รายงานการวิจัยฉบับสมบูรณ์

เรื่อง

การเรียนรู้รวมแบบลูกผสมสำหรับการประมาณมูลค่าซอฟต์แวร์

(Hybrid Ensemble Learning for Software Cost Estimations)

โครงการวิจัยนี้ได้รับการสนับสนุนทุนวิจัย
จาก
สำนักงานคณะกรรมการวิจัยแห่งชาติ
ปีงบประมาณ พ.ศ. ๒๕๕๕

คณะผู้วิจัย

นายกฤษณะ ชินสาร	หัวหน้าโครงงานวิจัย
นางสาวสุวรรณา รัศมีขวัญ	ผู้ร่วมวิจัย
นางสาวสุนิสา ริมเจริญ	ผู้ร่วมวิจัย
นายภูสิต กุลเกษม	ผู้ร่วมวิจัย
นางสาวเบญจภรณ์ จันทรกองกุล	ผู้ร่วมวิจัย
นางสาวสุภาวดี ศรีคำดี	ผู้ร่วมวิจัย

ศูนย์วิจัย Knowledge and Smart Technology
คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

บทคัดย่อ

การพัฒนาซอฟต์แวร์เป็นกิจกรรมหลักซึ่งมีค่าใช้จ่ายสูงมากในเกือบทุกองค์กรและมีการดำเนินการในหลากหลายบริบท งานวิจัยนี้ได้นำเสนอวิธีการประเมินมูลค่าของซอฟต์แวร์ ซึ่งหมายถึงกระบวนการที่ใช้ในการพยากรณ์ความพยายามที่ต้องใช้ในการพัฒนาซอฟต์แวร์ ได้มีงานวิจัยด้านการประเมินมูลค่าซอฟต์แวร์ซึ่งแบ่งออกเป็น 2 กลุ่มหลัก คือ กลุ่มวิธีการแบบพาราเมติก (Parametric Models) และกลุ่มวิธีการแบบแมชชีนเลินนิ่ง (Machine Learning) มาแล้วไม่น้อยกว่า 30 ปี แต่ปรากฏว่าผลที่ได้รับนั้นยังไม่มีประสิทธิภาพเท่าที่ควร ในงานวิจัยนี้ได้นำเสนอวิธีการแบบเอนเซมเบิ้ลนิวรอนเน็ตเวอร์ค (Ensemble Neural Network) ซึ่งหมายถึงการใช้การผสมผสานวิธีการที่เหมาะสมจากกลุ่มของวิธีการแมชชีนเลินนิ่ง (Machine Learning) เข้าด้วยกัน เพื่อแก้ข้อบกพร่องของวิธีการแบบพาราเมตริก (Parametric Models) ในประเด็นที่ไม่สามารถจัดการกับเงื่อนไขพิเศษ เช่น บุคลากร ทีมงาน และ การจับคู่ระหว่างระดับความชำนาญกับงาน นอกจากนั้นในวิธีการที่มีอยู่เดิม ถ้าหากมีความจำเป็นในการปรับแต่งค่าในภายหลังก็ต้องจัดการด้วยตนเอง

สำหรับขั้นตอนในการประเมินมูลค่าซอฟต์แวร์นั้นประกอบด้วยกระบวนการหลัก 3 ขั้นตอน กล่าวคือ 1. การวัดขนาดซอฟต์แวร์ (Software Size) ซึ่งโดยทั่วไปดำเนินการด้วย 2 วิธีการ คือ วัดจากจำนวนชุดคำสั่ง (Code size metrics) และ วัดจากจำนวนฟังก์ชัน (Functionality metrics) 2. การวัดระดับค่าความพยายาม (Software Effort) ซึ่งทั่วไปจะวัดอยู่ในรูปของระยะเวลาที่ต้องใช้ต่อคน เช่น วัดเป็นหน่วยของจำนวนคนต่อหน่วยของเวลา ซึ่งต้องพิจารณาประเด็นของภาษาที่ใช้ในการพัฒนา เครื่องมือที่ใช้ในการพัฒนา ปริมาณขององค์ประกอบที่ได้จากระบบเดิม เวลาที่สามารถใช้ในการทำงานได้ ผลผลิตต่อบุคคล ความยากง่ายของงาน เป็นต้น 3.การคิดค่าใช้จ่าย (Software Cost) ซึ่งเป็นขั้นตอนที่นำผลที่ได้จากขั้นตอนที่ 1 และ 2 มาคำนวณกับค่าแรงมาตรฐานตามความชำนาญเฉพาะทางของบุคลากรในทีม

ในงานวิจัยนี้ ได้นำเสนอการพยากรณ์ราคาซอฟต์แวร์ โดยได้นำเสนอกระบวนการลูกผสมนิวรอนเน็ตเวิร์ค ซึ่งเป็นการนำขั้นตอนวิธีการรู้จำแบบแพร่ย้อนกลับจำนวน 4 ตัวแบบที่มีสถาปัตยกรรมและพารามิเตอร์แตกต่างกันมาเรียนรู้เพื่อหาผลลัพธ์ของการพยากรณ์ราคาซอฟต์แวร์ของฐานข้อมูลมาตรฐานของ NASA จากนั้นรวมผลลัพธ์ที่ได้ด้วยฟังก์ชันการพยากรณ์ที่ใช้กลยุทธ์เชิงวิวัฒนาการในการช่วยหาค่าสัมประสิทธิ์ที่เหมาะสม ในการทดลองแสดงให้เห็นว่าวิธีการที่นำเสนอมีค่าความคลาดเคลื่อนเฉลี่ยสัมพัทธ์ (MMRE) ดีกว่าวิธีการของ COCOMO ทั้ง 2 ชุดข้อมูลทดสอบในภาพรวม โดยวิธีการที่นำเสนอให้ค่าความคลาดเคลื่อนที่ต่ำกว่าค่าความคลาดเคลื่อนเฉลี่ยของวิธี COCOMO ประมาณ 0.1

Abstract

Software Development is a crucial activity and a high cost one in any type of organization which has different context. Software cost estimation is a process to forecast effort needed for any software development. There are two main approaches in software cost estimation. The first one is Parametric Models and the second one is Machine Learning Models. For the last 30 years, there is a great number of researches that try to come-up with a more and more efficient method. In this research report, we have proposed "Ensemble Neural Network" as an alternative approach to estimate software cost. The proposed method will help to deal with factors on human resource skill, team work, skill matching and those automatic adjustments that Parametric Methods seem to have some difficulty.

In estimating software cost, there are three steps: 1. Measuring software size; in terms of code size metrics or functionality metrics 2. Forecasting effort need; in terms of time needed for each staff in performing their tasks and 3. Estimating software cost; in terms of amount of money. According to these steps, we proposed an alternative approach that will select more proper features in steps one and two. So that the software cost estimation in step no. 3 will be improved.

In this research, ensemble learning using neural networks is proposed. BPNNs based on 4 different structures are used for predicting software cost. The proposed networks are evaluated with standard benchmark, normally used in software cost prediction problem, released by NASA. After the prediction with 4 BPNNs, evolutionary strategies are used for improving the predicting performance. It can be seen that the performance of the proposed method is better the predicting performance obtained from the COCOMO method for all data set used in the experiments. In other word, the MMRE of the proposed methods is approximately lower than the COCOMO methods about 0.1

สารบัญ

บทที่ 1 บทนำ

1.1 ที่มาและความสำคัญของปัญหา

การพัฒนาซอฟต์แวร์เป็นกิจกรรมที่ถูกดำเนินการในหลากหลายบริบท ได้แก่ การพัฒนาซอฟต์แวร์เพื่อการปรับปรุงกระบวนการทำงานของส่วนราชการ การพัฒนาซอฟต์แวร์เพื่อการเพิ่มขีดความสามารถในการแข่งขันได้ของภาคธุรกิจ ตลอดจนการพัฒนาซอฟต์แวร์เพื่อสร้างการเรียนรู้ทั้งในระดับหน่วยงานย่อยขององค์กร ตลอดไปจนถึงการเรียนรู้ของนักเรียน นิสิต นักศึกษา และประชาชนโดยทั่วไป ซึ่งการสร้างซอฟต์แวร์สำหรับบริบทที่แตกต่างกันจำเป็นต้องอาศัยวิธีการ เครื่องมือ เทคโนโลยี ทักษะของบุคลากร และ ค่าใช้จ่ายในการพัฒนาที่แตกต่างกัน การที่โครงการพัฒนาซอฟต์แวร์ใดจะได้รับการพิจาณาให้ดำเนินการได้นั้น ประเด็นสำคัญอันดับต้นๆ คือ การประเมินความคุ้มค่าในการลงทุน ซึ่งผู้บริหารระดับสูงใช้เป็นเกณฑ์ในการพิจารณาการให้ลำดับความสำคัญของโครงการ การจะประเมินความคุ้มค่าในการลงทุนได้นั้นจำเป็นต้องมีการประเมินมูลค่าของซอฟต์แวร์ก่อนที่จะนำไปเปรียบเทียบกับมูลค่าของผลประโยชน์ที่จะได้รับ ซึ่งผลประโยชน์ที่จะได้รับนั้นอาจอยู่ในรูปของรายรับที่เพิ่มขึ้นหรือค่าใช้จ่ายที่ลดลง ในรูปของการได้ทำตามข้อบังคับหรือกฎหมาย หรือในรูปของการสร้างหรือรักษาชื่อเสียงและภาพพจน์ของหน่วยงาน

