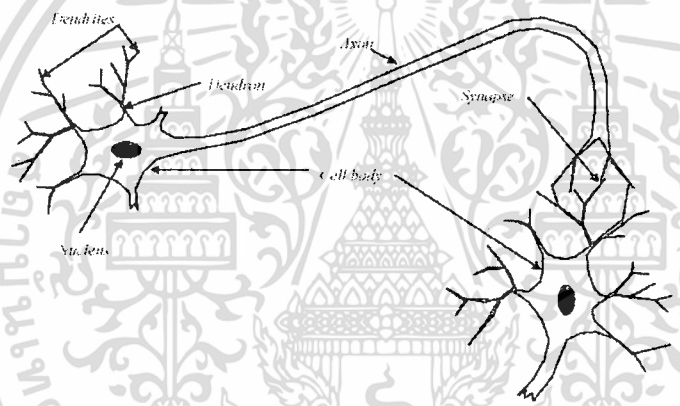
Text Processing Kohonen Neural Networks เป็นโครงข่ายประสาทเทียม (neural network) ที่ขยายความสามารถของ Self-Organizing Feature Maps เพื่อการทำ clustering ข้อมูลที่มีลักษณะเป็นข้อความได้โดยตรง ไม่ต้องผ่านกระบวนการในการแปลงรูปข้อมูล โดยประยุกต์แนวคิดเรื่องการเปรียบเทียบความแตกต่างของ symbolic data [6] เข้าไปในส่วนของการทำ competition และ Synaptic Adaptation ของโครงข่ายประสาทเทียมแบบ Self-Organizing Feature Maps.

โครงข่ายประสาทเทียมแบบใหม่สามารถรับข้อมูลที่เป็นข้อความได้โดยตรงและทำการ clustering ข้อมูลที่คุณสมบัติมีค่าเป็นข้อความได้เป็นอย่างดี

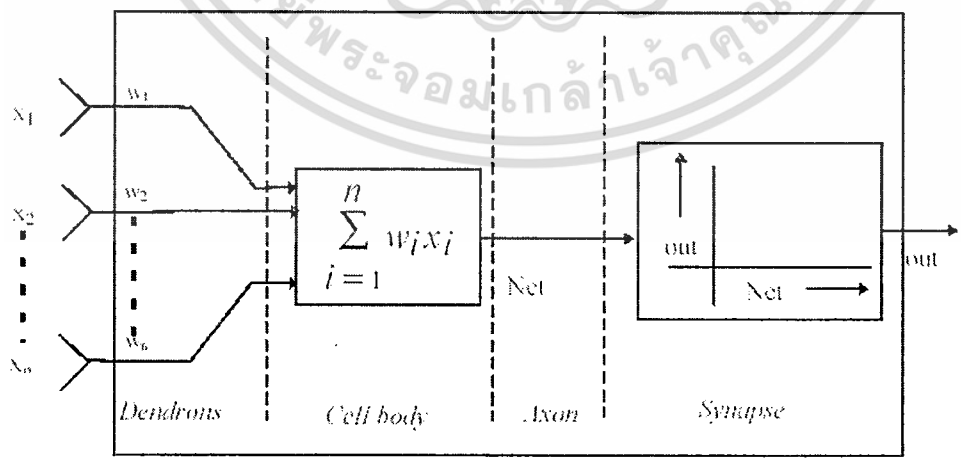
2. ทฤษฎี

2.1 โครงข่ายประสาทเทียม (Neural Network)

โดยทั่วไปแล้ว โครงสร้างตามธรรมชาติของ nervous system (ระบบเส้นประสาท)ของมนุษย์ก็คือโครงข่ายประสาทที่ซับซ้อนมากๆนั่นเอง โดยมีสมองเป็นจุดศูนย์กลางการทำงาน nervous system ของมนุษย์ ประกอบด้วยเซลล์ประสาทจำนวน 10^{10} เซลล์ประสาท ซึ่งจะเชื่อมต่อกับเซลล์ประสาทอื่นๆทางโครงข่ายย่อยๆ แต่ละเซลล์ประสาทในสมองจะประกอบด้วยส่วน body , Axon(แกนของเซลล์ประสาทที่นำส่งกระแสประสาท), และ dendrites(ส่วนของเซลล์ประสาทที่มีลักษณะคล้ายกิ่ง) จำนวนมาก ดังแสดงในรูปที่ 1



รูปที่ 1(a) แสดงลักษณะของเซลล์ประสาท



รูปที่ 1(b) แสดงลักษณะของเซลล์ประสาทเทียม

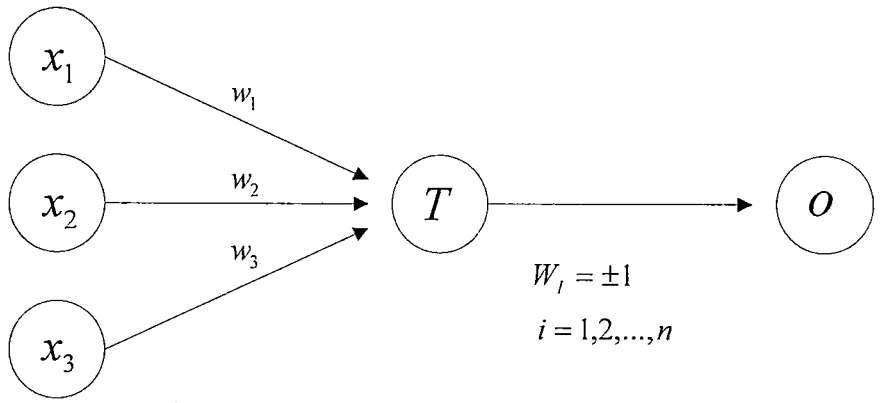
การทำงานของเซลล์ประสาทประกอบด้วย Axon ซึ่งนำส่งกระแสสัญญาณเอาต์พุต โดยเอาต์พุตนี้จะส่งสัญญาณจาก neural หนึ่งไปยังอีก neural หนึ่ง ผ่าน Axon โดยที่จะมีเซลล์ประสาทรับสัญญาณเรียกว่า dendrites เป็นเซลล์ที่ใช้รับสัญญาณเข้ามาให้กับ neural โดย neural เป็นจุดกลางที่ใช้ประมวลผลเมื่อประมวลผลเสร็จ ก็ส่งเอาต์พุตออกไปให้กับ dendrites ของอีกเซลล์หนึ่ง ช่วงต่อระหว่าง Axon กับ dendrites จะมีช่องว่างเล็กๆเรียกว่า synapse(ส่วนที่มาบรรจบกันของเซลล์ประสาท) โดย synapse จะทำหน้าที่หลั่งสารเคมีออกมาเมื่อต้องการส่งสัญญาณ สารนี้ก็จะเข้าไปสู่ dendrites ของอีก neural หนึ่ง

