การใช้ Hopfield Network ในการหาทางเลือกที่เหมาะสมที่สุด สำหรับ Optimization Problems

Using Hopfield Network for Solving Optimization Problems

โดย

นายอภิชาต อนุมัติ

รหัส 43067115

อาจารย์ที่ปรึกษา

ดร. อาริต ธรรมโน

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2544

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การใช้ Hopfield Network ในการหาทางเลือกที่เหมาะสมที่สุด สำหรับ Optimization Problems
นักศึกษา	นายอภิชาต อนุมัติ
อาจารย์ที่ปรึกษา	ดร. อาริต ธรรมโน
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

การทำงานที่ให้คอมพิวเตอร์มาช่วยแก้ปัญหาที่มีความยุ่งยากซับซ้อนมากๆ นั่นก็หมายความว่า เราจะต้องใช้เวลาในการคำนวนและต้องการใช้ทรัพยากรของระบบคอมพิวเตอร์ส่วนหนึ่ง ซึ่งจะใช้เวลามากขึ้นถ้าปัญหานั้นมีความซับซ้อนมาก ในที่นี้จึงเสนอวิธีการใช้ Hopfield Network ในการหาทางเลือกที่เหมาะสมที่สุด สำหรับปัญหา Optimization Problems ที่มีทางเลือกหลายๆทาง ได้ในเวลาที่สั้นกว่าเดิม ในที่นี้ได้ใช้ปัญหา Traveling Salesperson Problem เป็นตัวอย่าง นอกจากนี้ยังได้เสนอการปรับเปลี่ยนค่า activation function หรือ output function เพื่อเพิ่มประสิทธิภาพในการใช้ Hopfield Network กับปัญหา Optimization Problems ในการทดลองเบื้องต้นใช้จำนวนเมืองสำหรับ TSP เพียง 5 เมืองเท่านั้น โดยทำการสุ่มการกระจายของเมือง 60 ชุด ให้แต่ละชุดจะทำการทดลองชุดละ 10 ครั้งโดยแต่ละครั้งจะเพิ่มสิ่งรบกวน noise เข้าไปเป็น Input ต่างกัน จากผลการทดลองพบว่าการปรับเปลี่ยนค่า output function สามารถลดค่าความผิดพลาดจากเดิมได้ 19 % และถ้าเพิ่มจำนวนรอบการคำนวนมากขึ้นเป็น 150 รอบ จะมีจำนวนครั้งของ network ที่เชื่อถือได้เพิ่มขึ้นมากกว่าเดิม 2.3 %

Title	Using Hopfield Network for Solving Optimization Problems
Student	Mr. Apichart Anumat
Advisor	Dr. Arit Thammano
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2001

ABSTRACT

The computation time and the requirements for space resources usually grow with the size of the problem. Means that the time or resource depends on a problem. Computational complexity theory classifies problems to be computed as polynomial (P) and nondeterministic polynomial (NP) problems. This means a separation of problems that can be solved on a deterministic computer and problems that require a nondeterministic computer. So this paper shows the Hopfield Network for solving Optimization Problem that can find a near-optimal in a short time. A classical example of combinatorial optimization problems is the Traveling Salesperson Problem, which belongs to the class of NP problems. And for more performance, this paper have improved a new activation (output) function of the Hopfield Network for solving Optimization problems. From the practical experiments, compared with the old function using 60 random 5-TSP Distributions each with 10 runs of different setting noise added to the initial input values of the neurons, the new function reduces the error rate of tour length by 19 % and increases the percentage of valid tour by 2.3 %.

สารบัญ

สารบัญภาพ

หน้า

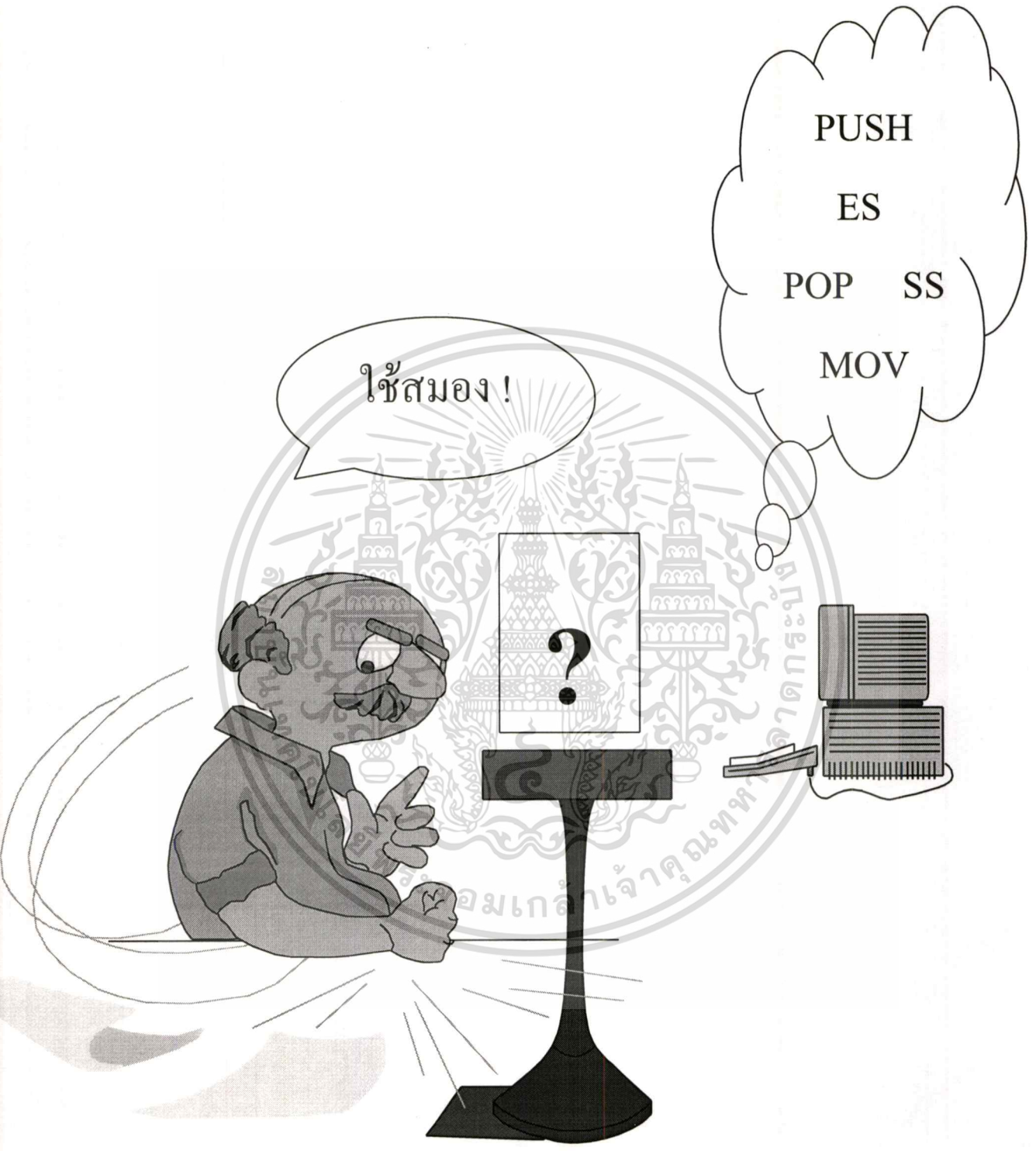
ภาพที่

บทที่ 1
ทฤษฎีเบื้องต้นของโครงข่ายประสาทเทียม

ปัจจุบันแม้ว่าคอมพิวเตอร์มีความสามารถในการประมวลผลได้สูงและถูกนำไปใช้งานหลายๆด้านได้เป็นอย่างดี ซึ่งเมื่อเทียบกับมนุษย์แล้วคอมพิวเตอร์สามารถทำงานได้เร็วกว่าในงานหลายๆด้าน เช่น การจัดการด้านฐานข้อมูล, การประมวลผลด้าน word processing, โครงข่ายคอมพิวเตอร์ (Computer Network), งานประมวลด้านการคำนวณ เป็นต้น แต่งานบางอย่างเมื่อนำมาทำด้วยคอมพิวเตอร์และเปรียบเทียบผลกับมนุษย์แล้ว มนุษย์สามารถทำงานได้ดีกว่าและเร็วกว่ามาก เช่น การจดจำภาพหน้าคน แม้ว่าคนๆนั้นจะเปลี่ยนทรงผมหรือมีสีผิวเปลี่ยนแปลงไปก็ตามมนุษย์เราก็ยังสามารถจดจำคนๆนั้นได้ ซึ่งถ้าเป็นคอมพิวเตอร์แล้วอาจจะไม่สามารถจดจำคนๆนั้นได้ อีกทั้งเวลาที่ใช้ในการประมวลผลก็ใช้เวลานานมากด้วย

จากตัวอย่างที่กล่าวข้างต้น มนุษย์เราสามารถทำการวิเคราะห์ได้อย่างรวดเร็ว เนื่องจากสถาปัตยกรรมของสมองมนุษย์มีความแตกต่างจากคอมพิวเตอร์ กล่าวคือ สมองมนุษย์ประกอบด้วยเซลประสาทจำนวนมหาศาลที่เชื่อมต่อกัน ซึ่งเซลเหล่านี้ทำหน้าที่เป็นหน่วยประมวลผลให้กับสมองมนุษย์ การส่งสัญญาณของเซลประสาทระหว่างเซลหนึ่งสู่อีกเซลหนึ่งใช้เวลาเร็วมากคือประมาณสิบส่วนล้านวินาที ซึ่งรูปแบบการส่งสัญญาณทำในลักษณะขนานกัน และ วิธีการแก้ปัญหาของสมองมนุษย์ ใช้ประสบการณ์ที่เกิดจากการเรียนรู้ในครั้งอดีตนำมาวิเคราะห์เพื่อแก้ปัญหานั้นๆ แต่สถาปัตยกรรมของคอมพิวเตอร์ประกอบด้วยหน่วยประมวลเพียงหน่วยเดียว และการทำงานใช้ชุดคำสั่งสั่งงานเป็นลำดับขั้น อีกทั้งวิธีการแก้ปัญหาต่างๆนั้นจะต้องทราบลำดับขั้นตอนการทำงานที่แน่นอนของปัญหานั้น

ดังนั้นเมื่อเราต้องการให้คอมพิวเตอร์สามารถจัดการกับปัญหาในลักษณะที่กล่าวข้างต้นจึงได้มีการนำสถาปัตยกรรมของสมองมนุษย์มาเป็นตัวแบบใหม่ในการประมวลผล ซึ่งเรียกว่า Artificial Neural Systems หรือ โครงข่ายประสาทเทียม (Neural Network)



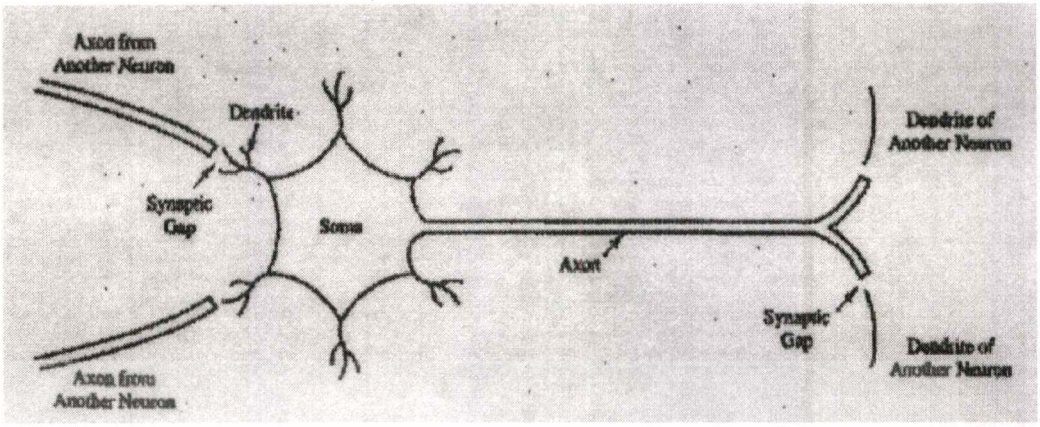
ภาพที่ 1.1 มนุษย์เปรียบเทียบกับคอมพิวเตอร์

โครงข่ายประสาทเทียมประกอบด้วยหน่วยประมวลผลแบบง่ายๆจำนวนมหาศาลที่เชื่อมต่อกัน ซึ่งเรียกหน่วยนี้ว่า นิวรอล (Neurons), เซล (Cell) หรือ โหนด (Node) และเรียกส่วนที่เชื่อมต่อกันระหว่างโหนดนี้ว่า อินเตอร์คอนเน็กชั่น (Interconnection) โดยที่อินเตอร์คอนเน็กชั่นจะเก็บความรู้ที่ใช้ในการแก้ปัญหานั้นๆของโครงข่าย ซึ่งความรู้ที่ใช้ในการแก้ปัญหานี้ได้มาจากการเรียนรู้ตัวอย่างของปัญหานั้น เราสามารถเปรียบเทียบโครงข่ายประสาทเทียมกับสมองมนุษย์ได้ในแง่ของการเก็บข้อมูลต่างๆในรูปของแพตเทิร์นของอินเตอร์คอนเน็กชั่น (Pattern of Interconnections) และลักษณะการแก้ปัญหาด้วยวิธีการเรียนตัวอย่างซึ่งเทียบได้กับการสะสมประสบการณ์ของมนุษย์