ในงานวิจัยนี้ได้นำเสนอวิธีการประเมินมูลค่าของซอฟต์แวร์ ซึ่งการประเมินมูลค่าซอฟต์แวร์หมายถึงกระบวนการที่ใช้ในการพยากรณ์ความพยายามที่ต้องใช้ในการพัฒนาซอฟต์แวร์ ได้มีงานวิจัยด้านการประเมินมูลค่าซอฟต์แวร์ด้วยวิธีการที่หลากหลายซึ่งสามารถแบ่งออกเป็น 2 กลุ่มหลัก คือ กลุ่มวิธีการแบบพาราเมติก (Parametric Models) และกลุ่มวิธีการแบบแมชชีนเลินนิ่ง (Machine Learning) มาแล้วไม่น้อยกว่า 30 ปี แต่ปรากฏว่าผลที่ได้รับนั้นยังไม่บรรลุวัตถุประสงค์ที่ต้องการ เนื่องจากค่าใช้จ่ายที่เกิดจากการประมาณการ (Estimated Cost) ยังคงมีความแตกต่างจากค่าใช้จ่ายที่เกิดขึ้นจริง (Actual Cost) ค่อนข้างมาก ในงานวิจัยนี้ได้นำเสนอวิธีการแบบเอนเซมเบิ้ลนิวรอนเน็ตเวอร์ค (Ensemble Neural Network) ซึ่งหมายถึงการใช้การผสมผสานวิธีการที่เหมาะสมจากกลุ่มของวิธีการแมชชีนเลินนิ่ง (Machine Learning) เข้าด้วยกัน เนื่องจากผู้วิจัยมองเห็นว่าเป็นวิธีการที่จะช่วยแก้ข้อบกพร่องของวิธีการแบบพาราเมตริก (Parametric Models) ในประเด็นที่ไม่สามารถจัดการกับเงื่อนไขพิเศษ เช่น บุคลากร ทีมงาน และ การจับคู่ระหว่างระดับความชำนาญกับงาน นอกจากนั้นหากต้องการปรับแต่งค่าใดๆ ในภายหลังก็คงต้องจัดการด้วยตนเอง

สำหรับขั้นตอนในการประเมินมูลค่าซอฟต์แวร์นั้นประกอบด้วยกระบวนการหลัก 3 ขั้นตอน กล่าวคือ 1. การวัดขนาดซอฟต์แวร์ (Software Size) ซึ่งโดยทั่วไปดำเนินการด้วย 2 วิธีการ คือ วัด

จากจำนวนชุดคำสั่ง (Code size metrics) และ วัดจากการให้คะแนนจากงานที่ทำ (Functionality metrics)

2. การวัดระดับค่าความพยายาม (Software Effort) ซึ่งที่นิยมทั่วไปจะวัดอยู่ในรูปของระยะเวลาที่ต้องใช้ต่อคน เช่น วัดเป็นหน่วยของจำนวนคนต่อหน่วยของเวลา ซึ่งต้องพิจารณาประเด็นของภาษาที่ใช้ในการพัฒนา เครื่องมือที่ใช้ในการพัฒนาที่สามารถสร้างองค์ประกอบสำเร็จรูปได้ (Component) ปริมาณขององค์ประกอบที่ได้จากระบบเดิม เวลาที่สามารถใช้ในการทำงานได้ผลผลิตต่อบุคคล ความยากง่ายของงาน

3. การคิดค่าใช้จ่าย (Software Cost) ซึ่งเป็นขั้นตอนที่นำผลที่ได้จากขั้นตอนที่ 1 และ 2 มาคำนวณกับค่าแรงมาตรฐานตามความชำนาญเฉพาะทางของบุคลากรในทีม

จากขั้นตอนการประมาณมูลค่าซอฟต์แวร์ที่กล่าวไว้ข้างต้น พบว่าทั้งขั้นตอนที่ 1 และขั้นตอนที่ 2 นั้น ยังไม่มีมาตรฐานสากลที่เป็นข้อตกลงร่วมกันในการวัดค่าทั้ง 2 อย่างชัดเจน ดังนั้นผู้วิจัยคาดหวังว่าการวิจัยในโครงการนี้จะนำไปสู่การเลือกวิธีการและปัจจัยที่เหมาะสมในการประมาณค่าทั้ง 2 ชุด ดังกล่าว ซึ่งจะส่งผลต่อการประมาณค่าใช้จ่ายที่ใกล้เคียงกับค่าใช้จ่ายที่จะเกิดขึ้นจริงโดยมีค่าความคลาดเคลื่อนน้อยลงจากวิธีการที่มีอยู่ในปัจจุบัน

1.2 วัตถุประสงค์ของโครงการวิจัย

1. เพื่อศึกษาวิธีการวัดขนาดซอฟต์แวร์เพื่อใช้ในการหาปัจจัยที่ควรใช้ในการวัดขนาดซอฟต์แวร์ ที่จะนำไปสู่ขั้นตอนการประเมินการวัดค่าความพยายาม

2. เพื่อศึกษาวิธีการวัดค่าความพยายามที่จะทำให้ได้ค่าความพยายามที่ใกล้เคียงกับความเป็นจริงเพื่อนำไปคำนวณค่าใช้จ่าย

3. เพื่อศึกษาการคำนวณค่าใช้จ่ายที่เกิดจากการนำค่าความพยายามมาคำนวณกับค่าแรงมาตรฐานเพื่อให้ได้ค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ที่มีค่าความคลาดเคลื่อนน้อยลงกว่าวิธีการในปัจจุบัน

4. เพื่อให้ผู้ที่สนใจสามารถนำแนวความคิดที่นำเสนอ ไปศึกษาเพื่อทำการพัฒนาหรือประยุกต์ใช้ในงานวิจัยของตนเองต่อไป

1.3 ขอบเขตของโครงการวิจัย

การวิจัยครั้งนี้มุ่งที่จะศึกษาและพัฒนาการนำเทคนิคการเรียนรู้แบบรวมมาทำการประมาณค่าของขนาดและค่าความพยายามในการผลิตซอฟต์แวร์ เพื่อนำไปคำนวณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ โดยมีขอบเขตดังนี้

1.งานวิจัยนี้จะสามารถรองรับการประมาณค่าขนาดและค่าความพยายามของซอฟต์แวร์ได้หลากหลายบริบท

2. งานวิจัยนี้สามารถรองรับการประมาณค่าขนาดและค่าความพยายามของซอฟต์แวร์ได้หลากหลายวิธีการและหลากหลายเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ขั้นตอนวิธีในการประมาณมูลค่าซอฟต์แวร์ ที่สามารถนำไปใช้เพื่อประมาณค่าใช้จ่ายซอฟต์แวร์ได้หลากหลายบริบท

2. สามารถนำขั้นตอนวิธีข้างต้นไปพัฒนาระบบซอฟต์แวร์ เพื่อใช้ในการประมาณมูลค่าซอฟต์แวร์ที่ใกล้เคียงค่าความเป็นจริง ซึ่งจะส่งผลให้หน่วยงานสามารถบริหารทุนในการพัฒนาซอฟต์ได้อย่างเหมาะสม

3. ขั้นตอนวิธีที่นำเสนอจะสามารถใช้เป็นต้นแบบในการศึกษาขั้นสูงต่อไป

1.5 ระยะเวลาทำการวิจัยและแผนการดำเนินงานตลอดโครงการวิจัย

ดำเนินการวิจัยแบบ 2 ปีต่อเนื่อง (ปีงบประมาณ 2554-2555)

แผนการดำเนินงานปีที่ 1 (ปีงบประมาณ 2554-2555)

แผนการดำเนินงานวิจัย	เดือนที่											
	1	2	3	4	5	6	7	8	9	10	11	12
➢ การจัดเตรียมเครื่องมือ อุปกรณ์และวัสดุวิจัย	→											
➢ การศึกษาวิธีการประมาณมูลค่าซอฟต์แวร์	→	→	→	→	→	→						
➢ การจัดทำรายงานความก้าวหน้าครั้งที่ 1							→					
➢ การศึกษาการเรียนรู้รวม							→	→	→	→	→	→
➢ การจัดทำรายงานความก้าวหน้าครั้งที่ 2												

แผนการดำเนินงานปีที่ 2 (ปีงบประมาณ 2555-2556)

แผนการดำเนินงานวิจัย	เดือนที่											
	1	2	3	4	5	6	7	8	9	10	11	12
➢ การศึกษาวิธีการแบบลูกผสมสำหรับการเรียนรู้รวมเพื่อใช้กับการประมาณมูลค่าซอฟต์แวร์	→	→	→	→	→	→						
➢ การจัดทำรายงานความก้าวหน้าครั้งที่ 3							→					
➢ พัฒนาเป็น Web-based Application สำหรับผู้ใช้ทั่วไป							→	→	→	→	→	→
➢ จัดส่งรายงาน												→

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ระบบโครงข่ายประสาทเทียม *(Neural Networks)*

หลายกิจกรรมในชีวิตประจำวันของเราเกี่ยวข้องกับงานปัญญาประดิษฐ์ หรือการรู้จำ ซึ่งปัญหาการรู้จำเป็นปัญหาที่มีความยากและมีความซับซ้อนมากถ้าจะพัฒนาให้เป็นระบบอัตโนมัติในเครื่องคอมพิวเตอร์ แต่ในทางตรงกันข้ามงานดังกล่าวนี้กลับสามารถดำเนินการได้โดยง่ายโดยมนุษย์ กล่าวคือ มนุษย์สามารถรู้จำหรือแยกแยะวัตถุต่างๆ ได้อย่างมากมาย ทั้งที่วัตถุดังกล่าวนั้นกำลังอยู่ในสภาวะแวดล้อมที่มีความหลากหลาย ยกตัวอย่างเช่น นายขาวจะมีความสามารถในการจำเสียงของนายดำ และไม่ว่าในดำจะโทรศัพท์มาคุยกับนายขาวจากสภาพแวดล้อมใดๆ ก็ตามนายขาวยังคงจำเสียงนายดำได้เสมอ เป็นต้น ดังนั้น จึงเป็นเหตุผลที่สำคัญที่เราต้องพัฒนาระบบการคำนวณ (Computing system) ที่สามารถเข้าใจและเลียนแบบการทำงานของมนุษย์ระบบดังกล่าวนี้ เราเรียกว่าระบบโครงข่ายประสาท (Neural Networks)