โครงข่ายประสาทเทียมเป็นวิธีหนึ่งที่ช่วยให้คอมพิวเตอร์มีความสามารถมากขึ้น โดนเฉพาะการประมวลผลข่าวสารที่มีความยุ่งยากซับซ้อน เช่นการทำให้คอมพิวเตอร์สามารถรู้จำตัวอักษร ซึ่งต้องมีการตัดสินใจที่ต้องอาศัยความรู้และประสบการณ์ โดยถ้าใช้เทคนิคทางคณิตศาสตร์ธรรมดาในการแก้ปัญหาก็จะมีความซับซ้อนมาก แต่ถ้าใช้ระบบโครงข่ายประสาทเทียมก็จะช่วยลดความยุ่งยากลงได้มาก ซึ่งระบบที่สร้างขึ้นมานี้ถูกเรียกว่าระบบแบบจำลองโครงข่ายประสาทเทียม (Artificial Neural Network System: ANNS) ที่ทำให้คอมพิวเตอร์มีความสามารถในการเรียนรู้และตัดสินใจให้ระบบได้ โดยจะถอดแบบมาจากการทำงานของระบบสมองของมนุษย์

แบบจำลองโครงข่ายประสาทเทียมที่ใช้ในการประมวลผลโดยเครื่องคอมพิวเตอร์เพื่อนำไปใช้ควบคุมรักษาสมดุลต่างๆนั้น ระบบแรกถูกนำเสนอโดย McCulloch และ Pitt ในปีค.ศ. 1943 ซึ่งเป็นแบบจำลองของเซลล์ประสาทดังรูปที่ 2 อินพุต x_i (สำหรับ i = 1, 2……, n) จะมีค่าเป็น {0, 1} ซึ่งจะขึ้นอยู่กับสัญญาณอินพุตจากเซลล์อื่นในขณะนั้นว่าจะมีหรือไม่มีสัญญาณ ส่วนสัญญาณที่จะส่งต่อไปยังเซลล์ถัดไปซึ่งเป็นเซลล์ผลลัพธ์ (จะแทนด้วย o) และ Firing Level ของแบบจำลองนี้ถูกกำหนดโดย

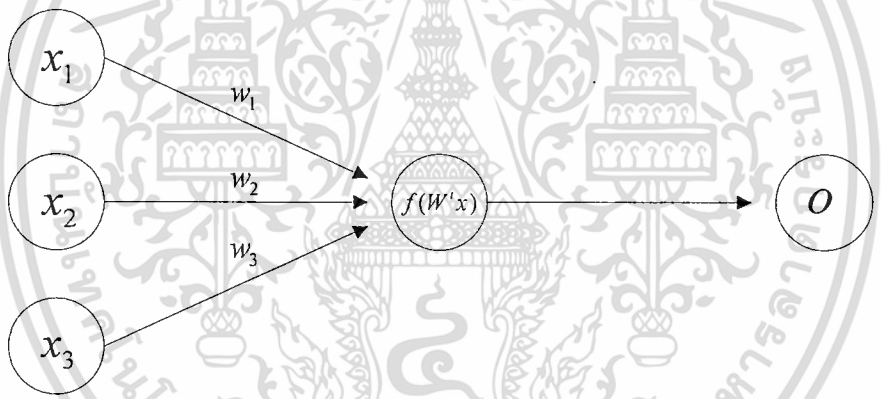
$$o^{k+1} = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & Otherwise \end{cases}$$

โดยที่ k =0,1,2,… เป็นช่วงเวลาแบบไม่ต่อเนื่อง w_i เป็นค่าถ่วงน้ำหนัก ที่เชื่อมต่อกับอินพุตที่ I ซึ่งถ้า $w_i = +1$ แสดงถึงการกระตุ้นของ synapse และถ้า $w_i = -1$ synapse จะยับยั้งการส่งผ่านสัญญาณ และ T เป็นค่าความต่างศักดิ์เทรสโฮลด์หรือขีดเริ่มเปลี่ยนซึ่งถ้าค่าผลรวมของผลคูณระหว่างค่าถ่วงน้ำหนักกับสัญญาณอินพุตจะต้องมากกว่า T จึงจะมีสัญญาณผ่านไปยังเซลล์อื่นได้



รูปที่ 2 แบบจำลองเซลล์ประสาทของ McCulloch-Pitts

โครงข่ายอีกแบบหนึ่งซึ่งคล้ายกับแบบจำลองของ McCulloch-Pitts แต่แตกต่างกันตรงที่ค่าของตัวแปรต่างๆที่ใช้ในแบบจำลองโครงข่ายประสาทเทียม เป็นเลขจำนวนจริงและมีค่าถ่วงน้ำหนักจะได้จากการเรียนรู้ของระบบ ซึ่งแบบจำลองนี้แสดงในรูปที่ 3



รูปที่ 3แบบจำลองเซลล์ประสาทเทียม

จากรูปที่ 3 แสดงโครงข่ายการเชื่อมต่อของแบบจำลองเซลล์ประสาทที่สามารถสอนให้โครงข่ายตัดสินใจได้ โดยมี x_i เป็นสัญญาณอินพุต และ w เป็นค่าถ่วงน้ำหนักที่ได้จากการสอนโครงข่าย และแต่ละโหนดในโครงข่ายจะใช้แทนเซลล์ประสาทแต่ละเซลล์ ซึ่งบางครั้งจะเรียกว่าหน่วยประมวลผลพื้นฐาน (Process Element Unit) และมี synapse ซึ่งจะเชื่อมต่อโนดเพื่อใช้ในการส่งสัญญาณการกระตุ้นหรือยับยั้งสัญญาณจะขึ้นอยู่กับค่าถ่วงน้ำหนัก w_i และสำหรับสัญญาณเอาต์พุตสามารถคำนวณได้ดังนี้

$$o = f(W'x)$$

โดยที่ W เป็นเวคเตอร์ของค่าถ่วงน้ำหนักซึ่งสามารถกำหนดได้ดังนี้

$$W = [w_1, w_2, ..., w_n]'$$

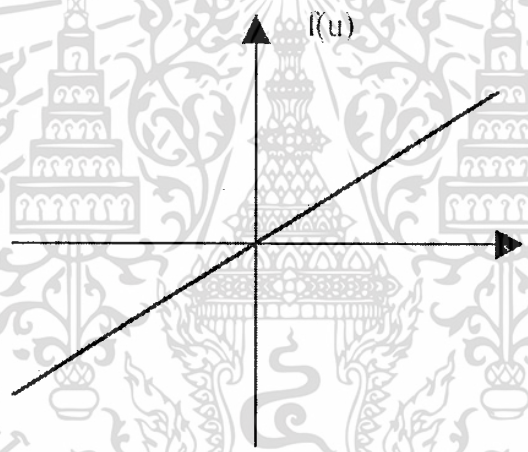
และ X เป็นเวกเตอร์อินพุต $X = [x_1, x_2, ..., x_n]'$ เมื่อ t เป็นตัวดำเนินการทรานสโพสต์ (Transpose)ของเมตริกซ์ ฟังก์ชันกำหนดสัญญาณเอาต์พุตในสมการที่ต่อไปนี้ถูกเรียกว่าฟังก์ชันการเร่งเร้าหรือแอกติเวชันฟังก์ชัน (Activation Function) และกำหนดให้

$$net = W'X = \sum_{i=1}^{n} w_i x_i$$

ฟังก์ชันการเร่งเร้ามีคุณสมบัติคล้ายกับกราฟของศักย์ไฟฟ้าขณะทำงาน มีด้วยกันสองชนิดคือ ชนิดที่เป็นเชิงเส้นและชนิดที่ไม่เป็นเชิงเส้น การเลือกใช้ฟังก์ชันการเร่งเร้าใดใน ANNS จะต้องพิจารณาให้เหมาะสมกับปัญหานั้นๆ แอกติเวชันฟังก์ชันที่นิยมใช้กันประกอบด้วย