โครงข่ายประสาทเทียมได้ถูกนำมาประยุกต์ในงานด้านต่างๆหลายด้าน เช่น storing และ recall data, pattern classification, การแบ่งกลุ่มแพตเทิร์นที่มีลักษณะเหมือนกัน เป็นต้น โครงข่ายประสาทเทียมมีคุณสมบัติซึ่งเรียกว่า generalization กล่าวคือ โครงข่ายสามารถวิเคราะห์หรือสร้างคำตอบสำหรับปัญหาซึ่งมีลักษณะที่เหมือนหรือคล้ายกับตัวอย่างที่นำมาสอนได้อย่างมีประสิทธิภาพ โดยที่ปัญหานั้นไม่เคยถูกนำมาสอนให้กับโครงข่ายเลย

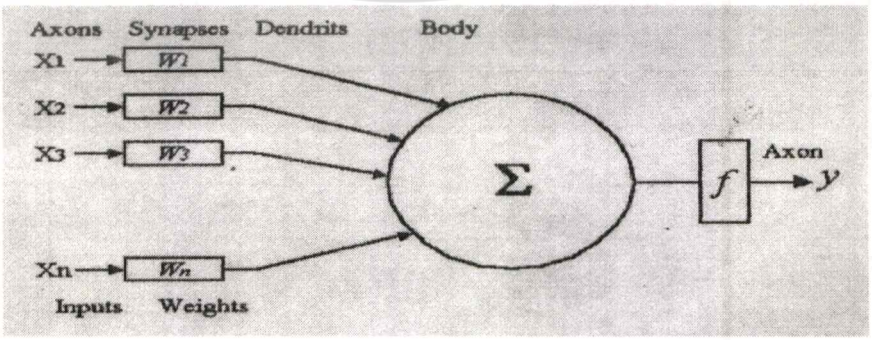
1.1 ตัวแบบของเซลประสาทเทียม (Artificial Neuron Model)

ลักษณะเซลประสาททางชีวภาพประกอบด้วย 3 ส่วนหลัก คือ เดนไดรซ์ (Dendrites), โซมา (Soma) หรือ ตัวเซล (Cell Body) และแอคซอน (Axon) ส่วนที่เชื่อมระหว่างตัวเซลกับแอคซอนของเซลอื่นๆที่อยู่รอบๆข้างเรียกว่า ไซแนปส์ (Synapse) โดยตัวเดนไดรซ์จะรับสัญญาณจากเซลประสาทที่อยู่รอบๆข้างผ่านทางไซแนปส์ด้วยปฏิกิริยาทางเคมี ซึ่งสัญญาณที่เข้ามานี้คือ ประจุอิเล็กตรอนโดยปฏิกิริยาทางเคมีที่เกิดขึ้นจะทำการปรับเปลี่ยนความถี่ของสัญญาณที่เข้ามา ตัวเซลทำหน้าที่รวมสัญญาณที่เข้ามาแล้วทำการส่งสัญญาณออกให้กับแอคซอนเพื่อผ่านต่อให้กับเซลอื่นๆ ซึ่งสัญญาณที่ออกจากตัวเซลมีลักษณะเป็นแบบสัญญาณกระตุ้นให้กับเซลอื่น กล่าวคือ ถ้าค่าสัญญาณที่ส่งออกจากตัวเซลมีจำนวน 100 ครั้งต่อวินาที ถือว่าค่าสัญญาณที่ส่งออกเป็นสถานะการกระตุ้น (Fire) โดยทั่วไปเราสนใจสัญญาณที่ออกจากเซลประสาทในรูปของสถานะการกระตุ้นและไม่กระตุ้น (Not Fire) ให้กับเซลที่อยู่รอบข้างมากกว่าสนใจเงื่อนไขที่เกี่ยวข้องที่ทำให้เซลประสาทเกิดสถานะนี้ขึ้นมา ภาพที่ 1.2 แสดงลักษณะของเซลประสาททางชีววิทยา



ภาพที่ 1.2 แสดงลักษณะของเซลประสาท

ตัวแบบเซลประสาทเทียมที่ใช้ใน Artificial Neural Network มีคุณลักษณะพื้นฐานเหมือนกับเซลประสาททางชีววิทยา ภาพที่ 1.3 แสดงลักษณะของตัวแบบเซลประสาทเทียมที่ใช้กันโดยทั่วไปซึ่งใช้พื้นฐานจากตัวแบบที่เสนอโดย McCulloch และ Pitt ในปี 1943 จากภาพที่ 1.3 เซลประสาทเทียมรับสัญญาณ $x_1 - x_n$ จากเซลรอบข้าง โดยไซแนปส์หรืออินเตอร์คอนเน็กชั่นจะทำการปรับเปลี่ยนหรือถ่วงน้ำหนักค่าสัญญาณที่เข้ามาด้วย $w_1 - w_n$ สมการที่ 2.1 แสดงการหาค่าผลรวมสัญญาณที่เข้าสู่เซลประสาทเทียม โหนดที่ i ซึ่งเรียกว่าค่า net input และ y คือค่าสัญญาณที่ออกจากเซลประสาทเทียมซึ่งจะปรากฏที่แอคซอน โดยค่าสัญญาณนี้จะส่งผ่านทางฟังก์ชั่น กระตุ้น (Activation Function) $f( )$ ซึ่งฟังก์ชั่นนี้ทำหน้าที่เลือกผ่านค่าสัญญาณให้กับแอคซอน โดยจะทำการย่อขนาดของผลรวมสัญญาณที่ออกจากตัวเซลให้อยู่ในช่วง 0 ถึง 1 ในบางครั้งอาจไม่มีการใช้ฟังก์ชั่นก็ได้ สมการที่ 2.2 แสดงการหาค่าฟังก์ชั่นกระตุ้นของสมการที่ 2.1



ภาพที่ 1.3 แสดงตัวแบบของเซลประสาทเทียม

$$net_i = \sum_j^n x_i w_{ij} \qquad \text{.....(2.1)}$$

$$y_i = f_i(net_i) \qquad i = 1,2,...m \text{ .....(2.2)}$$

โดยที่ x_j : ค่าสัญญาณลำดับที่ j ที่เข้าสู่เซลประสาทเทียมโหนดที่ i

w_{ij} : ค่าถ่วงน้ำหนักสัญญาณลำดับที่ j ที่เข้าสู่เซลประสาทเทียมโหนดที่ i

net_i : ค่า net input ของเซลประสาทเทียมโหนดที่ i

$f_i(\ )$: ฟังก์ชั่นกระตุ้นของเซลประสาทเทียมโหนดที่ i

y_i : ค่าสัญญาณที่ออกจากเซลประสาทเทียมโหนดที่ i

m : จำนวนเซลประสาทเทียมของโครงข่าย

ค่าของอินเตอร์คอนเน็กชั่นของเซลประสาทเทียมอาจมีค่าเป็นบวก, ลบ หรือ ศูนย์ก็ได้ ดังนั้นในการปรับแต่งค่าสัญญาณที่เข้าสู่เซลประสาทด้วยค่าถ่วงน้ำหนักของอินเตอร์คอนเน็กชั่น ถ้าข้อมูลที่เข้าสู่เซลประสาทเทียมมีค่าเป็นบวกและค่าของอินเตอร์คอนเน็กชั่นเป็นบวกด้วยก็จะก่อให้เกิดสภาพเสริมกัน (Excitiatory) และถ้าค่าของอินเตอร์คอนเน็กชั่นมีค่าเป็นลบก็จะลดค่าความเข้มของข้อมูลที่เข้ามานั้น ส่วนในกรณีที่ค่าของอินเตอร์คอนเน็กชั่นเท่ากับศูนย์ อินเตอร์คอนเน็กชั่นตรงส่วนนั้นหยุดทำงาน

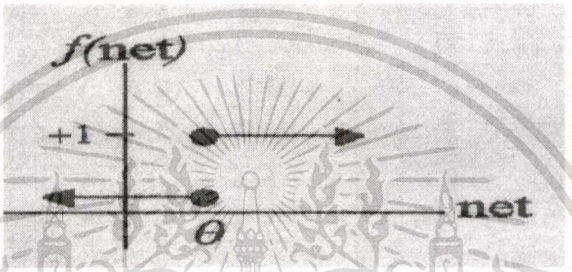
ค่าผลลัพธ์ที่ได้จากฟังก์ชั่นกระตุ้นนี้ทำให้เซลประสาทเทียมมีลักษณะเหมือนกับเซลประสาททางชีววิทยา กล่าวคือค่าผลลัพธ์ที่ได้จากฟังก์ชั่นกระตุ้นเป็นค่าที่อยู่ในช่วง 0 ถึง 1 และค่าสัญญาณที่ออกจากเซลประสาททางชีววิทยามีลักษณะเป็นสถานะของการกระตุ้น ดังนั้นเราสามารถแทนความหมายสถานะการกระตุ้นของเซลประสาทด้วยค่า 1 และ การไม่กระตุ้นด้วยค่า 0 ได้รูปแบบฟังก์ชั่นกระตุ้นที่นิยมใช้มีดังต่อไปนี้

1. Step Function ฟังก์ชั่นนี้นิยมใช้ในโครงข่ายแบบชั้นเดียว (Single Layer) ซึ่งใช้ในการแปลงค่า net input ที่อยู่ในรูปของค่าต่อเนื่องให้อยู่ในรูปของไบนารี่ (Binary) คือ 1 และ 0 , หรือไบโพลล่า (Bipolar) คือ 1 และ –1 ฟังก์ชั่นนี้จะใช้ค่าเทรสโฮลด์ (Threshold) θ ในการกำหนดการแปลงค่า ซึ่งเรียกฟังก์ชั่นนี้ว่า Threshold Function หรือ Heavisde Function ตัวอย่างโครงข่ายที่ใช้ฟังก์ชั่นนี้ได้แก่ Adaline สมการที่ 2.3 แสดงสมการของ Threshold Function

$$f(\text{net}) = \begin{cases} 1 \text{ if net} \geq \theta \\ 0 \text{ if net} < \theta \end{cases} \quad \text{.....(2.3)}$$

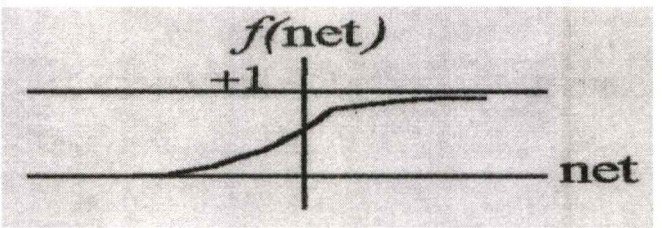
โดยที่ net : ค่า net input ที่ของเซลประสาทเทียม

θ : ค่าเทรสโฮลด์ที่ใช้ในการแปลงค่าของเซลประสาทเทียม



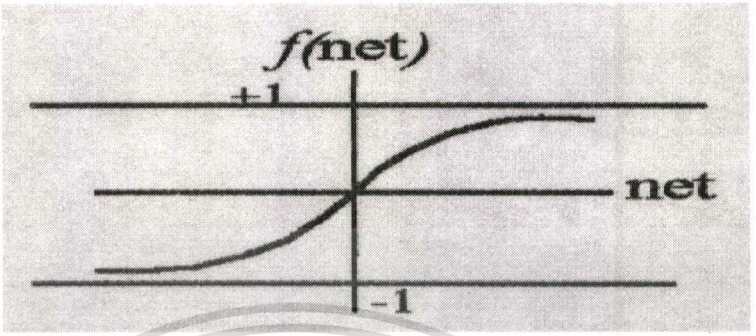
ภาพที่ 1.4 แสดงลักษณะของ Binary Step โดยใช้ค่า Threshold

2. Sigmoid Function (S-Shaped Curves) ฟังก์ชั่นนี้ปกติแล้วนิยมนำมาใช้ในการแปลงค่า net input ให้อยู่ในรูปของช่วง 0 ถึง 1 ซึ่งเรียกว่า Binary Sigmoid ฟังก์ชั่น หรืออยู่ในช่วง –1 ถึง 1 ซึ่งเรียกว่า Bipolar Sigmoid ฟังก์ชั่นที่นำมาใช้ในการแปลงค่าเป็น Sigmoid Function โดยทั่วไปแล้วใช้ Logistic Function และ Hyperbolic Tangent Function แสดงในสมการที่ 2.4 และ 2.5 ตามลำดับ ซึ่งฟังก์ชั่นเหล่านี้มีประโยชน์อย่างมากในการสอนโครงข่ายแบบแพร่ย้อนกลับ (Backpropagation Network) ภาพที่ 1.5 และ ภาพที่ 1.6 แสดงตัวอย่างของ Binary Sigmoid และ Bipolar Sigmoid ด้วย Logistic Function และ Hyperbolic Tangent Function ตามลำดับ