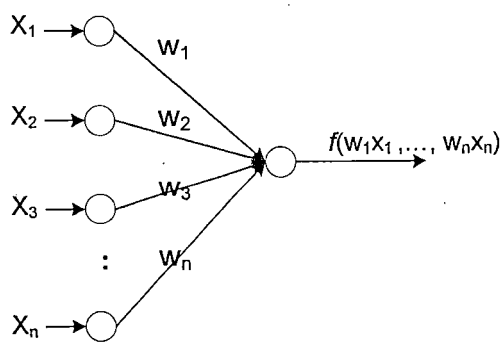
ระบบโครงข่ายประสาทเทียม (Artificial Neural Networks: ANN) หรือ บางครั้งอาจสั้นๆ เรียกว่า ระบบโครงข่ายประสาท (Neural Networks) ก็ได้ เกิดมาจากแรงบันดาลใจเกี่ยวกับความต้องการเลียนแบบการทำงานของสมองมนุษย์ด้วยเครื่องคอมพิวเตอร์ เพื่อสร้างระบบคอมพิวเตอร์แบบใหม่ที่มีความสามารถในการเรียนรู้ด้วยตนเอง ซึ่งแตกต่างจากระบบคอมพิวเตอร์ในปัจจุบัน ที่ต้องทำงานตามชุดคำสั่งที่มนุษย์เขียนสั่งไว้ล่วงหน้า (Software/Program) เท่านั้น และ ก็เป็นที่ทราบทั่วไปว่าการประมวลผลของสมองมนุษย์เรานั้นมีความซับซ้อน มีความไม่เป็นเชิงเส้น และ เป็นแบบขนานอย่างมาก นอกจากนี้ สมองมนุษย์เรานั้นยังมีความสามารถทางการคำนวณ (เช่น การรู้จำ การรับรู้ และ การควบคุมเครื่องจักร) ได้อย่างรวดเร็วกว่าเครื่องคอมพิวเตอร์ที่เรามีใช้อยู่ในปัจจุบันเราเป็นอย่างมาก ยกตัวอย่างเช่น การมองเห็นของมนุษย์ ซึ่งถือว่าเป็นการประมวลผลสารสนเทศแบบหนึ่ง ในระบบการมองเห็นของมนุษย์เรานั้น จะประกอบด้วยขั้นตอนการแทนข้อมูลสภาพแวดล้อมต่างๆ ที่ได้รับ (Data Representation) และ รวมไปถึงความสามารถในการติดต่อกับสภาพแวดล้อมต่างๆ เหล่านั้นด้วย กล่าวคือ ทันทีที่เรามองเห็น เราจะสามารถแทนข้อมูลที่มองเห็นได้แทบจะทันที และหลังจากนั้นเราจะยังสามารถจะรู้จำสภาพแวดล้อมต่างๆ นั้นได้ (การรู้จำ หมายถึง ความสามารถในการจดจำ และ บรรยายสิ่งต่างๆ ที่มองเห็นให้กับผู้อื่นได้ทราบ) กล่าวกันว่า สมองมนุษย์เรานั้นมีความสามารถในการรู้จำสิ่งต่างๆ ที่มองเห็นภายในเวลาประมาณ 100 - 200 มิลลิวินาที ซึ่งไม่มีเครื่องคอมพิวเตอร์ใดสามารถทำได้

แบบจำลองอย่างง่ายของระบบโครงข่ายประสาทเทียมจะมีความคล้ายกับโครงสร้างทางชีววิทยาทางสมองของมนุษย์อย่างมาก ดังแสดงในตารางที่ 2-1 และ รูปที่ 2-1 โดยมีข้อกำหนดเบื้องต้น ดังนี้

1. ตำแหน่งของค่าน้ำหนักบนโครงข่ายไม่มีความสัมพันธ์กัน
2. คำตอบของแต่ละโหนดจะมีเพียงค่าเดียว ซึ่งจะกระจายไปยังโหนดต่างๆ ที่มีการเชื่อมโยงถึง โดยตำแหน่งของการเชื่อมก็ไม่มีผลเช่นเดียวกัน
3. ข้อมูลที่เข้ามายังแต่ละโหนดในเวลาเดียวกันนั้นจะต้องคงสถานะเดิมไปจนกว่าการคำนวณของฟังก์ชัน $f(w_1x_1,\ldots,w_nx_n)$ จะเสร็จสิ้นลง

ตารางที่ 2-1 คำศัพท์เฉพาะเพื่อเทียบเคียง (ที่มา : K. Mehrotra et al., Element of Artificial Neural Network)

คำศัพท์เฉพาะชีววิทยาทางสมองของมนุษย์	คำศัพท์เฉพาะของโครงข่ายประสาทเทียม
Neuron	Node/Unit/Cell/Neurode
Synapse	Connection/Edge/Link
Synaptic Efficiency	Connection Strength/Weight
Firing Frequency	Node Output

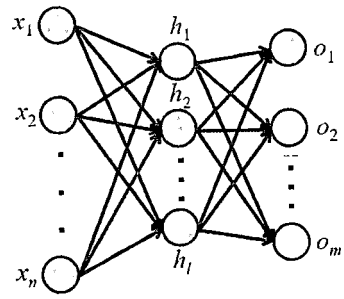


รูปที่ 2-1 ระบบโครงข่ายประสาทเทียมอย่างง่าย

2.1.1 ระบบโครงข่ายประสาทเทียมแบบวิธีการแพร่กระจายย้อนกลับ (Back propagation Algorithm)

ขั้นตอนวิธีการแพร่กระจายย้อนกลับ เป็นขั้นตอนวิธีที่ใช้ในการเรียนรู้ของเครือข่ายประสาทเทียมวิธีหนึ่งที่นิยมใช้ในโครงข่ายประสาทเทียมหลายชั้น (Multilayer neural network) เพื่อใช้ใน

การปรับค่าน้ำหนักในเส้นเชื่อมต่อระหว่างโหนดให้เหมาะสม โดยการปรับค่านี้จะขึ้นกับความแตกต่างของค่าเอาต์พุตที่คำนวณได้กับค่าเอาต์พุตที่ต้องการ พิจารณารูปต่อไปนี้ประกอบ



รูปที่ 2-2 ตัวอย่างข่ายงานประสาทเทียมแบบหลายชั้น

ตัวอย่างในรูปด้านบนแสดงข่ายงานป้อนไปหน้าแบบหลายชั้นซึ่งประกอบไปด้วยชั้นอินพุต ชั้นฮิดเดนหรือชั้นซ่อน และชั้นเอาต์พุต ในรูปแสดงชั้นฮิดเดนเพียงชั้นเดียวแต่อาจมีมากกว่าหนึ่งชั้นก็ได้ เส้นเชื่อมจะเชื่อมต่อเป็นชั้น ๆ ไม่ข้ามชั้นจากชั้นอินพุตไปชั้นฮิดเดน ถ้ามีชั้นฮิดเดนมากกว่าหนึ่งชั้นก็เชื่อมต่อกันไป และสุดท้ายจากชั้นฮิดเดนไปชั้นเอาต์พุต

ในการปรับค่าน้ำหนักโดยขั้นตอนวิธีการแพร่กระจายย้อนกลับนั้น เราต้องนิยามค่าผิดพลาดการสอนสำหรับข่ายงาน $E(\vec{w})$ จากนั้นจะหาค่าน้ำหนักที่ให้ค่าผิดพลาดต่ำสุด นิยามค่าผิดพลาดดังนี้

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_{kd} - o_{kd})^2 \qquad (1)$$

โดยที่ *Outputs* คือเซตของเอาต์พุตโหนดในข่ายงานประสาทเทียม t_{kd} และ o_{kd} เป็นค่าเอาต์พุตเป้าหมายและเอาต์พุตที่ได้จากข่ายงานประสาทเทียมตามลำดับของเอาต์พุตโหนดที่ k ของตัวอย่างที่ d ขั้นตอนการแพร่กระจายย้อนกลับจะค้นหาค่าน้ำหนักที่ให้ค่าผิดพลาดต่ำสุด

ขั้นตอนของ Back-propagation Algorithm มีดังนี้

1. กำหนดค่าอัตราเร็วในการเรียนรู้ (Learning rate parameter: r)

2. สำหรับแต่ละตัวอย่างอินพุตให้ทำตามขั้นตอนต่อไปนี้จนกว่าได้ระดับ performance ที่ต้องการ

- คำนวณหาค่าเอาต์พุตโดยใช้ค่าน้ำหนักเริ่มต้นซึ่งอาจได้จากการสุ่ม
- คำนวณหาค่า β ซึ่งแทนประโยชน์ที่จะได้รับสำหรับการเปลี่ยนค่าเอาต์พุตของแต่ละโหนด
- ในชั้นเอาต์พุต

$$\beta_z = t_z - o_z \qquad (2)$$

เมื่อ t_z = ค่าเอาต์พุตที่ต้องการ

o_z = ค่าเอาต์พุตที่คำนวณได้

- ในชั้นฮิดเดน

$$\beta_j = \sum_k w_{jk} o_k (1-o_k) \beta_k \quad (3)$$

เมื่อ w_{jk} = น้ำหนักของเส้นเชื่อมระหว่างชั้นที่ j กับ k

- คำนวณค่าน้ำหนักที่เปลี่ยนแปลงไปสำหรับในทุกน้ำหนัก ด้วยสมการต่อไปนี้

$$\Delta w_{ij} = r o_i o_j (1-o_j) \beta_j \quad (4)$$

- เพิ่มค่าน้ำหนักที่เปลี่ยนแปลง สำหรับตัวอย่างอินพุตทั้งหมด และเปลี่ยนค่าน้ำหนัก

2.2 การเรียนรู้ด้วยระบบโครงข่ายประสาทเทียมแบบผสม *(Neural Network Ensemble Learning)*