1. Linear Function

Linear function เป็นฟังก์ชันหนึ่งที่นิยมใช้กันมาก กราฟของฟังก์ชันสามารถแสดงดังรูปที่ 4



รูปที่ 4 Linear Activation Function

สมการทางคณิตศาสตร์สำหรับฟังก์ชันนี้สามารถเขียนได้ดังนี้

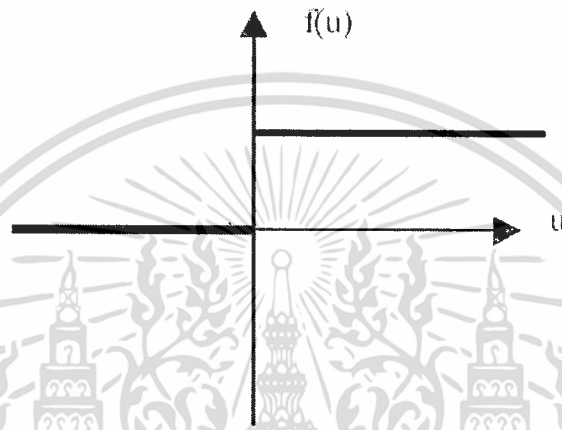
$$y = f(u) = \alpha.u$$

โดย α เป็นค่าความชัน (slope) ของ linear function ถ้าค่าความชัน α เป็น 1 แล้ว linear activation function นี้จะถูกเรียกว่า identify function โดย output (y) ของ identify function จะเท่ากับ input function (u) ถึงแม้ว่าฟังก์ชันนี้อาจดูเป็นกรณีที่ไม่มีสาระนัก แต่กระนั้นมันก็มีประโยชน์อย่างยิ่งในบางกรณีเช่น ในขั้นตอนสุดท้ายของ multilayer neural network

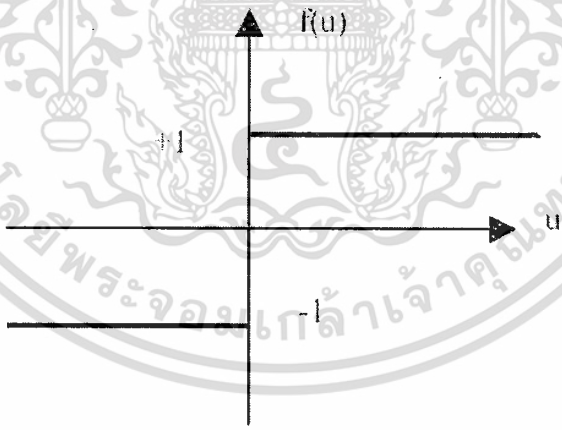
2. Threshold Function

Threshold (hard-limiter) activation function เป็นฟังก์ชันที่ให้ผลลัพธ์เป็นแบบ binary (ระบบเลขฐาน 2 คือ 0 และ 1) หรือ bipolar (มี 2 ขั้ว) ดังแสดงในรูปที่ 5 และ 6 ตามลำดับ ผลลัพธ์ของแบบ binary threshold function สามารถเขียนได้ดังนี้

$$y = f(u) = \begin{cases} 0 & if \quad u < 0 \\ 1 & if \quad u \geq 0 \end{cases}$$



รูปที่ 5 Binary Threshold Activation Function



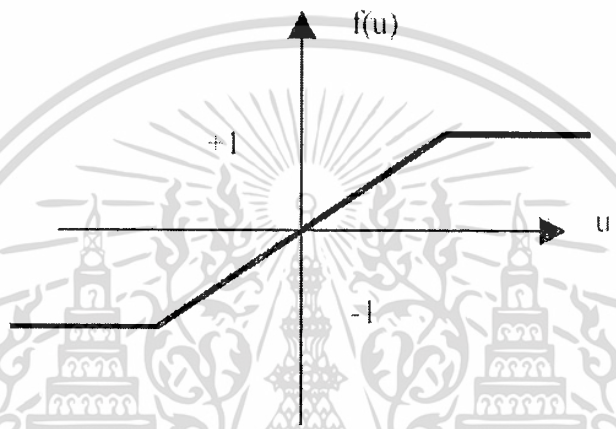
รูปที่ 6 Bipolar Threshold Activation Function

โครงข่ายประสาทเทียมที่ใช้ฟังก์ชันการเร่งเร้าแบบนี้ได้แก่ แบบจำลองของ McCulloch-Pitts

3. Piecewise Linear Function

ฟังก์ชันการเร่งเร้าแบบนี้ถูกเรียกอีกอย่างว่า saturating linear function และสามารถให้ค่าได้ทั้งแบบ binary หรือ bipolar อย่างใดอย่างหนึ่ง สมการทางคณิตศาสตร์ของ symmetric saturation function (รูปที่ 7) แสดงได้ดังนี้

$$y = f(u) = \begin{cases} -1 & if \quad u < -1 \\ u & if \quad -1 \le u \le 1 \\ 1 & if \quad u \ge 1 \end{cases}$$



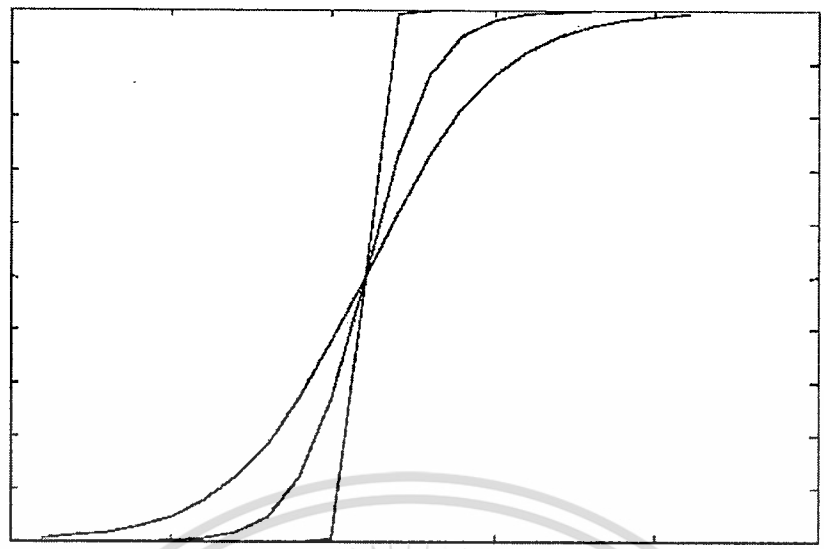
รูปที่ 7 Piecewise Linear Activation Function