ภาพที่ 1.5 แสดงลักษณะของ Binary Sigmoid

$$f(net) = 1/(1+e^{-net}) \quad \text{.....(2.4)}$$



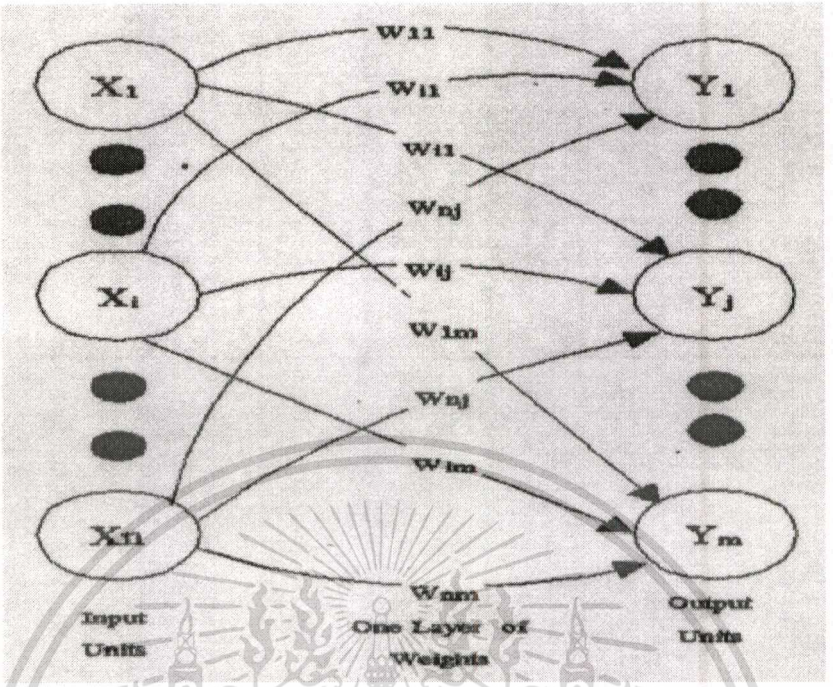
ภาพที่ 1.6 แสดงลักษณะของ Bibolar Sigmoid

$$f(net) = (e^{net} - e^{-net})/(e^{net} + e^{-net}) \quad \text{.....(2.5)}$$

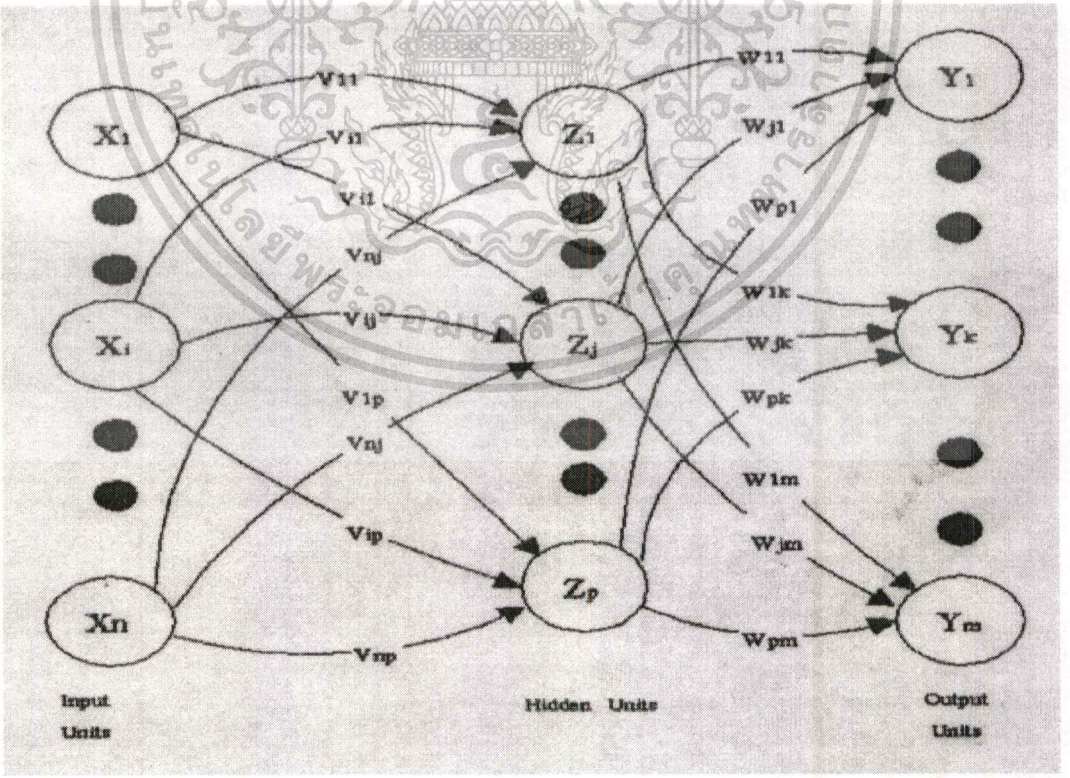
โดย net : ค่า net input ที่ได้ของเซลประสาทเทียม

1.2 สถาปัตยกรรมของโครงข่ายประสาทเทียม (Neural Network Architecture)

โครงข่ายประสาทเทียมประกอบด้วยเซลประสาทเทียมหรือโหนดจำนวนมากที่เชื่อมต่อกัน ซึ่งการเชื่อมต่อจะแบ่งออกเป็นกลุ่มย่อยเรียกว่า ชั้น (Layer) โดยทั่วไปแล้วแบ่งออกเป็น 2 แบบ คือ โครงข่ายแบบชั้นเดียว (Single Layer) และ โครงข่ายแบบหลายชั้น (Multilayer) การกำหนดจำนวนชั้นของโครงข่ายนี้ ส่วนที่เป็นหน่วยรับข้อมูลเข้า (Input Unit) จะไม่ถูกนับด้วย เนื่องจากเป็นส่วนที่ไม่มีการคำนวน (หนังสือบางเล่มถือว่าหน่วยข้อมูลเข้านี้เป็นชั้นของโครงข่ายด้วย) ดังนั้น เราอาจกล่าวได้ว่าจำนวนชั้นของโครงข่ายคือ จำนวนชั้นของอินเตอร์คอนเน็กชั่นที่ถูกถ่วงน้ำหนัก โดยปกติแล้วโครงข่ายแบบหลายชั้นสามารถแก้ปัญหาที่มีความซับซ้อนได้ดีกว่าโครงข่ายแบบชั้นเดียว และในบางปัญหานั้นไม่สามารถแก้ปัญหาด้วยโครงข่ายแบบชั้นเดียวได้ ภาพที่ 1.7 แสดงลักษณะการต่อของโครงข่ายแบบชั้นเดียว ซึ่งตัวแบบสถาปัตยกรรมแบบนี้ ได้แก่ BAM, Hopfield ส่วนภาพที่ 1.8 แสดงสถาปัตยกรรมแบบหลายชั้น ซึ่งตัวแบบได้แก่ Backpropagation, Self-Organizing Maps, Couterpropagation



ภาพที่ 1.7 แสดงตัวอย่างของโครงข่ายแบบชั้นเดียว

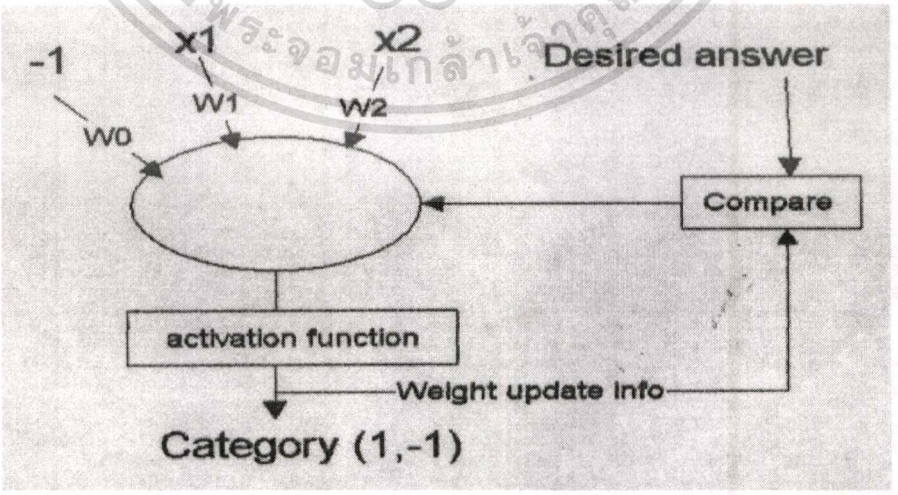


ภาพที่ 1.8 แสดงตัวอย่างของโครงข่ายแบบหลายชั้น

1.3 การฝึกสอนโครงข่ายประสาทเทียม (Training)

จากที่กล่าวข้างต้นก่อนการนำโครงข่ายประสาทเทียมไปใช้งานจะต้องมีการสร้างความรู้ (Knowledge) ให้กับโครงข่ายก่อน ซึ่งเรียกขั้นตอนนี้ว่า การฝึกสอนโครงข่าย ข้อมูลที่นำมาเป็นชุดฝึกสอนให้แก่โครงข่ายคือตัวอย่างข้อมูลของปัญหาที่ต้องการนำโครงข่ายไปใช้ในการแก้ปัญหา โดยการเรียนรู้ของโครงข่ายจะทำการปรับค่าถ่วงน้ำหนักของอินเตอร์คอนเน็กชั่นที่เชื่อมต่อกันในแต่ละชั้นของโครงข่ายเพื่อสร้างฐานความรู้ของปัญหานั้นๆให้กับโครงข่าย ลักษณะการสอนของโครงข่ายประสาทเทียม แบ่งได้เป็น 2 แบบ คือ การฝึกสอนแบบมีผู้สอน (Supervised Training) และการฝึกแบบไม่มีผู้สอน (Unsupervised Training)

การฝึกสอนแบบมีผู้สอน (Supervised Training) ข้อมูลที่นำเข้าสู่โครงข่ายด้วยการสอนวิธีนี้ ประกอบด้วย 2 ส่วน คือ อินพุตเวคเตอร์ (Input Vector) ที่เป็นตัวอย่างข้อมูลของปัญหานั้นๆ และ เวคเตอร์เป้าหมาย (Target Vector) ซึ่งเป็นผลลัพธ์ที่เราต้องการให้โครงข่ายสร้าง เมื่อมีการนำอินพุตเวคเตอร์ของตัวอย่างข้อมูลมาสอนให้กับโครงข่าย ขณะสอนโครงข่ายด้วยวิธีนี้จะมีการกำหนดค่าผลลัพธ์เป้าหมายให้กับแต่ละอินพุตเวคเตอร์ที่นำเข้ามาสอน โดยโครงข่ายจะนำค่าผิดพลาดที่ได้จากการคำนวนของโครงข่าย (Actual Output) กับผลลัพธ์เป้าหมายที่กำหนดขึ้นมา ใช้ในการเรียนหรือปรับค่าถ่วงน้ำหนักของอินเตอร์คอนเน็กชั่น ตัวแบบโครงข่ายที่การเรียนรู้ลักษณะนี้ ได้แก่ Adaline, Perceptron, Back-propagation เป็นต้น



ภาพที่ 1.9 แสดงตัวอย่างการปรับค่าถ่วงน้ำหนักของเพอร์เซปตรอน

ภาพที่ 1.9 แสดงตัวอย่างโครงสร้างการสอนของเพอร์เซปตรอน (Perceptron) เพื่อใช้ในการแยกข้อมูลสองกลุ่ม ตารางที่ 1 แสดงข้อมูลที่เข้าสู่โครงข่ายและผลลัพธ์ของข้อมูลนั้นที่นำมาสอนโครงข่าย โดยที่ x_1, x_2 เป็นข้อมูลที่เข้าสู่โครงข่าย และ F คือผลลัพธ์ที่ต้องการ จากภาพที่ 1.9 ตัวเซลของเพอร์เซปตรอนรับข้อมูลเข้า 3 ทาง คือ –1, x_1, x_2 โดยเรียกข้อมูลเข้าที่เป็นค่าคงที่ –1 (อาจเป็น +1) ว่า ไบเอส (Bias)

ข้อมูลที่เข้าสู่โครงข่าย		ผลลัพธ์ที่ต้องการ
x_2	x_2	F
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

ตารางที่ 1. แสดงข้อมูลที่นำมาสอนเพอร์เซปตรอนและผลลัพธ์ที่ต้องการ

สมการที่ 2.6 และ 2.7 แสดงการหาค่าของฟังก์ชั่นการกระตุ้นและการปรับค่าถ่วงน้ำหนักของเพอร์เซปตรอนตามลำดับ

$$\begin{aligned} y &= +1 \text{ ถ้า } net \geq 0 \\ &= -1 \text{ ถ้า } net < 0 \end{aligned} \quad \text{.....(2.6)}$$

$$W(t+1) = W(t) + \alpha(EX) \quad \text{.....(2.7)}$$

โดยที่ E = ค่าผิดพลาดที่ใช้ปรับค่าถ่วงน้ำหนัก (Desired Category-actual Category)

α = ค่า learning rate ($0 < \alpha < 1$)

W = ค่าเวคเตอร์ของค่าถ่วงน้ำหนักของเซลเพอร์เซปตรอน ที่เวลา t

X = ค่าอินพุตเวคเตอร์ (Input Vector) ที่เข้าสู่เพอร์เซปตรอน

ตัวอย่างการสอนเซลประสาทเพอร์เซปตรอน

ขั้นตอนข้างล่างนี้แสดงตัวอย่างการสอนเพอร์เซปตรอนเพื่อให้สามารถแยกแยะข้อมูลของตารางที่ 1 บางส่วน โดยกำหนดให้ค่า $w_0 = 0$, $w_1 = 0$, $w_2 = 0$ และให้ค่า α มีค่าเท่ากับ 0.5