การเรียนรู้ด้วยระบบโครงข่ายประสาทเทียมแบบผสม เป็นวิธีการเรียนรู้ที่นำผลลัพธ์จากการใช้วิธีการเรียนรู้หลายๆ แบบมารวมกัน เพื่อเพิ่มความแม่นยำของผลลัพธ์ ซึ่งมักจะนำมาใช้ในการเพิ่มประสิทธิภาพของการจำแนกข้อมูล การพยากรณ์ผลลัพธ์ การประมาณค่าของฟังก์ชัน เป็นต้น การรวมผลลัพธ์ที่ได้มาจากขั้นตอนวิธีการเรียนรู้ที่มากกว่า 1 แบบทำให้เพิ่มความถูกต้องว่าผลการตัดสินใจจากขั้นตอนวิธีเหล่านั้นให้ความมั่นใจได้มากขึ้น ตัวอย่างเช่น ในปัญหาการจำแนกข้อมูล มีงานวิจัยจำนวนมากที่นำเสนอขั้นตอนวิธีในการแก้ปัญหานี้ ในการใช้อัลกอริทึมที่ต่างกัน ผลลัพธ์ที่ได้จากการเรียนรู้ในแต่ละครั้งก็มักจะให้ผลลัพธ์ที่แตกต่างกันด้วย ทำให้เกิดปัญหาตามมาว่า ถ้าในกรณีที่ตัวเรียนรู้แต่ละตัวให้ผลไม่ตรงกัน ควรจะเลือกผลลัพธ์ใดไปเป็นคำตอบ

จากปัญหาดังกล่าว แนวความคิดในการสร้างวิธีเรียนรู้แบบ Ensemble จึงเกิดขึ้นเพื่อเพิ่มประสิทธิภาพของคำตอบด้วยการรวมผลลัพธ์จากหลายๆ การเรียนรู้ ซึ่งขั้นตอนหลักของการเรียนรู้แบบ ensemble จะแบ่งเป็น 2 ขั้นตอนคือ

1. ทำการเรียนรู้โดยใช้ตัวเรียนรู้หลายๆ ตัว ซึ่งตัวเรียนรู้เหล่านี้อาจจะเป็นอัลกอริทึมเดียวกันที่ใช้ parameter ต่างกัน หรือว่าจะเป็นคนละอัลกอริทึมกันก็ได้

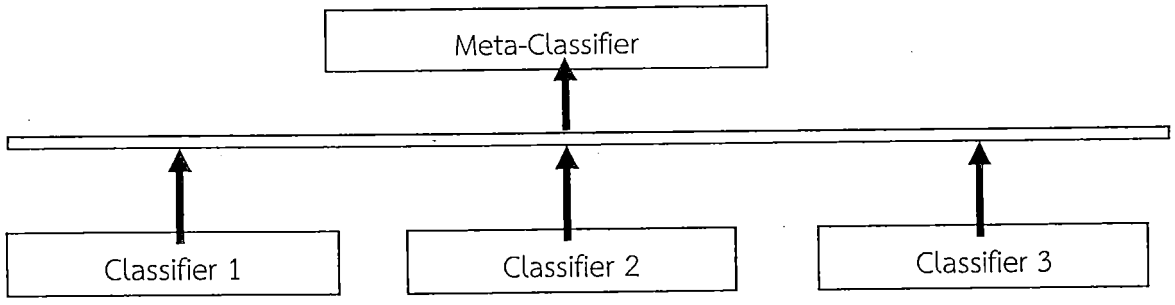
2. นำผลลัพธ์ที่ได้จากการเรียนรู้ที่มาจากหลายๆ แบบมารวมกัน แล้วทำให้จากหลายผลลัพธ์กลายเป็นคำตอบเดียวที่เหมาะสมที่สุด ซึ่งวิธีการรวมก็สามารถทำได้หลายวิธี เช่น อาจจะใช้การเฉลี่ยหรือการหาคะแนนนิยม เป็นต้น

ตัวอย่างของวิธีการเรียนรู้แบบผสม

1. Boosting (Schapire, 1990) เป็นการทำ ensemble โดยใช้ผลลัพธ์จากตัวเรียนรู้ตัวที่ให้ผลลัพธ์ยังไม่ค่อยดีเพื่อสร้างตัวเรียนรู้ที่ดีกว่า โดยการให้น้ำหนักของผลลัพธ์ที่เรียนรู้ได้ แล้วส่งให้ตัวเรียนรู้ตัวใหม่ใช้ข้อมูลนั้นในการเพิ่มประสิทธิภาพในการเรียนรู้ต่อไป โดยค่าน้ำหนักนี้จะเพิ่มขึ้นถ้าตัวเรียนรู้ไม่สามารถให้ผลลัพธ์ที่ถูกต้องได้ และค่าน้ำหนักจะลดลงถ้าตัวเรียนรู้สามารถเรียนรู้ข้อมูลสอนได้ถูกต้อง สาเหตุที่ต้องมีการกำหนดค่าน้ำหนักเช่นนี้เนื่องจากวิธีนี้ต้องการมุ่งเน้นไปที่การเรียนรู้ตัวอย่างยากที่ไม่สามารถให้ผลลัพธ์ที่ถูกต้องได้จากการเรียนรู้ในครั้งก่อนหน้า ขั้นตอนการเรียนรู้แบบนี้จะทำซ้ำหลายๆ รอบจนได้ตัวเรียนรู้ที่ให้ผลลัพธ์ที่ดีที่สุดออกมา วิธีการนี้เป็นวิธีการที่เสนอมาในยุคแรกๆ และเป็นที่นิยมใช้มาจนถึงทุกวันนี้ โดยอัลกอริทึมที่ถูกพัฒนาขึ้นมาในภายหลังและเป็นที่นิยมใช้กันในวงกว้างคือ อัลกอริทึม AdaBoost หรือ Adaptive boosting (Freund and Schapire, 1996)

2. Bagging (Breiman, 1996) วิธีการ Bagging หรือ Boostrap Aggregating จะใช้ข้อมูลสอนหลายๆ ชุดที่มาจากการสุ่มแบบใส่กลับคืน เพื่อสร้างชุดข้อมูลสอนที่แตกต่างกัน และข้อมูลสอนเหล่านี้จะถูกนำมาใช้ในการเรียนรู้ของตัวเรียนรู้แต่ละตัวแล้วจะนำผลลัพธ์ที่ได้มาจากตัวเรียนรู้แต่ละตัว ซึ่งผลลัพธ์อาจจะได้ออกมาเหมือนหรือต่างกัน มาทำการเฉลี่ยหรือ vote โดยกำหนดค่าน้ำหนักของผลลัพธ์ที่มาจากแต่ละตัวเรียนรู้ด้วยน้ำหนักที่เท่ากัน

3. Stacked generalization (Wolpert, 1992) เป็นวิธีการที่นำหลักการของ Meta Learner เข้ามาใช้ โดยการแบ่งตัวเรียนรู้ออกเป็นลำดับชั้น ตัวเรียนรู้ที่เป็น Master Model จะทำหน้าที่เป็นตัวตัดสินใจเพื่อให้คำตอบสุดท้ายออกมา ซึ่งการตัดสินใจนี้จะขึ้นอยู่กับผลลัพธ์จากตัวเรียนรู้ที่เหลือ ซึ่งลักษณะการทำงานแสดงดังรูปที่ 2-3



รูปที่ 2-3 การทำงานของการเรียนรู้แบบผสมแบบ Stacked generalization

การประมาณต้นทุนการพัฒนาซอฟต์แวร์ (Estimating Software Development Costs)

วิธีการในการประมาณมูลค่าซอฟต์แวร์แบ่งออกเป็นสองชนิดหลักได้แก่ วิธีการแบบ Non-algorithmic ซึ่งหมายถึงการวิธีการคิดที่ไม่สามารถระบุเป็นขั้นตอนวิธีได้อย่างชัดเจน และ วิธีการแบบ Algorithmic ซึ่งสามารถระบุขั้นตอนวิธีได้อย่างชัดเจน โดยสร้างเป็นตัวแบบอัลกอริทึมที่มีพื้นฐานในการใช้สูตรทางคณิตศาสตร์ บางตัวแบบใช้วิธีการคำนวณเชิงสถิติ เช่น การคำนวณหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน และบางตัวแบบมีพื้นฐานมาจากตัวแบบการถดถอย สมการอนุพันธ์ เป็นต้น ตัวแบบเหล่านี้ไม่สามารถนำมาใช้งานได้ทันทีแต่จำเป็นต้องมีการปรับค่าและประยุกต์ให้เข้ากับสถานการณ์เฉพาะที่เพื่อเพิ่มความเที่ยงตรงของตัวแบบ

Non-algorithmic Methods

วิธีนี้ใช้ข้อมูลและความรู้จากผู้เชี่ยวชาญในการประมาณมูลค่าซอฟต์แวร์ ตัวแบบการประมาณมูลค่าซอฟต์แวร์ โดยไม่มีการนำตัวแบบเชิงคณิตศาสตร์มาเป็นพื้นฐานแต่อย่างใด วิธีการประมาณค่าซอฟต์แวร์ที่จัดอยู่ในกลุ่มนี้ได้แก่