4. Sigmoidal (S Shaped) Function

ฟังก์ชันแบบไม่เชิงเส้นนี้เป็นฟังก์ชันการเร่งเร้าที่ใช้โดยทั่วไปในการสร้างโครงข่ายประสาทเทียม มันมีลักษณะที่ดีในเชิงคณิตศาสตร์ สามารถคำนวณความเปลี่ยนแปลงได้และเป็นฟังก์ชันเพิ่ม(increasing function)ที่ดี sigmoidal transfer function(ฟังก์ชันการแปลงแบบ Sigmoid) สามารถเขียนในรูปสมการเชิงคณิตศาสตร์ได้ดังนี้

$$f(x) = \frac{1}{1 + e^{-\alpha x}}, \quad 0 \le f(x) \le 1$$

โดย α เป็น shape parameter ของ sigmoid function โดยการกำหนดค่าต่างๆของ shape parameter เราจะได้รูปกราฟดังแสดงในรูปที่ 8

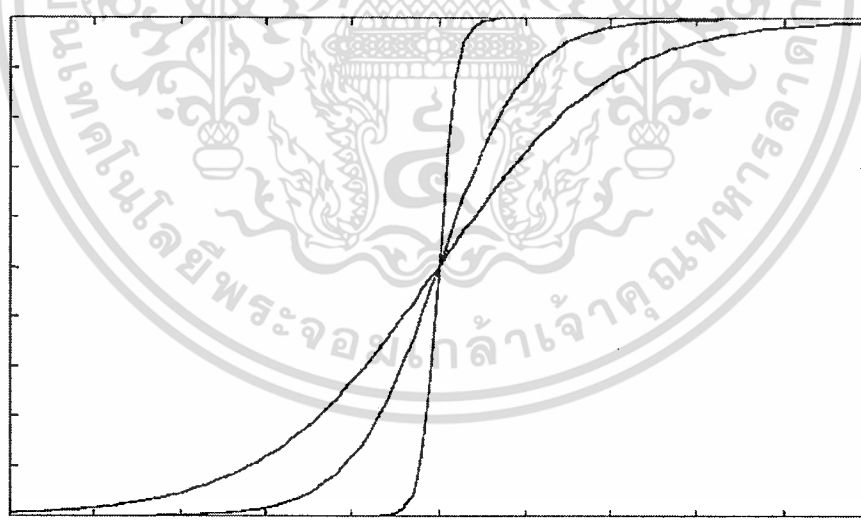


รูปที่ 8 A Sigmoid Activation Function

5. Tangent hyperbolic Function

ฟังก์ชันการแปลง(transfer function)แบบ Tangent hyperbolic Function (รูปที่ 9) นี้สามารถเขียนในรูปสมการทางคณิตศาสตร์ได้ดังนี้

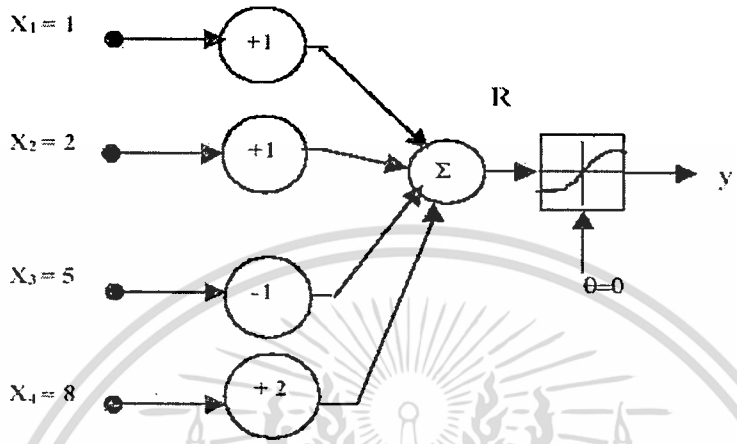
$$f(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}} \qquad -1 \le f(x) \le 1$$



รูปที่ 9 Tangent Hyperbolic Activation Function

ตัวอย่างการทำงานของ activation function

โครงข่ายประสาทเทียมนี้ประกอบด้วยโหนดข้อมูลเข้าจำนวน 4 โหนดและมีค่าถ่วงน้ำหนัก (weight) ดังรูป ที่ 10



รูปที่ 10 Neuron Structure

ผลลัพธ์ R ของโครงข่ายก่อนที่จะผ่านขั้นตอนของ activation function สามารถคำนวณได้ดังนี้

$$R = W^T.X = [1 \quad 1 \quad -1 \quad 2].\begin{bmatrix}1\\2\\5\\8\end{bmatrix} = 14$$

ด้วย binary activation function และ sigmoid function ผลลัพธ์ของโครงข่ายประสาทเทียมนี้จะเป็นไปตามลำดับดังนี้

$$y(Threshold) = 1$$

$$y(Sigmoid) = 1.5 * 2^{-8}$$

2.2 การเรียนรู้ของโครงข่ายประสาทเทียม

การเรียนรู้ของโครงข่ายประสาทเทียมจะมีประสิทธิภาพเพียงใดขึ้นอยู่กับค่าถ่วงน้ำหนักของโครงข่าย ซึ่งการฝึกสอน (training) โครงข่าย ก็คือการหาค่าถ่วงน้ำหนักที่เหมาะสมให้แก่โครงข่ายนั้นๆ วิธีการสอนโครงข่ายประสาทเทียม มีดังนี้

2.2.1 การเรียนรู้แบบชี้นำ (Supervised Learning) การสอน โดยวิธีนี้จะกำหนดเซตของการสอนให้กับโครงข่าย ซึ่งเซตนี้ประกอบด้วยอินพุตและเอาต์พุตที่ต้องการ เมื่อป้อนอินพุตให้กับโครงข่าย

โครงข่ายจะมีการประมวลผลจนได้คำตอบและค่าถ่วงน้ำหนักออกมาชุดหนึ่ง สำหรับคำตอบที่ได้จากโครงข่ายจะถูกนำมาคำนวณค่าความผิดพลาดโดยวัดเป็นระยะทางว่ามีความห่างจากคำตอบที่ต้องการของอินพุตในชุดเดียวกันมากน้อยเพียงใด ถ้ายังมีความผิดพลาดสูงอยู่ก็จะมีการปรับค่าถ่วงน้ำหนัก และทำการสอนต่อไปจนกว่าค่าความผิดพลาดระหว่างคำตอบโครงข่ายกับเอาต์พุตที่ต้องการมีค่าน้อยพอที่จะยอมรับได้จึงจะหยุดการสอน และค่าถ่วงน้ำหนักที่ได้จะเป็นเหมือนฟังก์ชันที่ใช้ในการแปลงข้อมูล

2.2.2 การเรียนรู้แบบไม่มีการชี้นำ (Unsupervised Learning) การสอนโดยวิธีนี้จะป้อนอินพุตเข้าสู่โครงข่ายและภายในโครงข่ายจะมีเอาต์พุตโนดอยู่หลายโนดด้วยกัน โดยแต่ละโนดแทนกลุ่มของข้อมูลที่มีคุณสมบัติเหมือนกัน เมื่อป้อนอินพุตเข้าสู่โครงข่าย โครงข่ายจะคำนวณค่าความสัมพันธ์ที่มีอยู่ภายในเซตของอินพุตโดยอาศัยค่าถ่วงน้ำหนักเป็นตัวแยกความแตกต่างของอินพุตไปเก็บไว้ในโนดเอาต์พุตของโครงข่าย การสอนโดยวิธีนี้ จะไม่สามารถระบุได้ว่าเอาต์พุตโนดใดเป็นของข้อมูลกลุ่มไหน ผู้ใช้จะต้องกำหนดเอง ซึ่งแตกต่างจากการสอนแบบชี้นำที่โครงข่ายสามารถระบุกลุ่มเอาต์พุตได้อย่างแน่นอน