1. $x_1 = 1$, $x_2 = 1$ และ F = 1

   net = (0)(-1) + (0)(1) + (0)(1) = 0

   y = 1 (เนื่องจาก net > 0)

2. $x_1 = 1$, $x_2 = -1$ และ F = -1

   net = (0)(-1) + (0)(1) + (0)(-1) = 0

   y = 1 (เนื่องจาก net > 0)

   ปรับค่าถ่วงน้ำหนัก

   E = -1 –1 = -2

   $w_0 = 0 + 0.5(-2)(-1) = 1$

   $w_1 = 0 + 0.5(-2)(1) = -1$

   $w_2 = 0 + 0.5(-2)(-1) = 1$

3. $x_1 = -1$, $x_2 = 1$ และ F = -1

   net = (1)(-1) + (-1)(-1) + (1)(1) = 1

   y = 1 (เนื่องจาก net > 0)

   ปรับค่าถ่วงน้ำหนัก

   E = -1 –1 = -2

   $w_0 = 1 + 0.5(-2)(-1) = 2$

   $w_1 = -1 + 0.5(-2)(-1) = 0$

   $w_2 = 1 + 0.5(-2)(1) = 0$

4. $x_1 = -1$, $x_2 = -1$ และ F = -1

   net = (2)(-1) + (0)(-1) + (0)(-1) = -2

   y = -1 (เนื่องจาก net < 0)

5. $x_1 = 1$, $x_0 = 1$ และ F = 1

net = (2)(-1) + (0)(1) + (0)(1) = -2

y = -1 (เนื่องจาก net < 0)

ปรับค่าถ่วงน้ำหนัก

E = 1 – (-1) = 2

$w_0 = 2 + 0.5(2)(-1) = 1$

$w_1 = 0 + 0.5(2)(1) = 1$

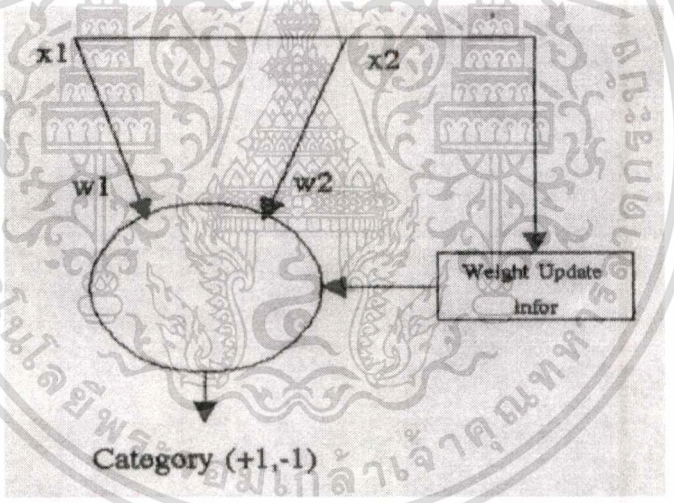
$w_2 = 0 + 0.5(2)(1) = 1$

ตารางที่ 2 แสดงผลสรุปการสอนเพอร์เซปตรอน ซึ่งการสอนมีจำนวน 3 รอบ โดยครั้งที่ 1-4, 5-8 และ 9 เป็นการสอนรอบที่ 1, 2 และ 3 ตามลำดับ จากตารางในรอบที่ 2 และ 3 พบว่า ค่าถ่วงน้ำหนักไม่มีการเปลี่ยนแปลงเลยและโครงข่ายสามารถแยกแยะข้อมูลได้ถูกต้อง ซึ่งแสดงให้เห็นว่าค่าถ่วงน้ำหนักของเซลประสาทเทียมเข้าสู่จุดสมดุลแล้วจึงทำการหยุดสอน

ครั้ง	Input		net	actual	desire	Error	New	New	New
	x_1	x_2		category	category	(E)	w_0	w_1	w_2
0	-	-	-	-	-	-	0	0	0
1	1	1	0	1	1	0	0	0	0
2	1	-1	0	1	-1	-2	1	-1	1
3	-1	1	1	1	-1	-2	2	0	0
4	-1	-1	-2	-1	-1	0	2	0	0
5	1	1	-2	-1	1	-2	1	1	1
6	1	-1	-1	-1	-1	0	1	1	1
7	1	1	-1	-1	-1	0	1	1	1
8	-1	-1	-3	-1	-1	0	1	1	1
9	1	1	1	1	1	0	1		

ตารางที่ 2. แสดงขั้นตอนของเพอร์เซปตรอนจนสามารถแยกข้อมูลที่นำมาสอน

การสอนแบบไม่มีผู้สอน (Unsupervised Training) ข้อมูลที่นำมาใช้ในการสอนโครงข่ายด้วยวิธีนี้มีเพียงอินพุตเวคเตอร์(Input Vector) ที่เป็นตัวอย่างข้อมูลของปัญหานั้นๆเพียงอย่างเดียว โดยขณะสอนจะไม่มีการกำหนดค่าผลลัพธ์เป้าหมายที่ต้องการเพื่อใช้ในการเรียนหรือปรับค่าถ่วงน้ำหนักให้กับโครงข่าย ซึ่งลักษณะการปรับค่าถ่วงน้ำหนักของโครงข่ายจะใช้ข้อมูลที่นำมาสอนในการปรับเปลี่ยนค่า โดยกลุ่มของโหนดที่ถูกปรับค่าถ่วงน้ำหนักนี้จะเป็นกลุ่มเดียวกับกลุ่มข้อมูลที่ให้ค่าผลลัพธ์คล้ายๆกัน การฝึกสอนด้วยวิธีนี้ใช้เวลาในการเรียนตัวอย่างได้รวดเร็วกว่าการฝึกสอนแบบมีผู้สอน ตัวอย่างแบบที่ใช้การเรียนลักษณะนี้ เช่น Kohonen Self-Organization Maps (SOM), Counterpropagation Network (CPN), Adaptive Resonance Theory (ART) ภาพที่ 1.10 แสดงรูปแบบการสอนแบบของโคโฮเนน (Kohonen) และสมการที่ 2.8 เป็นสมการที่ใช้ในการปรับค่าถ่วงน้ำหนักของโคโฮเนน ซึ่งจากสมการจะสังเกตุได้ว่าทิศทางของค่าถ่วงน้ำหนักปรับตามอินพุตเวคเตอร์ที่เข้าสู่โครงข่าย



ภาพที่ 1.10 แสดงตัวอย่างการปรับค่าถ่วงน้ำหนักของโคโฮเนน

$$W(t+1) = W(t) + \alpha\big(X - W(t)\big) \quad \text{.....(2.8)}$$

โดยที่ α = ค่า learning rate ($0 < \alpha < 1$)

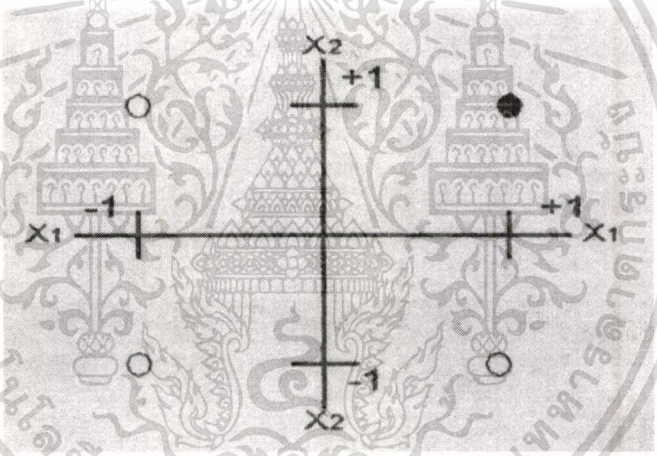
W = ค่าเวคเตอร์ของค่าถ่วงน้ำหนัก (Weight Vector) ที่เวลา t

X = ค่าอินพุตเวคเตอร์ (Input Vector) ที่เข้าสู่โครงข่าย

1.4 การแบ่งแบบเชิงเส้น (Linear Separability)

ปกติทั่วไปแล้วผลลัพธ์ที่ออกจากเซลประสาทเทียมเราต้องการคำตอบในรูปของตรรกศาสตร์ คือ ใช่หรือไม่ใช่ กล่าวคือคำตอบใช่เมื่อข้อมูลที่เข้าสู่เซลประสาทเทียมเป็นสมาชิกของกลุ่มนั้น และไม่ใช่เมื่อข้อมูลนั้นไม่ได้เป็นสมาชิก ซึ่งเทียบกับผลลัพธ์ที่ได้จากเซลประสาทเทียม คือ 1 และ –1 (0) การที่เราต้องการให้โครงข่ายสร้างคำตอบในลักษณะนี้ต้องใช้ฟังก์ชั่นกระตุ้นแบบ Step Function เนื่องจากผลลัพธ์ที่ออกจากฟังก์ชั่นนี้มีสองสถานะเช่นกัน

จากตัวอย่างการสอนเพอร์เซปตรอนข้างต้นเราสามารถแสดงความสัมพันธ์ของกลุ่มข้อมูลที่นำมาฝึกสอนด้วยกราฟภาพที่ 1.11 โดยจุดดำแสดงเวคเตอร์ (1,1) ซึ่งให้ผลลัพธ์คือ 1 และจุดขาวแสดงเวคเตอร์ (1,-1), (-1,1) และ (-1,-1) ซึ่งให้ผลลัพธ์ –1



ภาพที่ 1.11 แสดงความสัมพันธ์ของข้อมูลเชิงเรขาคณิตของตัวอย่างเพอร์เซปตรอน

ซึ่งจากตัวอย่างนี้สามารถเขียนสมการหาค่า net input ได้ดังสมการที่ 2.9

$$net = \sum_{i=1}^{n} x_i w_i - w_0 \quad \text{เมื่อ } x_0 \text{ เท่ากับ } -1 \quad \text{.....(2.9)}$$

โดยที่

$$f(net) = +1 \text{ ถ้า net} \geq 0$$

$$= -1 \text{ ถ้า net} < 0$$

และเรียกขอบเขตระหว่างพื้นที่ $f(net) > 0$ และ $f(net) < 0$ ว่าขอบเขตการตัดสินใจ (Decision Boundary) ซึ่งเราสามารถหาเส้นแบ่งได้จากสมการที่ 2.10

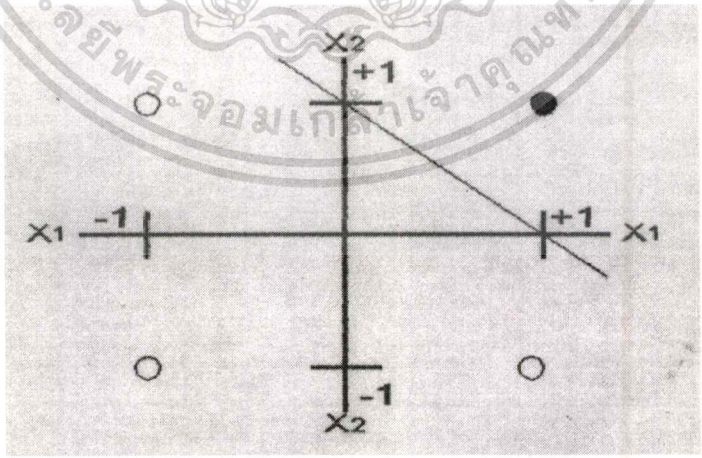
$$\sum_{i=1}^{n} x_i w_i - w_0 = 0 \quad \text{.....(2.10)}$$

ซึ่งสมการที่ 2.10 เป็นสมการเส้นตรง (line), เพลน (plane) หรือ ไฮเปอร์เพลน (hyprplane)

เนื่องจากข้อมูลที่เข้าสู่เพอร์เซปตรอนมีลักษณะเป็นคู่คือ (x_1, x_2) และค่าของน้ำหนักที่ได้จากการสอนครั้งสุดท้ายคือ $w_0 = 1$, $w_1 = 1$ และ $w_2 = 1$ เมื่อนำมาแทนค่าในสมการของเส้นขอบเขตการตัดสินใจแสดงได้ดังสมการที่ 2.11 นี้

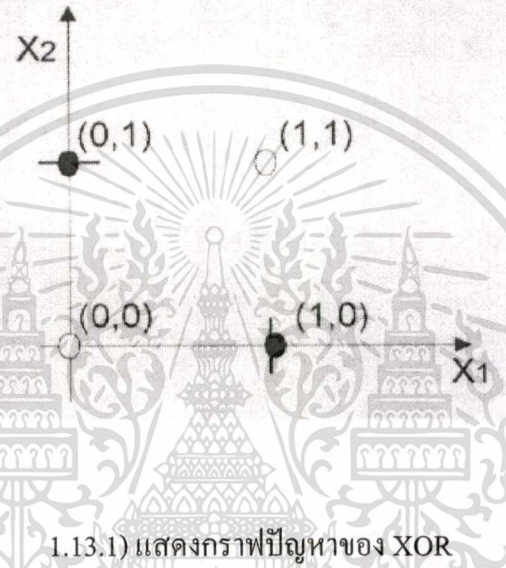
$$x_1 + x_2 = 1 \quad \text{.....(2.11)}$$