วิธี Analogy Costing วิธีนี้จำเป็นต้องมีโครงการที่สมบูรณ์ก่อนหน้านี้อย่างน้อย 1 โครงการที่มีลักษณะคล้ายคลึงกับโครงการใหม่เพื่อใช้ในการหาค่าใช้จ่ายที่เกิดขึ้นจริง การประมาณค่าใช้จ่ายโดยอาศัยความคล้ายคลึงสามารถทำได้ทั้งในระดับภาพรวมหรือส่วนใดส่วนหนึ่งของโครงการ ระดับภาพรวมมีประโยชน์ คือ องค์ประกอบทุกด้านของระบบถูกนำเข้ามาพิจารณา ขณะที่ระดับย่อยมีประโยชน์ในการให้รายละเอียดในความคล้ายคลึงและความแตกต่างระหว่างโครงการใหม่กับโครงการเก่าได้ดีกว่า จุดเด่นของวิธีนี้คือ การประมาณอยู่บนพื้นฐานของประสบการณ์ที่เคยเกิดขึ้นจริง อย่างไรก็ตาม โครงการก่อนไม่ชัดเจนพอที่จะใช้เป็นตัวแทนของเงื่อนไข สภาพแวดล้อม และการทำงานต่างๆที่จะจัดทำในโครงการใหม่ทั้งหมด

วิธี Expert Judgment วิธีนี้เกี่ยวข้องกับผู้เชี่ยวชาญอย่างน้อยหนึ่งคนขึ้นไป ผู้เชี่ยวชาญจะใช้วิธีและประสบการณ์ของตนเองในการประมาณค่า กลไกในการดำเนินการของวิธีการนี้ไม่ว่าจะเป็นวิธีการแบบ Delphi หรือ หลักการแบบ PERT จะใช้ในการแก้ปัญหาความไม่แน่นอนในการประมาณค่า

วิธีการ Delphi มีขั้นตอนโดยย่อดังนี้

1. ผู้ประสานงานส่งมอบข้อมูลทางเทคนิคและแบบฟอร์มให้แก่ผู้เชี่ยวชาญแต่ละคนเพื่อบันทึกการประมาณค่า
2. ผู้เชี่ยวชาญแต่ละคนกรอกข้อมูลลงในแบบฟอร์มโดยอิสระหากมีข้อสงสัยผู้เชี่ยวชาญสามารถถามรายละเอียดกับผู้ประสานงานได้

3. ผู้ประสานงานทำการสรุปข้อมูลการประมาณทั้งหมดที่ได้มาจากผู้เชี่ยวชาญ (ด้วยวิธีหาค่ากลาง เช่น ค่าเฉลี่ย หรือ ค่ามัธยฐาน) ลงบนแบบฟอร์มเพื่อให้ผู้เชี่ยวชาญแต่ละคนประมาณค่าอีกครั้งและให้เหตุผลประกอบ

4. ทำซ้ำขั้นตอนที่ 2-3 หลายๆครั้งตามความเหมาะสม กล่าวคือ ค่าที่ได้ค่อนข้างคงที่ หรือมีการเปลี่ยนแปลงค่าเพียงเล็กน้อย

Boehm และ Fahquhar (Boehm, 1981) เป็นผู้เสนอให้มีการปรับปรุงแก้ไขวิธี Delphi ให้มีประสิทธิภาพมากขึ้น โดยก่อนการประมาณค่า ผู้ประสานงานและผู้เชี่ยวชาญต้องเข้าร่วมประชุมเพื่ออภิปรายประเด็นต่างๆของการประมาณค่า และในขั้นตอนที่ 3 ผู้เชี่ยวชาญไม่จำเป็นต้องให้เหตุผลของการประมาณ แต่จะมีการนัดประชุมเพื่อให้ผู้เชี่ยวชาญอภิปรายถึงประเด็นต่างๆในกรณีที่มีความแตกต่างของการประมาณค่าเกิดขึ้น

วิธี Parkinson ในวิธีนี้มีแนวคิดที่ว่า “งานจะถูกขยายออกไปจนเต็มความสามารถของทรัพยากรที่มีอยู่” มูลค่าจะถูกกำหนด (ไม่ใช่เกิดจากการประมาณค่า) โดยทรัพยากรที่มีอยู่ทั้งหมด เช่น ถ้าซอฟต์แวร์ต้องส่งมอบภายใน 12 เดือน และมีบุคลากร 5 คน แต่ผลที่ได้จะถูกประมาณค่าเป็น 60 คน-เดือน ซึ่งในกรณีนี้ไม่สามารถเกิดขึ้นได้จริง อย่างไรก็ตามแม้ว่าบางครั้งการประมาณค่าจะให้ผลการประมาณค่าที่ดี แต่ไม่แนะนำให้ใช้เพราะอาจจะให้การประมาณค่าไม่สมจริง และไม่เป็นวิธีการที่ดีในทางปฏิบัติในแง่ของวิศวกรรมซอฟต์แวร์

วิธี Price-to-win มูลค่าซอฟต์แวร์ถูกประมาณค่าด้วยราคาที่ดีที่สุด การประมาณค่าขึ้นอยู่กับงบประมาณของลูกค้ามากกว่าฟังก์ชันการทำงานของซอฟต์แวร์ ตัวอย่างเช่น ถ้าการประมาณค่าที่สมเหตุสมผลสำหรับโครงการหนึ่งมีมูลค่าเท่ากับ 100 คน-เดือนแต่ลูกค้าสามารถจ่ายได้ 60 คน-เดือน ผู้ประมาณค่าจะทำการปรับค่าการประมาณค่าให้เท่ากับ 60 คน-เดือน เพื่อให้ได้รับงานนี้ ในทางปฏิบัติแล้ววิธีการนี้ไม่ใช่วิธีการที่ดีเนื่องจากอาจเป็นเหตุให้เกิดความล่าช้าในการส่งมอบหรือทำให้ทีมงานพัฒนาต้องทำงานล่วงเวลามากขึ้น

วิธี Bottom-up วิธีการนี้แต่ละองค์ประกอบของระบบซอฟต์แวร์จะถูกทำการประมาณค่าแยกกันและผลลัพธ์ของแต่ละส่วนจะถูกนำมารวมเพื่อทำการประมาณค่าระบบทั้งหมด หลักการที่สำคัญของวิธีการนี้ คือ จะต้องมีการออกแบบเบื้องต้นไว้แล้ว เพื่อเป็นตัวบ่งชี้ว่าระบบจะถูกแบ่งออกเป็นองค์ประกอบย่อยที่แตกต่างกันได้อย่างไรบ้าง

วิธี Top-down วิธีการนี้ตรงข้ามกับวิธี Bottom-up การประมาณมูลค่าทั้งหมดของระบบได้มาจากคุณสมบัติโดยรวม มูลค่ารวมสามารถแบ่งย่อยไปยังองค์ประกอบย่อยได้ วิธีการนี้เหมาะสมสำหรับการประมาณมูลค่าในระยะเริ่มต้นของโครงการ

Algorithmic methods

วิธีนี้มีพื้นฐานจากตัวแบบเชิงคณิตศาสตร์ซึ่งทำการประมาณค่าด้วยฟังก์ชั่นของตัวแปรต่างๆ ที่เป็นปัจจัยหลักของมูลค่าที่จะเกิดขึ้น ตัวแบบอัลกอริทึมจะอยู่ในรูปของ

$$Effort = f(x_1, x_2, ..., x_n)$$

โดยที่ $\{x_1, x_2, ..., x_n\}$ ใช้แทน *cost factors* วิธีการอัลกอริทึมที่มีอยู่ในปัจจุบันมีข้อแตกต่างอยู่สองจุด คือ รูปแบบของฟังก์ชั่น f และการเลือก *cost factors* ตลอดสามทศวรรษที่ผ่านมาตัวแบบการประมาณมูลค่าซอฟต์แวร์เชิงปริมาณได้มีการพัฒนาอย่างต่อเนื่อง เริ่มจากตัวแบบ Empirical ไปจนถึงตัวแบบ Analytical (Putnam, 1978; Parr, 1980, Cantone et al., 1986)โดยตัวแบบ Empirical จะใช้ข้อมูลจากโครงการก่อนหน้าเพื่อประเมินมูลค่าโครงการปัจจุบันและได้สูตรพื้นฐานมาจากการวิเคราะห์ระบบฐานข้อมูลเฉพาะทางที่มีอยู่ ตัวอย่างเช่น COCOMO ของ Boehm (Boehm, 1981) ในขณะที่ตัวแบบ Analytical นั้นจะใช้สูตรที่มีพื้นฐานมาจากสมมุติฐานรวม เช่น อัตราการแก้ปัญหาของนักพัฒนาและจำนวนปัญหาที่มีอยู่

ตัวแบบการประมาณมูลค่าซอฟต์แวร์ด้วยวิธีอัลกอริทึม

- Linear models อยู่ในรูปแบบ

$$Effort = a_0 + \sum_{i=1}^{n} a_i x_i$$

โดยที่ *a1,..., an* ถูกเลือกให้เหมาะสมกับข้อมูลของโครงการที่สมบูรณ์ งานของ Nelson ได้ใช้ตัวแบบนี้

- Multiplicative Models อยู่ในรูปแบบ

$$Effort = a_0 \prod_{i=1}^{n} a_i^{x_i}$$

โดยที่ *a1,..., an* ถูกเลือกให้เหมาะสมกับข้อมูลของโครงการที่สมบูรณ์ งานของ Walston-Felix ได้ใช้ตัวแบบนี้ และ *xi* มีค่าที่เป็นไปได้ 3 ค่าคือ -1, 0, +1 ตัวแบบ Doty จัดอยู่ในตัวแบบนี้เช่นเดียวกันโดย *xi* มีค่า 2 ค่าคือ 0 และ +1