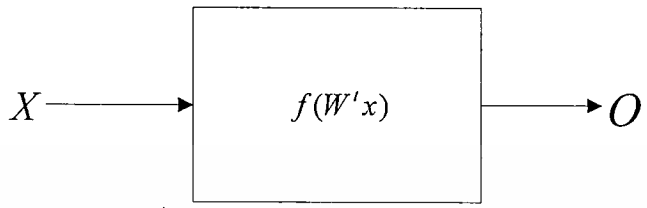
การสอนโครงข่ายเซลล์ประสาทเป็นการหาฟังก์ชันการแปลงและฟังก์ชันการแปลงที่ได้จะมีคุณสมบัติไม่เป็นเชิงเส้น ซึ่งฟังก์ชันการแปลงของโครงข่ายประสาทเทียมในที่นี้คือเซตของค่าถ่วงน้ำหนักของโครงข่าย ดังนั้นฟังก์ชันการแปลงจะมีศักยภาพมากน้อยเพียงใดนั้นจะขึ้นอยู่กับค่าถ่วงน้ำหนักของโครงข่ายนั้นๆว่ามีเสถียรภาพมากน้อยเพียงใดและค่าถ่วงน้ำหนักคำนวณได้จากการสอนโครงข่าย ซึ่งการสอนโครงข่ายมีหลายแบบด้วยกัน เช่น กฎการสอนของ Hebb(Hebbian Learning) กฎการสอนแบบเดลต้า(Delta Rule or Error-Correction Learning) กฎการเรียนรู้แบบแข่งขัน (Competitive Learning) เป็นต้น

2.3 แบบจำลองการทำงานของโครงข่ายประสาทเทียม

โครงข่ายประสาทเทียมและโครงข่ายของเซลล์ประสาทจริงของมนุษย์ จะมีการเชื่อมต่อกันของโนดในลักษณะของโครงข่ายอย่างแน่นหนา เพื่อให้โครงข่ายสามารถเรียนรู้และสามารถจดจำสิ่งที่ได้เรียนรู้มาแล้วได้ ซึ่งโครงสร้างการทำงานของโครงข่ายจะมีดังนี้

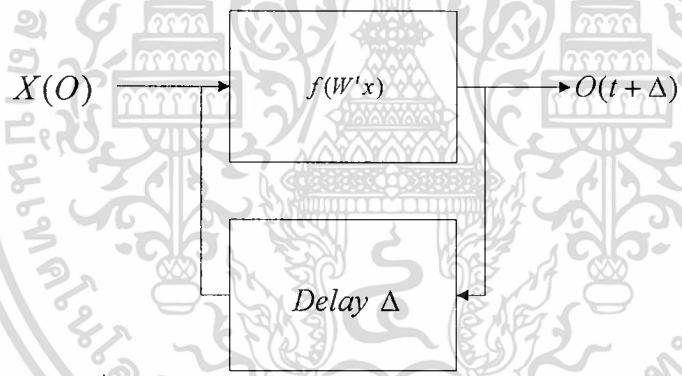
2.3.1 โครงข่ายที่ส่งสัญญาณไปข้างหน้า (Feedforward Networks) โครงข่ายชนิดนี้จะประกอบด้วยชั้นต่างๆของโครงข่าย โดยชั้นแรกจะเป็นอินพุตและชั้นสุดท้ายจะเป็นชั้นของเอาต์พุตส่วนระหว่างชั้นอินพุตและเอาต์พุตอาจจะมีหรือไม่มีชั้นที่แทรกอยู่ภายในก็ได้ ซึ่งจะขึ้นอยู่กับอัลกอริธึมที่ใช้ในการสอนโครงข่าย เช่นถ้าเป็นโครงข่าย Perceptron แบบหลายชั้น (Multilayer Perceptron) ก็จะมีชั้นที่อยู่ระหว่างระหว่างชั้นอินพุตและเอาต์พุตอีก ซึ่งอาจมีมากกว่าหนึ่งชั้นก็ได้ ส่วนโครงข่าย Self-

Organizing Map ของ kohonen จะมีเพียงชั้นของอินพุตและเอาต์พุตเท่านั้น การเชื่อมต่อระหว่างโครงข่ายแบบ Feedforward จะมีค่าถ่วงน้ำหนักเป็นตัวเชื่อมและสัญญาณอินพุตที่เข้ามาจะถูกส่งไปตามทิศทางของลูกศรจนถึงชั้นเอาต์พุตโดยไม่มีการป้อนกลับ ดังรูปที่ 11



รูปที่ 11 บล็อกไดอะแกรมของโครงข่าย Feedforward

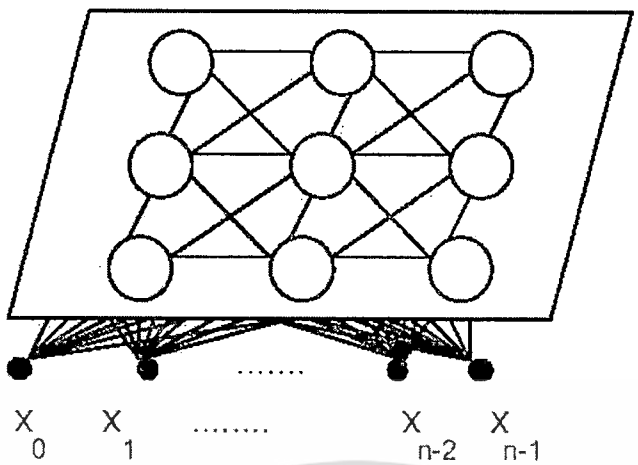
2.3.2 โครงข่ายที่มีการป้อนกลับ (Feedback Networks) ในส่วนแรกของโครงข่ายนี้จะเป็นโครงข่าย Feedforward เหมือนกับแบบแรก และส่วนที่เพิ่มเข้ามาคือส่วนของการป้อนกลับซึ่งมีการหน่วงเวลาไปจากเวลาเดิมเท่ากับ Δ ดังรูปที่ 12



รูปที่ 12 บล็อกไดอะแกรมของโครงข่าย Feedback

2.4 Self-Organizing Maps

Self-Organizing Maps(รูปที่ 13) ถูกนำเสนอโดย Tuevo Kohonen ในปี 1982 เป้าหมายของ algorithm นี้คือการแปลงข้อมูลที่เข้ามาให้อยู่ในรูปของตารางหนึ่งหรือสองมิติที่กำหนดไว้ โดยไม่ต้องอาศัยการสอนหรือชี้นำใดๆ



รูปที่ 13 โครงสร้างของ Self-Organizing Maps

ขั้นตอนการทำงานของ algorithm สามารถแบ่งได้เป็น 3 ส่วนสำคัญคือ Competition, Cooperation และ Synaptic Adaptation