จากการแทนค่าสมการของเส้นแบ่งขอบเขตการตัดสินใจข้างบน เมื่อนำมาวาดเส้นตรงแสดงดังภาพที่ 1.12 จากภาพที่ 1.12 จะเห็นว่าเส้นตรงนั้นสามารถแบ่งขอบเขตระหว่างข้อมูลทั้งสองกลุ่มได้ ซึ่งเราเรียกลักษณะแบ่งแบบนี้ว่า การแบ่งแบบเชิงเส้น (Linear Separability)

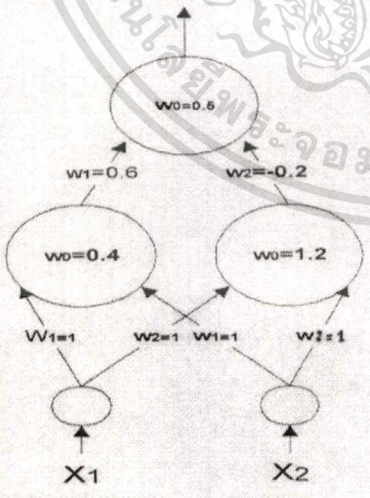


ภาพที่ 1.12 กราฟเส้นแบ่งขอบเขตที่ได้จากสอนเพอร์เซปตรอน

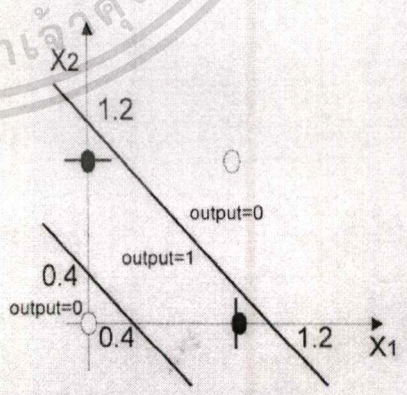
จากตัวอย่างปัญหาของเพอร์เซปตรอนข้างต้นเป็นการทำโอเปอร์เรชั่น (Operation) AND ซึ่งมีลักษณะเป็นแบบเชิงเส้น แต่กรณีที่ลักษณะของปัญหาเป็นแบบไม่เชิงเส้น (Nonlinear) เช่น XOR แสดงดังกราฟภาพที่ 1.13.1 ซึ่งเพอร์เซปตรอนไม่สามารถแก้ปัญหานี้ได้ ดังนั้นเพื่อให้สามารถแก้ปัญหานี้ได้จึงมีการเพิ่มชั้นของโครงข่ายแสดงดังภาพที่ 1.13.2 ภาพที่ 1.13.3 แสดงเส้นแบ่งที่ได้จากการสอนเพอร์เซปตรอนที่เพิ่มชั้นเพื่อใช้แยกข้อมูลปัญหา XOR



1.13.1) แสดงกราฟปัญหาของ XOR



1.13.2) แสดงการเพิ่มชั้นของเพอร์เซปตรอน



1.13.3) แสดงเส้นแบ่งที่ได้จากการสอนภาพที่ 1.13.1

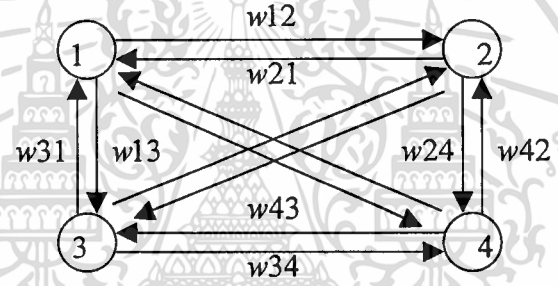
ภาพที่ 1.13 แสดงการแก้ปัญหาของ XOR ด้วยเพอร์เซปตรอนที่เพิ่มชั้น

บทที่ 2

Hopfield Network

Hopfield Network

Hopfield Network หรือ Hopfield Memory ได้พัฒนาต่อมาจาก BAM (Bidirectional Associative Memory) ซึ่ง BAM จะเป็น Network 2 layers ทุกๆสมาชิกใน layer หนึ่ง จะเชื่อมต่อกับทุกๆสมาชิกในอีก layer หนึ่ง เมื่อเป็น Hopfield Network จะลดขนาดของ BAM เหลือเพียง layer เดียว และเพื่อไม่ให้ weight matrix มีค่าเป็น 0 จึงได้มีการเพิ่ม external input ให้แก่ทุกๆสมาชิก



ภาพที่ 2.1 Layout of A Hopfield Network

ในภาพแสดง Hopfield Network ด้วย 1 layer ประกอบสมาชิก 4 neurons แต่ละสมาชิกต่างก็เชื่อมต่อซึ่งกันและกัน

2.1 Discrete Hopfield Network

จะได้ Weight ของ Hopfield

$$w = \sum_{l=1}^{L} x_l x_l^{'}$$

output เมื่อมีการเพิ่ม external input (I_i) เข้าไป

$$net_i = \sum_{j=1}^{n} w_{ij} x_j + I_i$$

เราสามารถให้ค่า threshold (U_i) กับ output ดังนี้

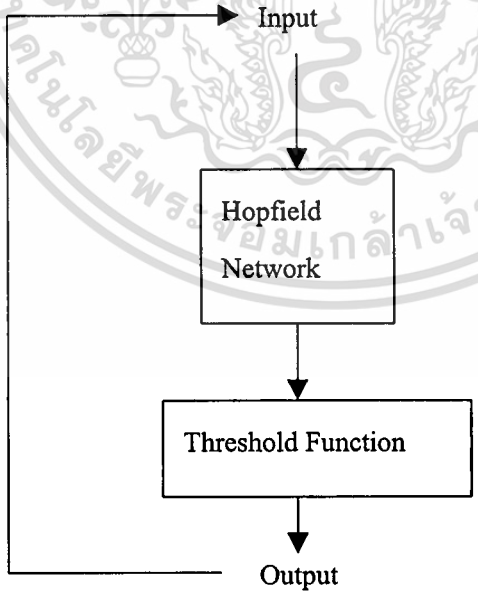
$$x_i(t+1) = \begin{cases} +1 & net_i > U_i \\ x_i(t) & net_i = U_i \\ -1 & net_i < U_i \end{cases}$$

เมื่อรวมเข้าด้วยกัน ดังนั้นเราจะได้ energy function ของ Hopfield Network คือ

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} v_i w_{ij} v_j - \sum_i I_i v_i + \sum_i U_i v_i$$

2.2 Continuous Hopfield Network

Hopfield Network เรียกว่าเป็น recurrent network แบบหนึ่ง ซึ่งก็คือ output ของ network สามารถ feed back กลับเข้าไปเป็น input ของ network ได้อีก



ภาพที่ 2.2 Feed Back in the Hopfield Network

แต่ละรอบ ค่า PE output function คือแต่ละรอบ ค่า PE output function คือ

$$v_i = g_i(\lambda u_i) = \frac{1}{2}(1 + \tanh(\lambda u_i))$$

λ เป็น gain parameter ซึ่งเป็นค่าคงที่

Hopfield Network จะมีการคงสถานะ ที่จุดๆหนึ่งเสมอ นั่นก็คือ เมื่อ output ถูก feed back กลับเข้าไปเป็น input แล้ว ไม่สามารถเปลี่ยนแปลงค่า output ได้อีกต่อไป

2.3 การสร้าง Neural Network

สิ่งที่ควรคำนึงถึงในการสร้างโครงข่ายนิวรอล Neural Network ประกอบด้วย 3 ส่วนหลักๆ ดังนี้ สิ่งแรกคือ โครงสร้าง Structure

โครงสร้างของ network จะหมายถึง network นี้ประกอบด้วยกี่ layers และมี functions อะไรบ้าง เช่น function สำหรับ input , output และยังรวมถึงการเชื่อมต่อของแต่ละ neurons ใน network เป็นอย่างไรด้วย

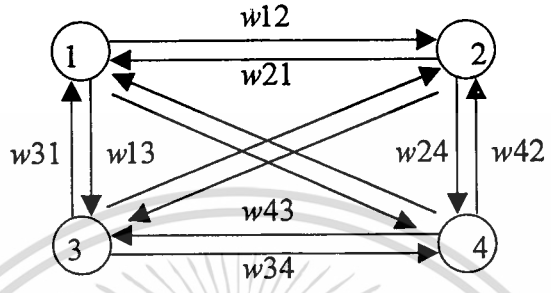
ส่วนที่สอง encoding หมายถึงรูปแบบที่ใช้สำหรับการตัดสินใจและเปลี่ยนแปลงค่าของ weights ระหว่างการเชื่อมต่อของ neurons

ส่วนสุดท้าย recall ซึ่งเป็นส่วนสำคัญใน neural network การ recall จะหมายถึงการได้ค่า output ตามที่คาดไว้ สำหรับ input ที่กำหนดมา นั่นคือถ้ามีการให้ input ที่เหมือนกับ input อันแรกเข้าสู่ network ก็ควรจะได้ผลลัพท์ output ที่เหมือนกัน แบบของ recall สามารถจำแนก network ได้เป็น autoassociative หรือเป็น heteroassociative autoassociative คือ ตัว input vector เอง เป็น output ออกมา ขณะที่ heteroassociative จะ recall vector อีกชุดหนึ่งที่กำหนดมา

ทั้งสามส่วนนี้เป็นส่วนสำคัญที่สามารถแบ่งแยกความแตกต่างของแต่ละ neural networks ออกมาอย่างชัดเจน เช่น architecture ที่ใช้กำหนด topology หรือ algorithms ที่ใช้ในการ encoding process

2.4 ตัวอย่าง Hopfield Network

ตัวอย่าง neural network ที่นำเสนอนี้คือ a Hopfield Network ประกอบด้วย 4 neurals ใน หนึ่ง layer ซึ่งต่างก็เชื่อมต่อซึ่งกันและกัน ดังภาพ



Layout of a Hopfield Network

เราจะทำการป้อน input pattern เข้าไปใน network และ recall ออกมา input pattern ที่ใช้ จะเป็น binary pattern ประกอบด้วย 0 กับ 1

ทั้งสอง input patterns ที่เราต้องการให้ network recall ออกมา คือ

A = (1,01,0) , B = (0,1,0,1) ทั้ง สอง vectors นี้ถูกเรียกว่าเป็น orthogonal กัน เพราะว่า ผล คูณ dot product ได้ผลลัพท์ เป็น 0 A1B1 + A2B2 + A3B3 + A4B4 = (1*0 + 0*1 + 1*0 + 0*1) = 0

ให้ matrix W แสดงค่า weights ของการเชื่อมต่อของ network

$$W = \begin{bmatrix} 0 & -3 & 3 & -3 \\ -3 & 0 & -3 & 3 \\ 3 & -3 & 0 & -3 \\ -3 & 3 & -3 & 0 \end{bmatrix}$$

เรากำหนดให้ threshold function เป็นดังนี้ ให้ θ มีค่าเป็น 0

$$f(t) = \begin{cases} 1 & \text{if } t >= \theta \\ 0 & \text{if } t < \theta \end{cases}$$

network นี้ เรามี 4 neurons ใน layer เดียว

ต่อไปเราจะทำการคำนวน activations ของแต่ละ neuron as the weights sum of its inputs

the activations ของ node แรก ก็คือ ผลคูณ dot product ของ input vector กับ column แรก ของ weight matrix, the activations ของ node ที่ 2 ก็ผลคูณ input vector กับ column ที่ 2 ทำอย่างนี้ไปเรื่อยๆ เราก็จะได้ activations ของทุกๆ node

output ของ network ก็คำนวนได้จาก การใช้ threshold function กับ activations ของแต่ละ node ที่คำนวนได้มา

ตัวอย่างถ้าเราส่ง input A เข้าไป เราจะได้ activations ของแต่ละ node เป็นดังนี้

0 คูณ (1 0 1 0) ได้ 3 และ -6 3 -6 ตามลำดับ

-3

3

-3

เมื่อคำนวนกับ threshold function จะได้ output (1 0 1 0) ซึ่งตรงกับ input A ก็หมายความว่า network สามารถ recall pattern A

เช่นเดียวกับ input A ถ้าเราส่ง input B เข้าไปก็จะได้ activations ของแต่ละ node เป็น –6 3 -6 3 และได้ output เป็น 0 1 0 1 ซึ่งตรงกับ input B

ดังนั้นแสดงว่า weights matrix ชุดนี้ทำงานได้ถูกต้อง เราไม่จำเป็นต้องปรับมันอีก

บทที่ 3

Optimization Problems

3.1 Nonlinear Optimization

Nonlinear optimization problem เป็นการศึกษาหาวิธีการ algorithms ที่มีประสิทธิภาพ ที่ใช้ในการ solve problem สำหรับแก้ปัญหาที่มีความซับซ้อนที่จะหาคำตอบ

optimization problem ประกอบด้วย objective function และ set of constraints on the variables วัตถุประสงค์ก็คือ หาค่าของตัวแปร variables ซึ่งนำไปสู่ optimum value สำหรับ objective function นั้นๆ โดยที่อยู่ภายให้ constraints ที่กำหนด

objective functions สามารถเป็นได้ ทั้ง linear และ nonlinear

- linear objective function = Maximize $z = 3x_1 + 4x_2 + 5.7x_3$
- linear equality constraints = $13x_1 - 4.5x_2 + 7x_3 = 22$
- linear inequality constraints = $3.6x_1 + 8.4x_2 - 17x_3 \leq 10.9$
- nonlinear objective function = Minimize $z = 5x^2 + 7xy + y^2$
- nonlinear equality constraints = $4x + 3xy + 7y + 2y^2 = 37.6$
- nonlinear inequality constraints = $4.8x + 5.3xy + 6.2y^2 \geq 34.56$