- Power function models อยู่ในรูปแบบ

$$Effort = a \times S^b$$

โดยที่ S เป็น code-size และ a, b เป็นฟังก์ชั่นของ cost factor อื่น ตัวแบบชนิดนี้ที่มีชื่อเสียงและเป็นที่นิยมได้แก่ COCOMO และ Putnam & SLIM

- Model Calibration โดยการใช้ linear regression ตัวแบบที่กล่าวมาข้างต้นทั้งหมดไม่ได้พิจารณาถึงสถานการณ์ท้องถิ่น อย่างไรก็ตามสามารถปรับ cost factors โดยใช้ข้อมูลท้องถิ่นและวิธีการถดถอยเชิงเส้น
- Discrete Models อยู่ในรูปแบบของตารางซึ่งโดยปกติจะเกี่ยวข้องกับ ความพยายาม (Effort) ระยะเวลา ความยาก และ cost factors อื่นๆ ตัวแบบในกลุ่มนี้ได้แก่ตัวแบบ Aron (Aron, 1969) ตัวแบบ Wolverton (Wolverton, 1974) และ ตัวแบบ Boeing (Black et al., 1977) ตัวแบบเหล่านี้ได้รับความนิยมในยุคแรกๆเนื่องจากใช้งานง่าย
- Price-S เป็นตัวแบบที่ประมาณมูลค่าซอฟต์แวร์กรรมสิทธิ์พัฒนาและบำรุงรักษาโดย RCA, New Jersey (Park, 1988) เริ่มจากการประมาณขนาด ชนิดและความยากของโครงการ ตัวแบบคำนวณมูลค่าโครงการและตารางการทำงาน
- SoftCost ใช้ขนาด ความพยายาม และระยะเวลาเพื่อระบุความเสี่ยงโดยใช้รูปแบบของการกระจายความน่าจะเป็นของ Rayleigh (Tausworthe, 1981) ตัวแบบนี้มี Heuristics เพื่อใช้เป็นแนวทางให้กับผู้ประมาณในการจัดการกับเทคโนโลยีใหม่ๆและความสัมพันธ์ที่ซับซ้อนระหว่างพารามิเตอร์ต่างๆที่เกี่ยวข้อง

ตารางที่ 2-2 การแบ่งประเภทของตัวแบบอัลกอริทึม

ตัวแบบอัลกอริทึม					
	Linear	Multiplicative	Power function	Discrete	Others
Empirical	Nelson	Walston-Felix Herd et al.	COCOMO(s)	Aron Boeing Wolverton	Price-S
Analytical			Putnam		SoftCost

นอกจากขนาดของซอฟต์แวร์ยังมี Cost Factors ประเภทอื่นๆ ซึ่งกลุ่มของ Cost Factors ที่ใช้กันอย่างกว้างขวางและถูกเสนอและใช้โดย Boehm et al. ในตัวแบบ COCOMO II (Boehm et al. 1996) cost factors เหล่านี้แบ่งออกเป็น 4 ชนิด คือ ปัจจัยทางด้านผลิตภัณฑ์ (Product factors) ปัจจัยทางด้านคอมพิวเตอร์ (Computer factors) ปัจจัยส่วนตัว (Personal factors) และปัจจัยทางด้านโครงการ (Project factors)

2.3 กลยุทธ์เชิงวิวัฒนาการ (*Evolutionary Strategy: ES)*

กลยุทธ์เชิงวิวัฒนาการถูกพัฒนาโดย อิงโก เรเชนเบิร์ค (Ingo Rechenberg) และ ฮานส์-พอลล์ ชเวอเฟล (Hans-Paul Schwefel) ในช่วง 1970s ซึ่งนำหลักการของการวิวัฒนาการทางธรรมชาติมาใช้ กล่าวคือ กระบวนการคัดเลือกตามธรรมชาติ และกฎการอยู่รอดของผู้ที่เหมาะสมที่สุด โดยในการทำซ้ำแต่ละครั้งจะมีการคัดเลือกเพื่อตัดคำตอบที่อ่อนแอกว่าออกไป และคำตอบที่เหลือที่มีค่าความเหมาะสม (Fitness Value) สูงกว่าจะถูกนำไปทำเป็นต้นแบบในการเปลี่ยนแปลงค่าของคำตอบต่างๆเล็กน้อย (Mutate) ในรูปแบบอื่น ๆ จากนั้นทำการคัดสรรคำตอบที่มีค่าความเหมาะสมที่สูงกว่าไปเป็นต้นแบบในรอบถัดไป ตัวแปรต่าง ๆ ในกลยุทธ์เชิงวิวัฒนาการจะถูกแทนในรูปแบบจำนวนจริงที่มีความยาวคงที่ (fixed-length real-valued vector) แต่ละตำแหน่งในเวกเตอร์จะสอดคล้องกับลักษณะของแต่ละตัวแปร โดยในการใช้กลยุทธ์เชิงวิวัฒนาการเพื่อหาค่าเหมาะสม ตัวอย่างเช่น การหาค่าสูงสุด ใช้สมการที่ (5)

$$f^* = f(\vec{x}^*) = \max\left\{f(\vec{x})\middle|\vec{x}\in M\subseteq \mathrm{R}^n\right\} \quad (5)$$

เมื่อ $f(\vec{x})$ คือ ฟังก์ชันค่าเหมาะสมที่สุด

$\vec{x}$ คือ เวคเตอร์ของจำนวนจริง n มิติ

$\vec{x}^*$ คือ เวคเตอร์ที่ทำให้ $f(\vec{x})$ มีค่าสูงที่สุด

M คือ เซตของคำตอบที่เป็นไปได้

กลยุทธ์เชิงวิวัฒนาการอย่างง่ายเริ่มต้นด้วยการมีประชากร 1 ตัว ที่สร้างประชากรใหม่ 1 ตัว เรียกว่า (1+1)-ES โดยขั้นตอนการทำงานของกลยุทธ์เชิงวิวัฒนาการแบบ (1+1)-ES เริ่มต้นจากการสุ่มค่าเวคเตอร์จำนวนจริงและค่าส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation) พร้อมทั้งหาค่าความเหมาะสมของประชากรเริ่มต้น แล้วสร้างประชากรใหม่ 1 ตัวด้วยการกลายพันธุ์โดยอาศัยการสุ่มค่าจำนวนจริงขึ้นมาจากการกระจายแบบปกติ (Normal Distribution) ที่มีค่าเฉลี่ย (Mean) เท่ากับ 0 และค่าส่วนเบี่ยงเบนมาตรฐานเท่ากับ σ จากนั้นทำการประเมินค่าความเหมาะสมของประชากรที่สร้างขึ้นใหม่โดยใช้ฟังก์ชันหาค่าความเหมาะสม แล้วเลือกประชากรตัวที่เหมาะสมกว่าเป็นประชากรในรุ่นต่อไป รหัสเทียมของ (1+1)-ES แสดงในรูปที่ 2-4

```
Procedure (1+1)-ES
begin
        generation = 0
        Initialize (x⃗, σ)
        Evaluate (x⃗)
        while ( termination criterion not fulfilled ) do
                (x⃗', σ') = mutate (x⃗, σ)
                Evaluate (x⃗')
                if (x⃗' ≤ x⃗) then
                        (x⃗, σ) = (x⃗', σ')
                generation = generation + 1
        end while
end
```

รูปที่ 2-4 รหัสเทียมของกลยุทธ์เชิงวิวัฒนาการแบบ (1+1)-ES

รายละเอียดของกลยุทธ์เชิงวิวัฒนาการดังรูปที่ 2-4 ประกอบด้วย การสร้างประชากรเริ่มต้น การประเมินค่าคำตอบ การสร้างประชากรรุ่นใหม่ด้วยการกลายพันธุ์ ซึ่งมีรายละเอียดดังนี้

2.3.1 การสร้างประชากรเริ่มต้น

เริ่มต้นการทำงานของกลยุทธ์เชิงวิวัฒนาการ จะสร้างประชากรเริ่มต้นโดยการสุ่มค่าจำนวนจริงขึ้นมา n ค่า (เป็นเวคเตอร์ของจำนวนจริง n มิติ) และสุ่มค่าส่วนเบี่ยงเบนมาตรฐานขึ้นมา 1 ค่า เช่น การกำหนดค่าเริ่มต้นสำหรับจำนวนจริง 4 ค่า แสดงดังตัวอย่างในรูปที่ 2-5

1.24	2.18	0.95	1.56	0.50
x_1	x_2	x_3	x_4	σ

รูปที่ 2-5 ตัวอย่างการกำหนดค่าเริ่มต้นสำหรับกลยุทธ์เชิงวิวัฒนาการ

2.3.2 การประเมินค่าคำตอบ

การประเมินค่าคำตอบสำหรับกลยุทธ์เชิงวิวัฒนาการนั้นทำได้โดยใช้ฟังก์ชันหาค่าความเหมาะสม เช่น จากตัวอย่างการกำหนดค่าเริ่มต้นโครโมโซมดังรูปที่ 2-5 ถ้าฟังก์ชันหาค่าความเหมาะสมคือ $f(x) = x_1 + x_2 + x_3 + x_4$ เมื่อนำโครโมโซมมาหาค่าความเหมาะสม จะได้ค่าความเหมาะสมเท่ากับ 5.93 ซึ่งคำนวณได้จาก (1.24+ 2.18 +0.95 + 1.56) เป็นต้น