1. Competition เป็นส่วนที่ทำการเปรียบเทียบความเหมือนกันระหว่างข้อมูลเข้ากับโนดต่างๆใน neural network โดยใช้กฎการเรียนรู้แบบแข่งขัน ซึ่งโนดที่มีค่าความเหมือนกันมากที่สุดจะเป็นโนดที่ได้รับการคัดเลือกเพียงโนดเดียว เราเรียกโนดนี้ว่า "wining neural"
2. Cooperation เป็นขั้นตอนในการหาโนดที่จะทำการ update ไปพร้อมๆกับ wining neural โดยผ่าน neighborhood function
3. Synaptic Adaptation เป็นขั้นตอนในการ update wining neural และโนดอื่นๆ ตามที่คำนวณได้จาก neighborhood function โดยการ update นี้จะทำให้โนดเหล่านี้มีค่าใกล้เคียงกับข้อมูลเข้ามากขึ้น

ขั้นตอนการทำงานของ SOFMs เป็นดังนี้

1. เริ่มจากการสุ่มค่าเพื่อกำหนดเป็นค่าเริ่มต้นให้กับ weight vector $w_j(0)$. โดยมีข้อแม้ว่าค่าเริ่มต้นนี้จะต้องไม่ซ้ำกันสำหรับ J= 1,2,3....N โดย N เป็นจำนวนของ neural ใน lattice วิธีหนึ่งในการกำหนดค่าของ $w_j(0)$ คือการสุ่มค่า $w_j(0)$ จากข้อมูลเข้า
2. ทำการเลือกข้อมูลเข้า x ด้วยการสุ่ม
3. เปรียบเทียบข้อมูลเข้า x เพื่อหา winning neuron $i(x)$ โดยใช้ minimum-distance Euclidean criterion:

$$i(x) = \arg_j \min \|x(n) - w_j\|, \qquad j = 1,2,...,N$$

4. ปรับค่า synaptic weight vectors โดยใช้

$$w_j(n+1) = \begin{cases} w_j(n) + \eta(n)[x(n) - w_j(n)], & j \in \wedge_{i(x)}(n) \\ w_j(n) & otherwise \end{cases}$$

โดย $\eta(n)$ เป็นค่า learning-rate และ $\wedge_{i(x)}(n)$ เป็น neighborhood function ซึ่งใช้ในการหา neighborhood ของwining neuron $i(x)$ ทั้ง $\eta(n)$ และ $\wedge_{i(x)}(n)$ จะมีค่าเปลี่ยนแปลงไปในแต่ละรอบของการเรียนรู้ ทั้งนี้เพื่อผลลัพธ์ที่ดี

5. เริ่มทำตั้งแต่ข้อ 2 ใหม่จนกว่าจะไม่มีความเปลี่ยนแปลงที่มากเกินกว่าค่าที่ยอมรับได้ใน feature map เกิดขึ้น

2.5 การเปรียบเทียบความเหมือนกันของเอกสาร

เอกสารต่างๆประกอบด้วยคุณสมบัติที่สามารถใช้แทนตัวเอกสารนั้นๆ เช่น เอกสารฉบับหนึ่งอาจประกอบด้วยชื่อหัวข้อของเอกสาร ชื่อผู้แต่ง คำที่พบมากในเอกสารนั้นๆ เป็นต้น ซึ่งเราสามารถเขียนแทนตัวเอกสารหนึ่งเอกสารซึ่งมีคุณสมบัติ n คุณสมบัติได้ดังนี้

$Doc = D_1 * D_2 * D_3 * ..... * D_n$ n เป็นจำนวนคุณสมบัติของเอกสาร

ตัวอย่างเช่น

$Book = Title * Author * keyword$ Title feature คือคำที่ใช้อธิบายชื่อหัวข้อเรื่องของเอกสารหรือหนังสือ

Author feature คือคำที่บ่งถึงชื่อผู้แต่ง

Keyword feature คือคำที่พบบ่อยๆในเอกสารนั้นๆ

ความแตกต่างของเอกสาร 2 เอกสาร A และ B นิยามตามแนวคิดของ El-Sonbaty [6] ได้ดังนี้

$$D(A,B) = \sum_{k=1}^{d} D(A_k, B_k)$$ สำหรับเอกสารที่มีคุณสมบัติ d คุณสมบัติ

การเปรียบเทียบ $D(A_k, B_k)$ สามารถแบ่งเป็น 2 ส่วนย่อยคือ การเปรียบเทียบในเชิง span, $D_s(A,B)$ และ การเปรียบเทียบในเชิง content, $D_c(A,B)$ ซึ่งนิยามได้ดังนี้

การเปรียบเทียบในเชิง content

$$D_c(A_k,B_k) = \frac{\left| Length\ \ of\ \ A_k\ \ + Lenght\ \ of\ \ B_k - 2 * Lenght\ \ of\ \ \text{intersection}\ \ of\ \ A_k\ \ and\ \ B_k \right|}{Span\ \ Length\ \ of\ \ A_k\ \ and\ \ B_k}$$

การเปรียบเทียบในเชิง span

$$D_s(A_k,B_k) = \frac{\left| Length\ \ of\ \ A_k\ \ - Lenght\ \ of\ \ B_k \right|}{Span\ \ Length\ \ of\ \ A_k\ \ and\ \ B_k}$$

Length of A_k คือ จำนวนค่าต่างๆในคุณสมบัติ A_k

Span Length of A_k *and* B_k คือจำนวนของค่าต่างๆในที่เกิดในการ union ของคุณสมบัติ A_k และ B_k

Length of Intersection of A_k *and* B_k คือจำนวนของค่าต่างๆในที่เกิดในการ intersection ของคุณสมบัติ A_k และ B_k

โดยความแตกต่างรวมของคุณสมบัติ 2 คุณสมบัติ A_k และ B_k จะหาได้จาก

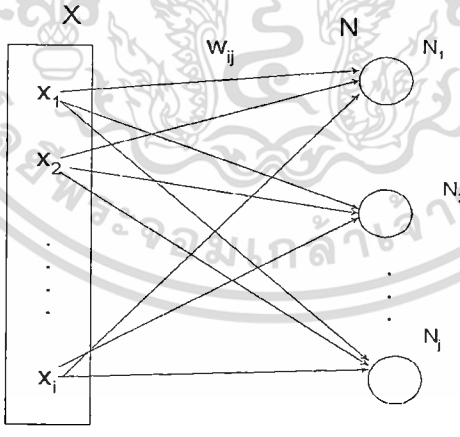
$$D(A_k, B_k) = D_s(A_k, B_k) + D_c(A_k, B_k)$$

ความแตกต่างของเอกสาร 2 เอกสาร A และ B คือ

$$D(A,B) = \sum_{k=1}^{d} D(A_k, B_k)$$ สำหรับเอกสารที่มีคุณสมบัติ d คุณสมบัติ

3. The Text Processing Kohonen Neural Networks

Text Processing Kohonen Neural Networks เป็น นิวรอลเน็ตเวิร์คที่ปรับปรุงมาจาก kohonen self-organizing map โดยนำแนวคิดเกี่ยวกับการเปรียบเทียบความแตกต่างระหว่าง symbolic object มาประยุกต์เพื่อให้นิวรอลเน็ตเวิร์คนี้สามารถรับข้อมูลที่เป็นข้อความได้โดยตรง ซึ่งทำให้สามารถลดขั้นตอนในการปรับเปลี่ยนข้อมูลให้เป็นข้อมูลเชิงปริมาณและยังคงสามารถรักษาความหมายของข้อมูลให้คงเดิมอยู่ได้ โครงสร้างของ นิวรอลเน็ตเวิร์คนี้แสดงได้ดังรูปที่ 1