3.2 Traveling Salesperson Problem

Traveling Salesperson Problem จัดอยู่ในกลุ่ม NP complete problems (nondeterministic polynomial) ซึ่งก็หมายความว่า ทุกๆ algorithms ที่ใช้กับปัญหานี้จะเข้าใกล้กับการเป็น impractical เมื่อใช้กับปัญหาที่เกิดขึ้นจริง เมื่อใช้ neural network เข้ามาช่วยในการแก้ปัญหา จะสามารถใช้เวลาได้น้อยกว่า การใช้วิธีอื่นๆ ในแบบของ digital computer

ข้อกำหนดของ Traveling Salesperson Problem

- a traveling salesperson จะมีเมืองที่จะเข้าไปเยี่ยมอยู่จำนวนหนึ่ง
- ลำดับของการเข้าไปเยี่ยมแต่ละเมืองจะเรียกว่า tour
- a traveling salesperson จะต้องเข้าไปเยี่ยมให้ครบทุกๆเมือง โดยแต่ละเมืองเข้าไปเยี่ยมได้เพียงครั้งเดียว เสร็จแล้วกลับมายังเมืองเริ่มต้น
- ให้หา tour ที่ใช้ระยะทางน้อยที่สุด

สำหรับ n เมือง ให้ x_{ij} เป็น variable ซึ่งจะมีค่า 1 ถ้า salesperson เดินทางจากเมือง i ไปเมือง j

และมีค่าเป็น 0 ถ้า salesperson ไม่ได้เดินทางจากเมือง i ไปเมือง j

d_{ij} เป็นระยะทางจากจากเมือง i ไปเมือง j

Minimize the objective function : $\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} d_{ij}$

Subject to : $\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1$ for each j = 1 ..... n

$\sum_{\substack{i=1 \\ j\neq i}}^{n} x_{ij} = 1$ for each i = 1 ..... n

$\sum x_{ij} = 0$ for 1 for all i and j

cost ของ tour ก็คือ ระยะทางที่ใช้ทั้งหมด ที่เป็นระยะที่น้อยที่สุดด้วย ในการเดิน tour total distance คำนวนจาก การรวมระยะจากเมืองหนึ่งไปยังอีกเมือง ใน objective function จะมีอยู่ term หนึ่งที่ใช้แทน total distance , term ที่เหลือแต่ละ term จะหมายถึง constraint แต่ละอัน ตามลำดับ เมื่อนำเอา neural network เข้ามาแก้ปัญหานี้ objective function โดยทั่วๆไปก็จะหมายถึง energy function ของ network เป้าหมายก็คือ minimize energy function

เมื่อเราแทนค่า energy function ด้วย E และสัมประสิทธิ์แต่ละ term ด้วยค่าคงที่ A1 A2 A3 และ A4

d_{ij} เป็นระยะทางจากจากเมือง i ไปเมือง j ดังนั้น

$$E = A_1 \sum_i \sum_k \sum_{\substack{j \\ j\neq k}} x_{ik} x_{ij} + A_2 \sum_i \sum_k \sum_{\substack{j \\ j\neq k}} x_{ki} x_{ji} + A_3 \left[\left(\sum_i \sum_k x_{ik} \right) - n \right]^2$$

$$+ A_4 \sum_k \sum_{\substack{j \\ j\neq k}} \sum_i d_{kj} x_{ki} \left(x_{j,i+1} + x_{j,i-1} \right)$$

จะเห็นว่า E เป็น nonlinear function of the x's ดังนั้นเราจึงเรียก traveling Salesperson problem เป็น a nonlinear optimization problem

term แรก term ที่มี A1 เป็นสัมประสิทธ์ จะหมายถึง constraint no more than one neuron in the same row can output 1

term ที่สอง term ที่มี A2 เป็นสัมประสิทธ์ จะหมายถึง แต่ละเมืองในแต่ละ tour จะ visit only once

term ที่สาม term ที่มี A3 เป็นสัมประสิทธ์ จะหมายถึง มี minimum value 0 ,which is attained if and only if exactly n of the n2 x's have value 1 and the rest 0

term สุดท้าย term ที่มี A4 เป็นสัมประสิทธ์ จะหมายถึง total distances travel

3.3 Example of a Traveling Salesperson Problem for hand Calculation

ตัวอย่าง มี 4 เมือง โดยมีระยะห่างแต่ละเมือง ดังนี้

$$d = \begin{matrix} 0 & 10 & 14 & 7 \\ 10 & 0 & 6 & 12 \\ 14 & 6 & 0 & 9 \\ 7 & 12 & 9 & 0 \end{matrix}$$

tour 1 1-2-3-4-1 ไปเมืองที่ 2 ก่อน แล้วตามด้วย 3 4 แล้วกลับเมือง 1

เมืองที่ 2 เทียบได้กับ array (1,0,0,0) เพราะเป็นเมืองแรก ดังนั้น

เมืองที่ 3,4,1 เป็น (0,1,0,0),(0,0,1,0),(0,0,0,1) ตามลำดับจะได้

x21 = 1, x22 = 0, x23 = 0, x24 = 0

total distance of the tour is d12 + d23 + d34 + d41 = 10 + 6 + 9 +7 = 32

tour 2 1-3-4-2-1

tour 3 1-4-2-3-1

ตามหลักแล้วควรมี 3 tour จะเห็นว่ามี tour ที่เป็นไปได้อีก

tour 4 1-2-4-3-1

tour 5 3-2-4-1-3

tour 6 2-1-4-3-2

จะเห็นว่า tour 4-6 เป็นการซ้ำซ้อนกับ tour ต้นๆ ดังจะแสดง ดังนี้

จากสูตร

$$E = A_1 \sum_i \sum_k \sum_{j \neq k} x_{ik} x_{ij} + A_2 \sum_i \sum_k \sum_{j \neq k} x_{ki} x_{ji} + A_3 \left[\left(\sum_i \sum_k x_{ik} \right) - n \right]^2 + A_4 \sum_k \sum_{j \neq k} \sum_i d_{kj} x_{ki} \left(x_{j,i+1} + x_{j,i-1} \right)$$

พิจารณา term สุดท้ายโดย ขยายออกมา จะได้

$$\begin{aligned}
A_4 \{ & d_{12}[x_{12}(x_{23}+x_{21})+x_{13}(x_{24}+x_{22})+x_{14}(x_{21}+x_{23})]+ \\
& d_{13}[x_{12}(x_{33}+x_{31})+x_{13}(x_{34}+x_{32})+x_{14}(x_{31}+x_{33})]+ \\
& d_{14}[x_{12}(x_{43}+x_{41})+x_{13}(x_{44}+x_{42})+x_{14}(x_{41}+x_{43})]+ \\
& d_{21}[x_{21}(x_{12}+x_{14})+x_{23}(x_{14}+x_{12})+x_{24}(x_{11}+x_{13})]+ \\
& d_{23}[x_{21}(x_{32}+x_{34})+x_{23}(x_{34}+x_{32})+x_{24}(x_{31}+x_{33})]+ \\
& d_{24}[x_{21}(x_{42}+x_{44})+x_{23}(x_{44}+x_{42})+x_{24}(x_{41}+x_{43})]+ \\
& d_{31}[x_{31}(x_{12}+x_{14})+x_{32}(x_{13}+x_{11})+x_{34}(x_{11}+x_{13})]+ \\
& d_{32}[x_{31}(x_{22}+x_{24})+x_{32}(x_{23}+x_{21})+x_{34}(x_{21}+x_{23})]+ \\
& d_{34}[x_{31}(x_{42}+x_{44})+x_{32}(x_{43}+x_{41})+x_{34}(x_{41}+x_{43})]+ \\
& d_{41}[x_{41}(x_{12}+x_{14})+x_{42}(x_{13}+x_{11})+x_{43}(x_{14}+x_{12})]+ \\
& d_{42}[x_{41}(x_{22}+x_{24})+x_{42}(x_{23}+x_{21})+x_{43}(x_{24}+x_{22})]+ \\
& d_{43}[x_{41}(x_{32}+x_{34})+x_{42}(x_{33}+x_{31})+x_{43}(x_{34}+x_{32})]\}
\end{aligned}$$

โดยถ้าเรา แทนค่า A4 ด้วย ½ การคำนวน tour 1-2-3-4-1 จะเป็นดังนี้

$$
\begin{aligned}
&\tfrac{1}{2}\{10[0(0+1)+0(0+0)+1(1+0)]+\\
&\quad 14[0(0+0)+0(0+1)+1(0+0)]+\\
&\quad 7[0(1+0)+0(0+0)+1(0+1)]+\\
&\quad 10[1(0+1)+0(1+0)+0(0+0)]+\\
&\quad 6[1(1+0)+0(0+1)+0(0+0)]+\\
&\quad 12[1(0+0)+0(0+0)+0(0+1)]+\\
&\quad 14[0(0+1)+1(0+0)+0(0+0)]+\\
&\quad 6[0(0+0)+1(0+1)+0(1+0)]+\\
&\quad 9[0(0+0)+1(1+0)+0(0+1)]+\\
&\quad 7[0(0+1)+0(0+0)+1(1+0)]+\\
&\quad 12[0(0+0)+0(0+1)+1(0+0)]+\\
&\quad 9[0(1+0)+0(0+0)+1(0+1)]\}\\
&=1/2(10+0+7+10+6+0+0+6+9+7+0+9)\\
&=1/2(64)\\
&=32
\end{aligned}
$$

และจากทั้ง 6 tour จะได้

Tour #	No-zero x's	Value for the last term	Energy level	comment
1	$x_{14}, x_{21}, x_{32}, x_{43}$	32	32	1-2-3-4-1 tour
2	$x_{14}, x_{23}, x_{31}, x_{42}$	45	45	1-3-4-2-1 tour
3	$x_{14}, x_{22}, x_{33}, x_{41}$	39	39	1-4-2-3-1 tour
4	$x_{14}, x_{21}, x_{33}, x_{42}$	45	45	1-2-4-3-1 tour
5	$x_{13}, x_{21}, x_{34}, x_{42}$	39	39	3-2-4-1-3 tour
6	$x_{14}, x_{24}, x_{33}, x_{42}$	32	32	2-1-4-3-2 tour

บทที่ 4

Hopfield Network กับ Traveling Salesperson Problem

4.1 Formulation of Objective Function

การใช้ Hopfield Network กับ Traveling Salesperson Problem

Hopfield Network จะประกอบด้วย สมาชิก neurons จำนวนหนึ่ง และ จะมี links เชื่อมต่อ neurons ซึ่งกันและกัน กรณี Traveling Salesperson Problem ถ้าให้มีจำนวนเมืองที่จะไปเยี่ยม N เมือง ดังนั้นจะเกิดเป็น N x N neurons ใน Hopfield Network เชื่อมต่อซึ่งกันและกัน จำนวน row หมายถึงเมือง, จำนวน column จะหมายถึงลำดับการ tour, weights ของแต่ละ links ที่เชื่อมต่อกันจะหมายถึง constraints และ cost ของ function ซึ่ง weights และ cost กรณี TSP จะถูกแทนค่าด้วย energy function ดังนี้

$$E = A_1 \sum_i \sum_k \sum_{j \neq k} x_{ik} x_{ij} + A_2 \sum_i \sum_k \sum_{j \neq k} x_{ki} x_{ji} + A_3 \left[\left(\sum_i \sum_k x_{ik} \right) - n \right]^2$$

$$+ A_4 \sum_k \sum_{j \neq k} \sum_i d_{kj} x_{ki} \left(x_{j,i+1} + x_{j,i-1} \right)$$

ซึ่งสองเทอมแรกจะหมายถึงไม่มีเมืองใดเข้าเยี่ยมเกินหนึ่งครั้ง เทอมที่สามหมายถึงต้องเข้าเยี่ยมครบทุกๆเมือง และ เทอมสุดท้ายก็คือระยะทางระหว่างเมืองที่เชื่อมต่อกันซึ่งก็คือ cost function นั่นเอง จะเห็นว่าค่า E เป็น Nonlinear Function ดังนั้นเราเรียกสูตรของ Traveling Salesperson เป็น Nonlinear Optimization Problem

4.2 Inputs, Activations, Outputs and their Updating

การหา Inputs ของ Network สามารถกระทำได้ตามความพอใจ ตราบใดที่ผลที่เกิดขึ้นจากการใช้ Inputs นั้น สามารถทำให้เกิด activations ทำงานได้ Outputs สำหรับทุกๆเมืองและนำไปสู่การแก้ปัญหาได้ ก็จะถือว่าเป็นการ legal tour ที่เชื่อถือได้ แต่ปัญหาที่อาจเกิดขึ้นคือการที่ Network จะติดอยู่กับ local minimum ทางแก้ไขก็คือทำการ random noise ให้กับ Inputs โดยทั่วๆไปแล้ว input ของแต่ละ neuron จะเป็นจำนวนกี่เท่าของจำนวนเมืองบวกด้วย random noise เพื่อที่จะทำให้แต่ละ neuron มีความแตกต่างกัน