2.3.3 การสร้างประชากรรุ่นใหม่ด้วยการกลายพันธุ์

การสร้างประชากรใหม่ของกลยุทธ์เชิงวิวัฒนาการ ก่อนอื่นจะทำการปรับค่าส่วนเบี่ยงเบนมาตรฐานก่อน โดย (Rechenberg, 1973) ได้เสนอกฎความสำเร็จ 1/5 (1/5 success rule) สำหรับการปรับค่าส่วนเบี่ยงเบนมาตรฐานไว้ดังสมการที่ (6)

$$\sigma = \begin{cases} \sigma \div 0.817 & if\,(p > 1/5) \\ \sigma \times 0.817 & if\,(p < 1/5) \\ \sigma & if\,(p = 1/5) \end{cases} \qquad (6)$$

เมื่อ σ คือ ค่าส่วนเบี่ยงเบนมาตรฐาน

p คือ อัตราส่วนที่กลายพันธุ์แล้วนำไปสู่คำตอบที่ดีขึ้นในการทำงานที่ผ่านมา

การกลายพันธุ์สำหรับจำนวนจริงแต่ละค่าในเวคเตอร์จำนวนจริง n มิติ ทำโดยการสุ่มค่ามาจากการกระจายปกติ นั่นคือ Zi ~ Ni(0, σ) แล้วจะนำค่าที่สุ่มได้จากการกระจายปกตินี้บวกกับค่าจำนวนจริงเดิมทุกตัวในเวคเตอร์ ทำให้ได้ประชากรรุ่นใหม่ เช่น ประชากรเดิมดังรูปที่ 2-5 เมื่อทำการกลายพันธุ์ตามวิธีดังกล่าวจะได้ประชากรใหม่ดังรูปที่ 2-6

	x_1	x_2	x_3	x_4
ประชากรเดิม	1.24	2.18	0.95	1.56
$Z_i \sim N(0, \sigma)$	0.11	-0.23	0.07	-0.44
ประชากรใหม่	1.35	1.95	1.02	1.12

รูปที่ 2-6 ตัวอย่างการกลายพันธุ์ในกลยุทธ์เชิงวิวัฒนาการ

2.4 การทบทวนวรรณกรรม/สารสนเทศ (*Information*) ที่เกี่ยวข้อง

B. Tirimula Rao (Rao, 2009) และคณะ นำเสนองานวิจัยเรื่อง A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN) โดยในงานวิจัยนี้นำเสนอระบบโครงข่ายประสาทเทียมสำหรับการพยากรณ์มูลค่าของซอฟต์แวร์ซึ่งช่วยลดความซับซ้อนในการคำนวณและสามารถพยากรณ์มูลค่าซอฟต์แวร์แบบออนไลน์ได้ สถาปัตยกรรมของโครงข่ายประสาทเทียมที่นำเสนอในงานวิจัยนี้เป็นแบบที่ไม่มีชั้นซ่อน (Hidden Layer) กระบวนการสอนจะไม่ได้ใช้แบบแพร่ย้อนกลับแบบสมบูรณ์ คำตอบที่ใช้ในการเปรียบเทียบของการสอนจะใช้ค่าที่ได้จากการประมาณมูลค่าซอฟต์แวร์ด้วยวิธี COCOMO (Constructive Cost Model) จากผลการทดลองพบว่า วิธีการที่นำเสนอให้ผลการทดลองที่มีประสิทธิภาพสูงกว่าวิธีการโครงข่ายประสาทเทียมทั่วไป

N. Tadayon (Tadayon, 2005) นำเสนองานวิจัยเรื่อง Neural Network Approach for Software Cost Estimation โดยใช้ระบบโครงข่ายประสาทเทียมแบบเปอร์เซปตรอนหลายชั้น (Multi Layer Perceptron) ซึ่งมีชั้นซ่อนได้มากกว่าหนึ่งชั้น โดยใช้การคำตอบของการสอนที่ได้จากวิธี COCOMO การปรับค่าน้ำหนักของโครงข่ายประสาทเทียมจะใช้วิธี Gradient Descent Learning Rule

K. Vinay Kumar และคณะ (Kumar, 2008) นำเสนองานวิจัยเรื่อง Software Development Cost Estimation using Wavelet Neural Networks งานวิจัยนี้นำเสนอการประยุกต์ใช้ Wavelet Neural Networks สำหรับการพยากรณ์มูลค่าซอฟต์แวร์ โดย K. Vinay Kumar แบ่งการทำงานออกเป็นสองขั้นตอนใหญ่ๆ คือ 1.) การใช้ฟังก์ชัน Morlet และ ฟังก์ชันเกาส์เป็นฟังก์ชันสำหรับการทราสเฟอร์และ 2.) นำเสนอขั้นตอนวิธีสำหรับการหาค่าThreshold ที่สามารถยอมรับได้สำหรับการสอนของ Wavelet Neural Networks (TAWNN) จากผลการศึกษาเปรียบเทียบผลระหว่าง WNN กับวิธี MLP, RBF, MLR, DENFIS และ SVM พบว่า WNN ที่นำเสนอให้ผลการทดลองที่ดีกว่าบางวิธีการที่กล่าวไว้ข้างต้น

Y. Kultur และ คณะ (Kultur, 2009) นำเสนองานวิจัยเรื่อง Ensemble of Neural Networks with Associative Memory (ENNA) for Estimating Software Development Costs ในงานวิจัยนี้นำเสนอวิธีการเรียนรู้แบบผสมโดยใช้เทคนิค Associative Memory สำหรับการประมาณมูลค่าซอฟต์แวร์ ผลจากการนำเสนอในงานวิจัยนี้ทำให้ได้วิธีการที่สามารถเพิ่มประสิทธิภาพในการปรับเทียบค่าอย่างอัตโนมัติแทนการจัดการด้วยมือ นอกจากนี้ วิธีการที่นำเสนอยังให้ค่าความถูกต้องที่ดีกว่าวิธีโครงข่ายประสาทเทียมแบบมาตรฐาน

บทที่ 3 วิธีดำเนินการวิจัย

ในงานวิจัยนี้ได้นำเสนอวิธีการประเมินมูลค่าของซอฟต์แวร์ที่ใช้กลุ่มวิธีการแบบแมชชีนเลินนิ่ง (Machine Learning) แบบลูกผสมนิวรอนเน็ตเวอร์ค (Ensemble Neural Network) ซึ่งหมายถึงการใช้การผสมผสานวิธีการที่เหมาะสมจากกลุ่มของวิธีการแมชชีนเลินนิ่ง (Machine Learning) เข้าด้วยกัน

สำหรับขั้นตอนในการประเมินมูลค่าซอฟต์แวร์นั้นประกอบด้วยกระบวนการหลัก 3 ขั้นตอน กล่าวคือ

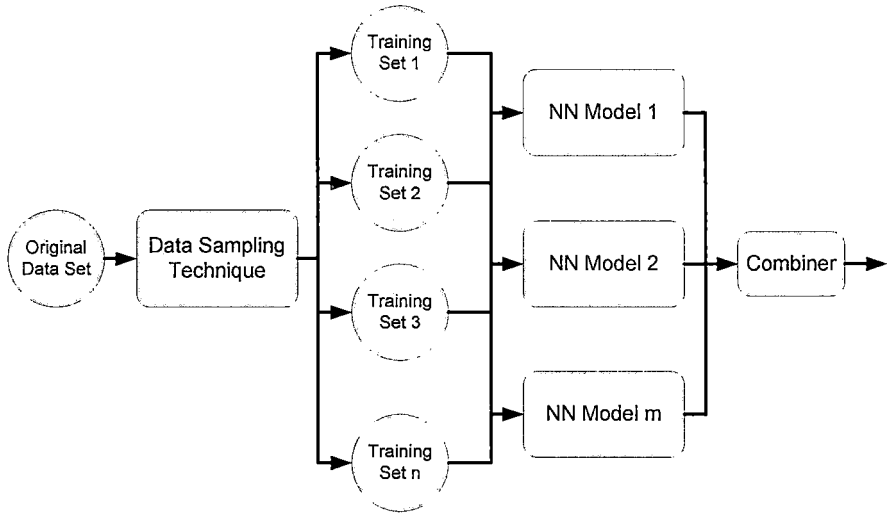
1. การวัดขนาดซอฟต์แวร์ (Software Size) ซึ่งโดยทั่วไปดำเนินการด้วย 2 วิธีการ คือ วัดจากจำนวนชุดคำสั่ง (Code size metrics) และ วัดจากการให้คะแนนจากงานที่ทำ (Functionality metrics)

2. การวัดระดับค่าความพยายาม (Software Effort) ซึ่งที่นิยมทั่วไปจะวัดอยู่ในรูปของระยะเวลาที่ต้องใช้ต่อคน เช่น วัดเป็นหน่วยของจำนวนคนต่อหน่วยของเวลา ซึ่งต้องพิจารณาประเด็นของภาษาที่ใช้ในการพัฒนา เครื่องมือที่ใช้ในการพัฒนาที่สามารถสร้างองค์ประกอบสำเร็จรูปได้ (Component) ปริมาณขององค์ประกอบที่ได้จากระบบเดิม เวลาที่สามารถใช้ในการทำงานได้ ผลผลิตต่อบุคคล ความยากง่ายของงาน

3. การคิดค่าใช้จ่าย (Software Cost) ซึ่งเป็นขั้นตอนที่นำผลที่ได้จากขั้นตอนที่ 1 และ 2 มาคำนวณกับค่าแรงมาตรฐานตามความชำนาญเฉพาะทางของบุคลากรในทีม