รูปที่ 14 แสดงโครงสร้างการทำงานของ Text Processing Kohonen Neural Networks

นิวรอลเน็ตเวิร์คที่นำเสนอประกอบด้วยสองส่วนหลักคือส่วนของ input unit และส่วนของ output unit ข้อมูลเข้า X_i ใน input unit จะเชื่อมต่ออย่างสมบูรณ์กับโหนดทุกโหนดใน output unit N_j เส้นที่เชื่อมระหว่าง input unit X_i กับ output unit N_j คือ weight ซึ่งนิยามได้ดังนี้

$$w_{ij} = \{(A_{1ij}, e_{1ij}), (A_{2ij}, e_{2ij}), (A_{3ij}, e_{3ij})..., (A_{pij}, e_{pij}) \}$$

โดย A_{pij} คือค่า qualitative value ของ weight W_{ij}

e_{pij} คือ ค่าแสดงความสมาชิกของ A_{pij} e_{pij} มีค่าระหว่าง 0 ถึง 1 โดย $e_{pij} = 0$ ถ้า qualitative value A_{pij} ไม่ได้เป็นส่วนหนึ่งของข้อมูลเข้า i เลย ในขณะที่ $e_{pij} = 1$ ถ้า qualitative value A_{pij} เป็นสมาชิกของข้อมูลเข้า i อย่างสมบูรณ์

3.1 กระบวนการเรียนรู้ของอัลกอริธึม

Text Processing Kohonen Neural Network เป็นการขยายความสามารถของ Kohonen Self-Organizing Map โดยเพิ่มเติมแนวคิดเกี่ยวกับการเปรียบเทียบคุณสมบัติที่มีค่าของข้อมูลเป็นแบบ qualitative(ข้อมูลเชิงคุณภาพ) ซึ่งทำให้อัลกอริธึมนี้สามารถจัดการกับข้อมูลแบบ qualitative(ข้อมูลเชิงคุณภาพ) ได้โดยตรงโดยไม่ต้องผ่านกระบวนการการแปลงค่า qualitative(ข้อมูลเชิงคุณภาพ) ให้เป็นข้อมูลเชิงตัวเลขอีกต่อไป ขั้นตอนการทำงานของอัลกอริธึมเป็นดังนี้

1: เริ่มจากการกำหนดค่าเริ่มต้นให้กับ weight w_{ij} ในแต่ละโหนดของ neural W_i. ค่าของ weight w_{ij} อาจได้จากการสุ่มเลือกจากข้อมูลที่นำมาใช้ในกระบวนการเรียนรู้

$W_i = \{w_{i1}, w_{i2}, ..., w_{ij}\}$ w_{ij} = เป็นค่า Weight ของ neural W_i คุณสมบัติที่ j

$w_{ij} = \{(A_{1ij}, e_{1ij}), (A_{2ij}, e_{2ij}), (A_{3ij}, e_{3ij})..., (A_{pij}, e_{pij}) \}$ A_{pij} = ค่าของสมาชิกตัวที่ p^{th} ใน w_{ij}

e_{pij} = ค่าความเป็นสมาชิกของ A_{pij}

2: เมื่อยังไม่ตรงกับเงื่อนไขในการหยุด ให้ทำข้อ 3 ถึง 6 ต่อไป

3: สุ่มเลือกข้อมูลเข้า X จากข้อมูลเข้าทั้งหมดโดยสำหรับแต่ละข้อมูลเข้า X นิยามได้ดังนี้

$$X = (x_1, x_2, ..., x_d)^T$$

4: เปรียบเทียบข้อมูลเข้า X กับแต่ละโหนดของ neural W_i โดยจาก

$\|X - W_i\| = \sum_{k=1}^{d} D(x_k, w_{ik})$ d =เป็นจำนวนของคุณสมบัติของข้อมูลเข้า X และโหนดของ neural W_i

เพื่อหาโหนดของนิวรอลตัวที่ "I" ที่ให้ค่า $\|X - W_I\|$ ต่ำที่สุด ซึ่งก็คือมีค่าใกล้เคียงกับข้อมูลเข้า X มากที่สุดนั่นเอง.

5: สำหรับโหนดที่ได้รับการคัดเลือก "I" และโหนดที่เป็น neighborhood $(\wedge_I)$ของ "I" ทำการปรับปรุงค่าดังนี้

$$w_{ij}^{(new)} = \begin{cases} w_{ij}^{(old)} \bigcup x_j & if \quad i \in \wedge_I \\ w_{ij}^{old} & Otherwise \end{cases}$$

$$e_{ijp}^{(new)} = \begin{cases} f(e_{ijp}^{(old)} + \eta) & if \quad A_{ijp} \in w_{ij} \cap x_j \\ f(e_{ijp}^{(old)} - \eta) & if \quad A_{ijp} \notin w_{ij} \cap x_j \end{cases}$$

โดย $f(.)$ นิยามได้ดังนี้

$$f(x) = \begin{cases} x & if \quad 0 \le x \le 1 \\ 0 & if \quad x < 0 \\ 1 & if \quad x > 1 \end{cases}$$

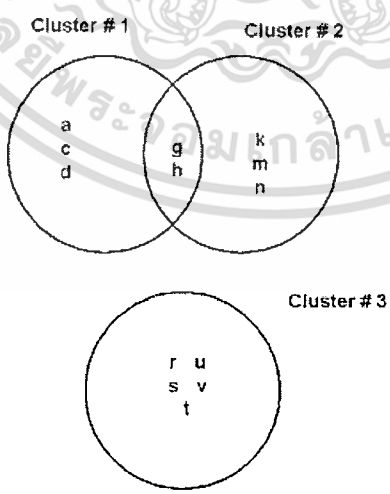
6: ทำตามขั้นตอนที่ 2-5 ต่อไปจนกว่าเงื่อนไขในการหยุดจะเกิดขึ้น.

3.2 ผลการทดลองเบื้องต้น

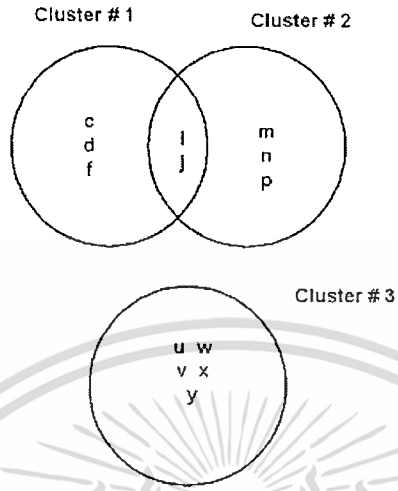
ในการทดลองเพื่อตรวจสอบความถูกต้องเบื้องต้นของอัลกอริธึม โดยใช้ชุดข้อมูลซึ่งสร้างขึ้นเพื่อใช้ในการเทรนนิ่งจำนวน 100 ชุด ชุดข้อมูลนี้สร้างจากกลุ่มของตัวอักษรจำนวน 3 กลุ่มซึ่งใช้เป็นตัวแทนของข้อมูลที่มีค่าของคุณสมบัติเป็นข้อความ ข้อมูลแต่ละตัวประกอบด้วยคุณสมบัติ 2 คุณสมบัติคือ Title และ keyword

$$Doc = Title * Keyword .$$

กลุ่มของตัวอักษรที่ใช้สร้างคุณสมบัติ Title และ keyword แสดงได้ดังรูปที่ 2 และ 3



รูปที่ 15 ชุดตัวอักษรที่ใช้สร้างข้อมูลในคุณสมบัติ Title



รูปที่ 16 ชุดตัวอักษรที่ใช้สร้างข้อมูลในคุณสมบัติ keyword

ตัวอย่างของข้อมูลที่สร้างขึ้นจากชุดของตัวอักษรในภาพที่ และ แสดงได้ดังตารางที่ 1

ตารางที่ 1. แสดงข้อมูลบางส่วนที่ใช้ในการเทรนนิ่ง

ลำดับที่	ค่าของคุณสมบัติ	
	Title	Keyword
1	c,d,g,h	d,j
2	N	i,m,np
3	r,s,t,v	u,w,y
4	h,k,n	j,p