การ update activation ของสมาชิกแต่ละรอบ

การ update activation ของ neuron แถวที่ i, คอลัมภ์ที่ j

$$a_{new} = a_{old} + \Delta a_{ij}$$

output neuron แถวที่ i, คอลัมภ์ที่ j คำนวนจาก

$$x_{ij} = (1 + \tanh(\lambda a_{ij}))/2$$

4.3 การคำนวน

Hopfield เป็น Network ที่พยายามจะ solve ปัญหา Traveling Salesperson Problem ประกอบด้วย n*n neurons เป็น array 2 มิติ neurons ทุกตัวเชื่อมซึ่งกันและกัน

เราจะแทนที่ neuron แต่ละตัวด้วย subscripts 2 ตัว ตัวหนึ่งใช้แทนเมือง อีกตัวหนึ่งใช้แทนลำดับ order of that city in the tour

ดังนั้นเราจะได้การแทนที่ค่าของ weights ระหว่างเมืองเป็น

$$w_{ik,lj} = -A_1\delta_{il}(1-\delta_{kj}) - A_2\delta_{kj}(1-\delta_{il}) - A_3 - A_4 d_{il}(\delta_{j,k+1} + \delta_{j,k-1})$$

จะหมายถึง weights ระหว่าง neuron ik และ lj

เราใช้ Kronecker Delta function เพื่อความสะดวกในการแทนที่ valid เมือง สัญลักษณ์

δ_{ik} จะมีค่า 1 ถ้า i = k และจะมีค่าเป็น 0 ถ้า i ไม่เท่ากับ k

ซึ่งเราจะใช้ในการคำนวนหา weights matrix

Initial Activations

เอา weights matrix ที่ได้ ไปคูณกับ random noise input vector ก็จะได้ initial activations ออกมา การที่เรา add noises เข้าไปก็เพื่อที่จะไม่ให้ matrix สมมาตร

Initial Outputs

การหา initial output ออกมา โดย initial acivations คำนวณ ผ่าน output function ซึ่ง hopfield and tank คำนวนไว้ คือ output $x_{ij} = (1 + \tanh(\lambda a_{ij}))/2$

ที่เปลี่ยนใหม่ใช้ $x_{ij} = (1 + \tanh(\lambda a_{ij})^* x0)/2$ หารด้วย $(1 + \tanh \lambda x0)$ โดย x0 มีค่า = 0.05

Final Outputs

จากนั้นนำไปวน loop ตามจำนวนรอบที่กำหนด โดยแต่ละรอบที่วนก็จะทำการ update activations โดย $a_{new} = a_{old} + \Delta a_{ij}$ และ คำนวณ output เหมือนเดิม

Find Tour

เมื่อครบกำหนด loop แล้ว จะสามารถ หา tour ออกมาโดย ใช้ค่าที่ได้จาก final outputs การหาค่าที่มากที่สุดของ output ของแต่ละรอบออกมาตามลำดับ เช่นรอบที่ เมือง เป็น 2 ได้ค่า output ตำหน่งแถวที่ 2 ,คอลัมภ์ที่ 5 ออกมาเป็นค่าที่มากที่สุด (โดยที่ไม่ซ้ำกับคอลัมภ์ที่ได้ค่ามากที่สุดในแถวแรก) แสดงว่า เมืองที่ 2 เป็นการไป tour เป็น ลำดับที่ 5 เมื่อได้ครบทุกแถวแล้วก็สามารถเรียงลำดับการทัวร์ได้

การคำนวนค่า Error Distances

การทดลองชุดหนึ่งๆ ประกอบด้วย tour หลายๆ tour แต่ละ tour จะทำการ add noises input vectors หลายชุดเช่นกัน แต่ละรอบคำนวน Error โดย (distance – optimal) / optimal หนึ่ง tour จะมีค่า Errors ค่าหนึ่ง ผลรวมของ Errors แต่ละรอบ tour หลายๆ tours คำนวณ Errors โดยรวม Errors ของแต่ละ Tour เข้าด้วยกัน แล้วหารด้วยจำนวน tours

การคำนวนค่า Valid

การคำนวนค่า Valid นี้เป็นเพียงการคำนวนหาความเสถียรของ Network เท่านั้น ซึ่งข้อจำกัดที่พบจากการทดลองได้แก่เมื่อมีการวน loop มากๆรอบขึ้นจะได้ final outputs ออกมาเป็นค่า 0 ทั้งหมด เนื่องจากในโปรแกรมภาษา Visual Basic ไม่มี function tanh ดังนั้นจึงแทนค่า tanh(x) ด้วย function exp(x)- exp(-x) หารด้วย exp(x)+ exp(-x) ซึ่งค่า x จะเชื่อถือได้ถ้าอยู่ในช่วง –744 ถึง 709

การคำนวนค่า Optimal

คำนวนโดยใช้วิธีที่คาดว่าจะมีความเป็นไ ปได้ทั้งหมดมาคำนวนหา Optimal

ตัวอย่าง

$$d = \begin{matrix} 0 & 10 & 14 & 7 \\ 10 & 0 & 6 & 12 \\ 14 & 6 & 0 & 9 \\ 7 & 12 & 9 & 0 \end{matrix}$$

เริ่มจาก เมืองที่ 1 ดูแถวแรก หาระยะที่น้อยที่สุด ได้คอลัมภ์ที่ 4

เริ่มจาก 4 ดูแถวที่ 4 หาระยะที่น้อยที่สุด โดยไม่รวมเมือง 1 ได้คอลัมภ์ที่ 3

เริ่มจาก 3 ดูแถวที่ 3 หาระยะที่น้อยที่สุด โดยไม่รวมเมือง 1,4 ได้คอลัมภ์ที่ 2

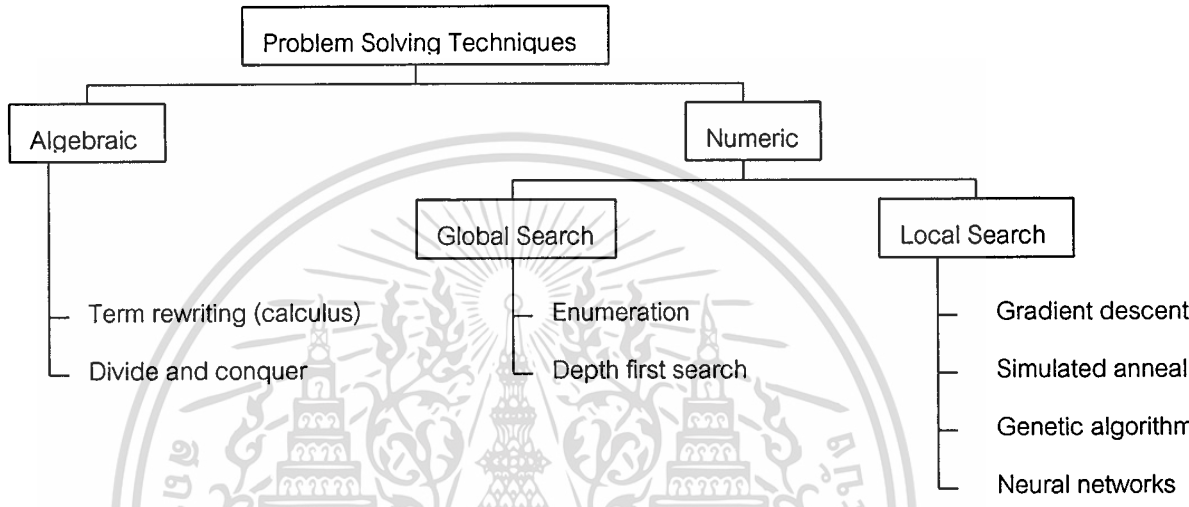
จะได้ระยะทางจาก 1,4,3,2,1 (7+9+6+10) = 32

จากนั้นทดลองเริ่มต้นที่เมืองอื่นๆจนครบทุกเมือง แล้วหาค่าที่น้อยที่สุดออกมา จะได้เป็นค่า Optimal ออกมา

บทที่ 5

การปรับค่า activations function

Optimization Techniques



ภาพที่ 5.1 Problem Solving Techniques

ในกรณี Hopfield Network TSP เราจะเน้นไปที่

Problem Solving Techniques

Numeric

Local Search

Neural Networks

โดยปกติ Neural Network จะเป็นวงจรดังนี้

Input จะถูก weights และ เปรียบเทียบกับค่า threshold

Activation จะเปลี่ยนสถานะ state ของ neurons

Output ได้มาจากการเปลี่ยน state โดยใช้ nonlinear function

การ Optimization สามารถกระทำได้โดย

Selection of Optimum set of Input nodes

Selection of Optimum numbers of Hidden nodes and links

บทที่ 6

การทดลองและผลการทดลอง

ภาพที่ 6.1 แสดงเมนูหลักของการทดลอง

เมนูหลักของโปรแกรมประกอบด้วย

Setup

กำหนดค่า parameters ต่าง

ให้ระยะทางระหว่างแต่ละเมืองและให้ค่า input vectors

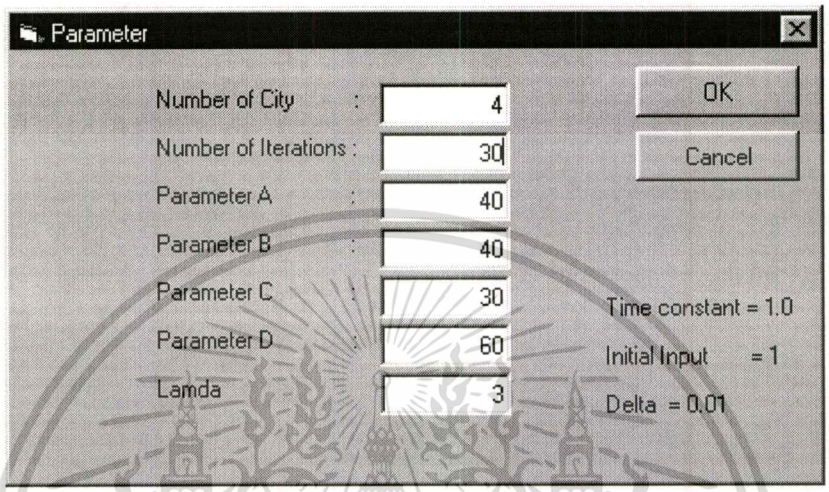
Findtour

ทำทีละ tour จะเห็นทุกๆ tour ใน set

ทำทุกๆ tour ใน set จนครบแล้วคำนวนค่า errors ออกมา

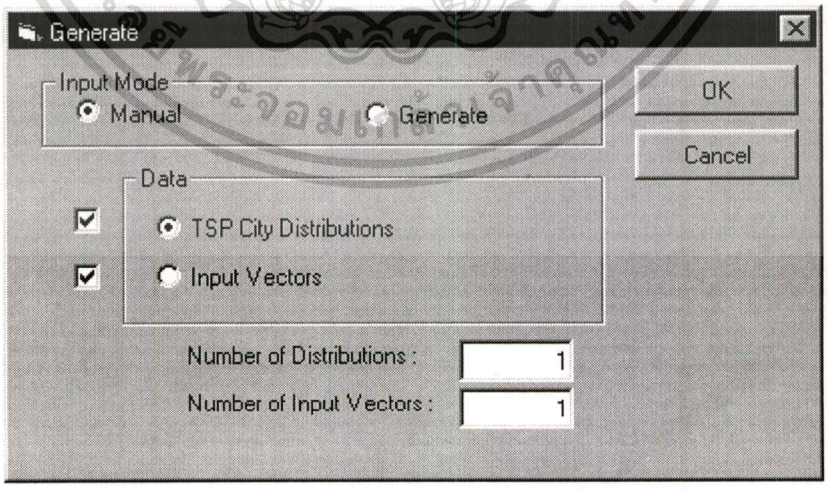
View

แสดงค่าที่ต้องการทราบของแต่ละ tour เช่น ระยะทางระหว่างแต่ละเมือง, input vectors, matrix, activations และ outputs เป็นต้น



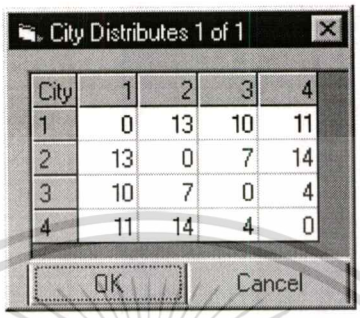
ภาพที่ 6.2 แสดงเมนูย่อยการกำหนดค่า parameters

เมนูย่อยการกำหนดค่า parameters สามารถกำหนดจำนวนเมืองที่จะใช้ในการหา tour และ จำนวนที่ใช้ในการวน loop กี่รอบ และค่าที่เหมาะสมอื่นๆอีก เป็นต้น



ภาพที่ 6.3 แสดงเมนูย่อยการกำหนดค่าระยะทางระหว่างเมืองและ input vectors

สามารถเลือกได้ว่าจะทำการป้อนตัวเลขเข้าไปเองหรือจะให้โปรแกรมคำนวนออกมา และกำหนดได้ว่าจะให้มีกี่ Distributions (จำนวนเมืองเท่ากันแต่ระยะทางจะต่างกันออกไป)