โดยข้อมูลที่นำมาใช้ในการเทรนนิ่งประกอบด้วยข้อมูลที่เป็นสมาชิกของ cluster 1 จำนวน 34 ข้อมูล ข้อมูลที่เป็นสมาชิกของ cluster 2 จำนวน 35 ข้อมูล ข้อมูลที่เป็นสมาชิกของ cluster 3 จำนวน 35 ข้อมูล สำหรับนิวรอลเน็ตเวิร์คที่ถูกใช้ในการเทรนนิ่งนี้ประกอบด้วยนิวรอล (neural) จำนวน 4 โหนด แต่ละโหนดประกอบด้วย weight(w_{ij}) ซึ่งเป็นตัวแทนของ feature 2 feature คือ title feature และ keyword feature ผลลัพธ์ที่ได้หลังจากการทำเทรนนิ่งแล้วเป็นดังตารางที่ 2

ตารางที่ 2 ผลลัพธ์ที่ได้จากการเทรนนิ่ง

Neural ลำดับที่	ผลลัพธ์จากการเทรนนิ่ง		แสดงความเป็นตัวแทนของ cluster
	Title	Keyword	
1	c,d,g,h,s,u,v	d,f,l,j,w,y	1
2	d,g,h,k,m,n,s,t,v	c,d,f,i,j,p,w,y	Other
3	A,g,h,k,n	D,f,l,j,m,n,p	2
4	S,t,u,v	U,v,w,x,y	3

หลังจากทำการเทรนนิ่งแล้ว เราจะได้ค่า weight(w_{ij}) ของนิวรอลซึ่งเป็นตัวแทนของ title feature และ keyword feature ของข้อมูลที่นำมาทำการเทรนนิ่ง โดยนิวรอลลำดับที่ 1 เป็นตัวแทนของ cluster 1 นิวรอลลำดับที่ 3 เป็นตัวแทนของ cluster 2 และนิวรอลลำดับที่ 4 เป็นตัวแทนของ cluster 3

ในการทดสอบความถูกต้องของนิวรอลเน็ตเวิร์คที่ได้จากการเทรนนิ่ง เราใช้ข้อมูลที่สร้างขึ้นจากชุดตัวอักษรเดียวกันกับข้อมูลที่ใช้ในการเทรนนิ่งจำนวน 1000 ตัวนำมาใช้เพื่อตรวจสอบความถูกต้องของโมเดลนิวรอลเน็ตเวิร์คที่ได้ โดยข้อมูลที่นำมาตรวจสอบนั้นประกอบด้วยข้อมูลที่เป็นสมาชิกของ cluster 1 จำนวน 298 ข้อมูล ข้อมูลที่เป็นสมาชิกของ cluster 2 จำนวน 277 ข้อมูลและข้อมูลที่เป็นสมาชิกของ cluster 3 จำนวน 339 ข้อมูล โดยผลลัพธ์จากการทดสอบเป็นดังตารางที่ 3

ตารางที่ 3 แสดงผลลัพธ์ของการทดสอบโมเดลนิลรอลเน็ตเวิร์คของข้อมูลทดสอบจำนวน 1000 ตัว

Cluster ลำดับที่	จำนวนสมาชิก	ผลลัพธ์ที่ได้จากการทำ clustering	จำนวนความผิดพลาดที่เกิดขึ้น
1	338	298	40
2	323	277	46
3	339	339	0
ค่าความถูกต้องของการแบ่งกลุ่ม (r)		91.4 %	

เพื่อวัดค่าความถูกต้องของโมเดลที่เสนอ ค่าความถูกต้องของการแบ่งกลุ่ม r นิยามดังนี้

$$r = \left[1 - \left(\frac{\sum_{i=1}^{c} doc_i}{n}\right)\right] * 100\%$$

doc เป็นจำนวนของเอกสารซึ่งเป็นความผิดพลาดในการ clustering

c คือจำนวนของ cluster ที่แท้จริงของข้อมูล

จากข้อมูลที่นำมาทดสอบ นิลรอลเน็ตเวิร์คที่สร้างขึ้นมีค่าความถูกต้องของการแบ่งกลุ่มเท่ากับ 91.4 %. ซึ่งแสดงให้เห็นว่านิลรอลเน็ตเวิร์คที่สร้างขึ้นสามารถทำงานได้เป็นอย่างดีในการแบ่งกลุ่มของข้อมูลที่เป็นข้อความ

4. เอกสารอ้างอิง

[1] Anil K. Jain, Richard C. Dubes. Algorithms for Clustering Data. Prentice Hall Englewood Cliffs, New Jersey 07632. ISBN: 0-13-022278-X

[2] Eric Backer. Computer-assisted Reasoning in Cluster Analysis. Prentice Hall International (UK) Limited Campus 400, Maylands Avenue Hemel Hempstead Hertfordshire, HP2 7EZ ISBN: 0-13-341884-7

[3] Merkl D., "Text Data Mining," A Handbook of Natural Language Processing:Techniques and Applications for The Processing of Language as Text, Edited by Dale R., Moisl H., and H., Mercel Dekker, New York, 1998.

[4] Kohonen T. " Self-Organization and Associative Memory" Berlin: Springer-Verlag 1989.

[5] Gowda C.K. and Diday E., "Symbolic Clustering Using a New Similarity Measure", IEEE Trans. On Syst., Man, Cybern., vol. 22, no. 2, pp.368-378, 1992.

[6] El-Sonbaty Y.A. and Ismail M.A., "Fuzzy Clustering for symbolic Data", IEEE Trans. On Fuzzy Systems, vol.6, no.2, pp.195-204, 1998.

[7] Raghu Krishnapuram, Anupam Joshi , Olfa Nasraoui , "Low Complexity Fuzzy Relational Clustring Algorithms for Web Mining" , IEEE Trans. On Fuzzy Systems,vol.9, no.4,pp.595-607,2001.

[8] P.H.A. Sneath and R.R. Sokal, Numerical Taxonomy: The Principles and Practice of Numerical Classification. Sna Francisco, CA: W.H. Freeman, 1973.

[9] J.Kaufman and P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Brussels, Belgium: Wiley, 1990.

[10] Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. "WEBSOM--self-organizing maps of document collections" Neurocomputing, volume 21, pp 101-117.