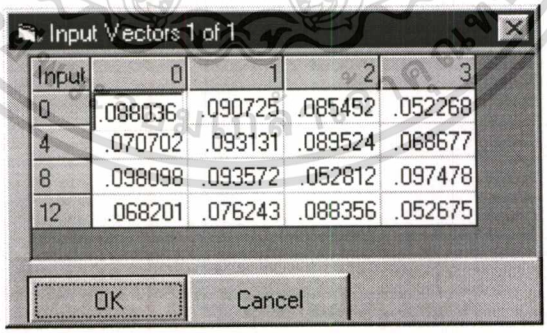
ภาพที่ 6.4 แสดงตัวอย่างระยะทางระหว่างเมือง

ตัวอย่างระยะทางระหว่างเมืองอยู่ในรูปของตารางคือ

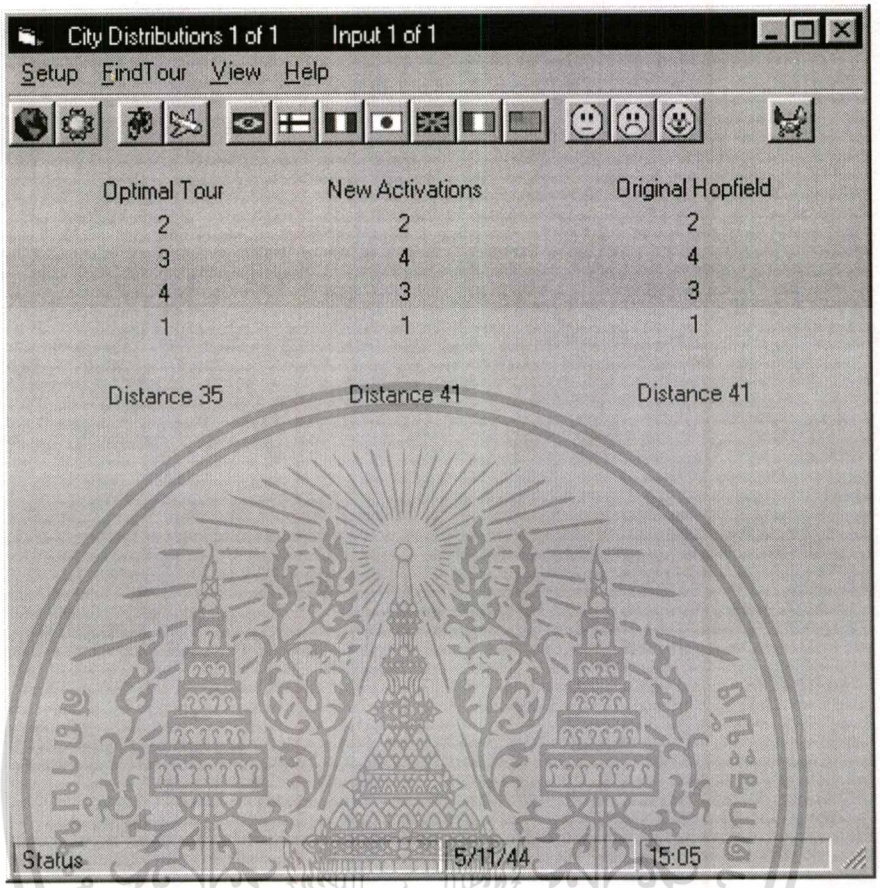
ระยะทางจากเมือง 1 ไป 2 คือ 13 ระยะทางจากเมือง 2 ไป 3 คือ 7

ระยะทางจากเมือง 1 ไป 3 คือ 10 ระยะทางจากเมือง 2 ไป 4 คือ 14

ระยะทางจากเมือง 1 ไป 4 คือ 11 ระยะทางจากเมือง 3 ไป 4 คือ 4

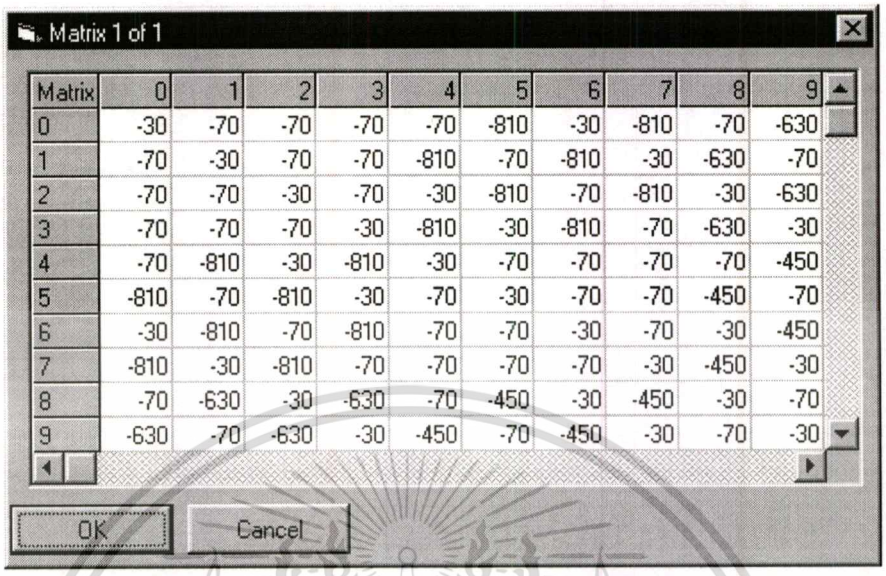


ภาพที่ 6.5 แสดง input vectors



ภาพที่ 6.6 แสดง tour ที่คำนวนได้

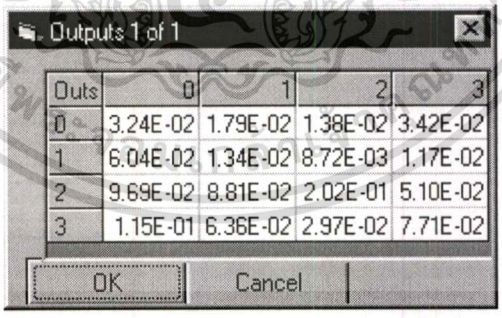
แสดง tour ที่คำนวนได้ของเมือง 4 เมือง ตามค่า parameters ที่กำหนดมา โดยแถวซ้ายสุดจะแสดงค่า optimal tour (tour ที่มีระยะทางสั้นที่สุด) ออกมาให้เปรียบเทียบด้วย



Matrix	0	1	2	3	4	5	6	7	8	9
0	-30	-70	-70	-70	-70	-810	-30	-810	-70	-630
1	-70	-30	-70	-70	-810	-70	-810	-30	-630	-70
2	-70	-70	-30	-70	-30	-810	-70	-810	-30	-630
3	-70	-70	-70	-30	-810	-30	-810	-70	-630	-30
4	-70	-810	-30	-810	-30	-70	-70	-70	-70	-450
5	-810	-70	-810	-30	-70	-30	-70	-70	-450	-70
6	-30	-810	-70	-810	-70	-70	-30	-70	-30	-450
7	-810	-30	-810	-70	-70	-70	-70	-30	-450	-30
8	-70	-630	-30	-630	-70	-450	-30	-450	-30	-70
9	-630	-70	-630	-30	-450	-70	-450	-30	-70	-30

ภาพที่ 6.7 แสดง weight matrix

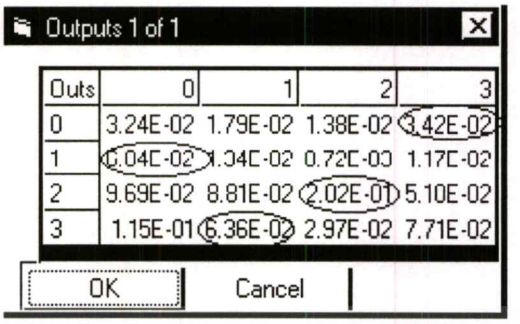
เมื่อคำนวนแต่ละ tour ออกมาแล้วเราสามารถดูค่าต่างๆที่เราสนใจได้ เช่น ค่า weight matrix ซึ่งจะใช้ในการคำนวนหา activations (สามารถ view ดูได้เช่นกัน)



Outs	0	1	2	3
0	3.24E-02	1.79E-02	1.38E-02	3.42E-02
1	6.04E-02	1.34E-02	8.72E-03	1.17E-02
2	9.69E-02	8.81E-02	2.02E-01	5.10E-02
3	1.15E-01	6.36E-02	2.97E-02	7.71E-02

ภาพที่ 6.8 แสดงค่า outputs สุดท้าย

last outputs เป็นผลลัพท์สุดท้ายที่เราต้องการ ซึ่งเราจะสามารถหา ลำดับของการ tour ได้มาจาก last outputs นี่เอง



ภาพที่ 6.9 แสดงการ findtour จาก last outputs

โดยเริ่มพิจารณาจากแถวแรกแถว 0 หาค่าที่มากที่สุด ได้แก่ 3.42E –02 เป็นคู่ลำดับ (0,3) ซึ่งหมายความว่า แถวที่ 0 จะได้รับการไป tour ในลำดับที่ 3

ต่อไปพิจารณาแถว 1 หาค่ามากที่สุดโดยไม่รวม column 3 ได้ค่า 6.04E –02 เป็นคู่ลำดับ (1,0) ซึ่งหมายความว่า แถวที่ 1 จะได้รับการไป tour ในลำดับที่ 0

ต่อไปพิจารณาแถว 2 หาค่ามากที่สุดโดยไม่รวม column 3 และ 1 ได้ค่า 2.02E -01 เป็นคู่ลำดับ (2,2) ซึ่งหมายความว่า แถวที่ 2 จะได้รับการไป tour ในลำดับที่ 2

สุดท้ายแถวที่ 3 ที่เหลือก็จะได้ค่า 6.36E –02 เป็นคู่ลำดับ (3,1) ซึ่งหมายความว่า แถวที่ 3 จะได้รับการไป tour ในลำดับที่ 1

เมื่อเรียงจากลำดับการไป tour เป็น 0, 1, 2, 3 ก็จะได้เมืองที่ไปเยือนเป็น 1, 3, 2, 0 ซึ่งนั่นก็คือจะได้ tour นี้เริ่มจากเมือง 2, 4, 3, 1 และกลับไปยังเมือง 2 ตามลำดับ

บทที่ 7

สรุปผลการศึกษา

สรุปผลการศึกษา

การทดลองโดยใช้ 5 เมือง 20 random Distributions กับ add random noises 5 ชุด

Total Errors	New Activations Errors	Old Activations Errors
1	5.4	5.95
2	5.6	6.8
3	5.6	6.6
4	7.0	6.67
5	6.5	7.62
6	5.8	6.7
7	4.9	6.44
8	5.4	7.5
9	5.9	6.63
10	5.6	7.65
	57.7	68.66

คิดเป็น 68.66*100/57.7=119%

ลด errors จากเดิมได้ 19 %

การทดลองโดยใช้ 10 เมือง 20 random Distributions กับ add random noises 5 ชุด

Total Errors	New Activations Errors	Old Activations Errors
1	20.2139	21.0925
2	21.7740	22.862
3	18.7579	19.48
4	17.92	19.80
5	19.05	20.89
6	18.31	19.66
7	20.35	20.55
8	17.51	18.85
9	17.76	19.32
10	19.70	20.64
	191.3458	203.98

ลด errors จากเดิมได้ 6.6 %

เราสามารถใช้ Hopfield Network ในการแก้ปัญหาที่เป็น Optimization Problems อย่างเช่น Traveling Salesperson Problem โดยได้ค่าที่เหมาะสมที่สุดในเวลาที่รวดเร็วกว่าการใช้วิธีอื่นๆ ทาง digital computer นอกจากนี้ จากการทดสอบเบื้องต้นพบว่า sigmoid activation function ของ Hopfield Network มีความอ่อนไหวง่ายต่อการเปลี่ยนค่าของสิ่งรบกวน noises ใน Network ดังนั้น ถ้าสามารถหา new activation function ที่สามารถทนทานต่อการเปลี่ยนแปลง noises ได้ น่าจะสามารถทำให้ประสิทธิภาพ performance ของ Hopfield Network เพิ่มขึ้นมากตามไปด้วย

ในการทดลองเบื้องต้นใช้จำนวนเมืองสำหรับ TSP เพียง 5 เมืองเท่านั้น โดยทำการสุ่มการกระจายของเมือง 60 ชุด ให้แต่ละชุดจะทำการทดลองชุดละ 10 ครั้งโดยแต่ละครั้งจะเพิ่มสิ่งรบกวน noise เข้าไปเป็น Input ต่างกัน จากผลการทดลองพบว่าการปรับเปลี่ยนค่า output function สามารถลดค่าความผิดพลาดจากเดิมได้ 19 % และถ้าเพิ่มจำนวนรอบการคำนวนมากขึ้นเป็น 150 รอบ จะมีจำนวนครั้งของ network ที่เชื่อถือได้เพิ่มขึ้นมากกว่าเดิม 2.3 %

บรรณานุกรม

Cornelius, T. Leondes 1998. **Optimization Techniques : Volume 2 of Neural Network Systems Techniques and Applications** : Academic Press.

James, A. Freeman and David, M. Skapura. 1991. **Neural Networks : Algorithms, Applications, And Programming Techniques** : Addison-Wesley.

Valluru, Rao and Hayagriva, Rao. 1995. **C++ Neural Networks & Fuzzy Logic,** Second Edition : MIS:Press.

Xinchuan, Zeng and Tont, R. Martinez. "A New Activation Function in Hopfield Network" Computer Science Department, Brigham Young University. Provo, Utah 84602.