จากขั้นตอนการประมาณมูลค่าซอฟต์แวร์ที่กล่าวไว้ข้างต้น พบว่าทั้งขั้นตอนที่ 1 และขั้นตอนที่ 2 นั้น ยังไม่มีมาตรฐานสากลที่เป็นข้อตกลงร่วมกันในการวัดค่าทั้ง 2 อย่างชัดเจน ในงานวิจัยนี้จึงนำเสนอวิธีการเรียนรู้แบบลูกผสมเพื่อการประมาณมูลค่าซอฟต์แวร์ โดยโครงข่ายประสาทเทียมสำหรับการเรียนรู้ในครั้งนี้จะเลือกใช้ Multi Layer Perceptron ที่มีสถาปัตยกรรมแตกต่างกัน ใช้การปรับค่าน้ำหนักแบบแพร่กระจายย้อนกลับ และผลลัพธ์จากตัวเรียนรู้เหล่านี้จะถูกนำมารวมกัน เพื่อให้ได้คำตอบที่ประเมินมูลค่าของซอฟต์แวร์ได้ใกล้เคียงที่สุด โดยขั้นตอนวิธีที่ใช้ตัดสินใจในขั้นสุดท้ายจะเป็นการนำผลลัพธ์ที่ได้จากการเรียนรู้ของทุกตัวเรียนรู้มารวมกันเป็นคำตอบเดียว โดยอาศัยการคำนวณจากฟังก์ชันการประมาณมูลค่าซอฟต์แวร์ที่ได้จากการเรียนรู้ของกลยุทธ์เชิงวิวัฒนาการ (Evolutionary Strategy: ES)

ผู้วิจัยคาดหวังว่าการวิจัยในโครงการนี้จะนำไปสู่การเลือกวิธีการและปัจจัยที่เหมาะสมในการประเมินค่าทั้ง 2 ชุด ดังกล่าว ซึ่งจะส่งผลต่อการคิดค่าใช้จ่ายที่ใกล้เคียงกับค่าใช้จ่ายที่จะเกิดขึ้นได้จริงโดยมีค่าความคลาดเคลื่อนน้อยลง โดยการเรียนรู้แบบผสมที่งานวิจัยนี้นำเสนอแสดงดังรูป 3-1



รูปที่ 3-1 การเรียนรู้แบบผสมที่นำเสนอ

รูปที่ 3-1 แสดงภาพรวมของขั้นตอนวิธีที่นำเสนอ ซึ่งมีหลักการคือ ทำการเลือกข้อมูลจากชุดข้อมูลที่มีอยู่ โดยแบ่งชุดข้อมูลออกเป็น *n* ชุด แล้วนำข้อมูลที่เลือกมาได้เข้าสู่กระบวนการเรียนรู้ซึ่งจะใช้ Multi Layer Perceptron ที่มีสถาปัตยกรรมแตกต่างกันจำนวน *m* ตัวแบบ และใช้การปรับค่าน้ำหนักแบบแพร่กระจายย้อนกลับ เพื่อทำการเรียนรู้ ผลจากการเรียนรู้จากแต่ละโมเดลจะถูกนำเข้าสู่กระบวนการรวมคำตอบ ซึ่งจะใช้กลยุทธ์เชิงวิวัฒนาการ (ES) เป็นขั้นตอนวิธีในการปรับค่าของผลลัพธ์ที่ได้มาจากหลายตัวเรียนรู้และทำการตัดสินใจในขั้นสุดท้ายก่อนที่จะให้ผลลัพธ์ที่ดีที่สุดออกไป กระบวนการทั้งหมดในข้างต้น มีรายละเอียดดังต่อไปนี้

3.1 การออกแบบสถาปัตยกรรมของโครงข่ายประสาทเทียม

สถาปัตยกรรมของโครงข่ายประสาทเทียมที่งานวิจัยนี้เลือกมาเป็นต้นแบบในการเรียนรู้แบบลูกผสม (Ensemble) มี 4 ตัวแบบ โดยแบ่งออกเป็น 2 กลุ่มย่อย คือ โครงข่ายประสาทเทียมที่ประกอบด้วย 1 ชั้นซ่อน จำนวน 2 ตัวแบบ และ โครงข่ายประสาทเทียมที่ประกอบด้วย 2 ชั้นซ่อน จำนวน 2 ตัวแบบ

ในการเลือกสถาปัตยกรรมที่เหมาะสมสำหรับการนำมารวมกันในขั้นตอนลูกผสม งานวิจัยนี้ได้ทำการศึกษาผลการเปลี่ยนแปลงของค่าพารามิเตอร์ต่างๆ ที่มีผลต่อการเรียนรู้ของโครงข่ายประสาทเทียมทั้ง 4 เพื่อให้ได้โครงสร้างที่เหมาะสมสำหรับการประมาณมูลค่าซอฟต์แวร์ โดยมีขั้นตอนดังต่อไปนี้

1. ศึกษาผลการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อน

- กรณีโครงสร้างที่มี 1 ชั้นซ่อน จะทำการเปลี่ยนแปลงจำนวนโหนดตั้งแต่ 10 โหนดไปจนกระทั่งจำนวนโหนดในชั้นแฝงมีค่าเท่ากับ 2 เท่าของจำนวนโหนดในชั้นอินพุต ซึ่งในที่นี้คือ 32 โหนด โดยพิจารณาจากค่า MSE ที่ต่ำที่สุด 2 อันดับแรก มาเป็นตัวแทนโครงสร้างที่จะนำผลลัพธ์ไปผสมกับโครงสร้างอื่นๆ ต่อไป ค่า MSE มีสมการดังนี้

$$\text{MSE} = \frac{\sum e^2}{n} \qquad (7)$$

โดย e แทนผลต่างระหว่างค่าจริง กับ ค่าที่ได้จากการประมาณมูลค่าซอฟต์แวร์

n แทน จำนวนข้อมูลที่ใช้ทดสอบ

- กรณีโครงสร้างที่มี 2 ชั้นซ่อน ในชั้นซ่อนที่ 1 จะเริ่มเปลี่ยนแปลงจำนวนโหนดจาก 5 โหนด และเพิ่มขึ้นที่ละ 5 ไปจนถึง 30 โหนด และแต่ละจำนวนโหนดในชั้นซ่อนที่ 1 ทำการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อนที่ 2 ในลักษณะเดียวกัน รายละเอียดแสดงดังตารางที่ 3-1 สำหรับโครงสร้างที่เลือกใช้ พิจารณาจากค่า MSE ที่ต่ำที่สุดเช่นเดียวกัน

ตารางที่ 3-1 การศึกษาการเปลี่ยนแปลงจำนวนโหนดของโครงสร้างแบบ 2 ชั้นซ่อน

ชั้นซ่อนที่ 1	ชั้นซ่อนที่ 2
5	5, 10, 15, 20, 25, 30
10	5, 10, 15, 20, 25, 30
15	5, 10, 15, 20, 25, 30
20	5, 10, 15, 20, 25, 30
25	5, 10, 15, 20, 25, 30
30	5, 10, 15, 20, 25, 30

2. ทำการศึกษาผลการเปลี่ยนแปลงอัตราการเรียนรู้ตั้งแต่ 0.1 – 0.9 ของทั้ง 4 ตัวแบบ โดยพิจารณาจากค่า MSE ของโครงข่ายประสาทเทียมที่ผ่านการศึกษาจำนวนโหนดในขั้นตอนที่ 1

3. ทำการศึกษาผลการเปลี่ยนแปลงค่าโมเมนตัมตั้งแต่ 0.1 – 0.9 สำหรับโครงสร้างที่มี 1 ชั้นซ่อน โดยพิจารณาจากค่า MSE โดยนำผลลัพธ์ของจำนวนโหนดในชั้นซ่อนที่ได้จากการศึกษาในขั้นตอนที่ 1 และ ค่าอัตราการเรียนรู้จากการศึกษาในขั้นตอนที่ 2 มาประยุกต์ใช้กับโครงข่ายประสาทเทียมเพื่อหาค่าโมเมนตัมที่ให้ค่า MSE ต่ำสุด

เมื่อได้โครงข่ายประสาทเทียมที่เหมาะสมแล้ว ขั้นตอนถัดไปคือ นำโครงข่ายประสาทเทียมทั้ง 4 รูปแบบไปทำการเรียนรู้กับชุดข้อมูลที่ได้แบ่งไว้ และใช้การปรับค่าน้ำหนักแบบแพร่กระจายย้อนกลับ เพื่อให้ได้ผลลัพธ์ของการประมาณมูลค่าซอฟต์แวร์ด้วยค่าความผิดพลาดที่ต่ำ

3.2 การเรียนรู้ของโครงข่ายประสาทเทียม

จากโครงข่ายประสาทเทียมทั้ง 4 โครงข่ายที่ได้ทำการศึกษาการปรับเปลี่ยนพารามิเตอร์ต่างๆจนได้แบบจำลองที่เหมาะสมจากขั้นตอนที่ 1 ในขั้นตอนนี้จะเป็นการนำแต่ละตัวแบบของโครงข่ายประสาทเทียมไปทำการเรียนรู้กับชุดข้อมูลตามกระบวนการที่ได้อธิบายไว้ในหัวข้อที่ 2.1

หลังจากที่โครงข่ายประสาทเทียมได้ทำการเรียนรู้ จนได้ค่าน้ำหนักที่ให้ค่าความผิดพลาดในเกณฑ์ที่งานวิจัยนี้กำหนดแล้ว นั่นคือ ค่าความผิดพลาดของการประมาณมูลค่าของซอฟต์แวร์ที่ได้โครงข่ายประสาทเทียมต่ำกว่า 0.01 หรือ จำนวนรอบในการเรียนรู้เพื่อปรับปรุงค่าน้ำหนัก 5000 รอบ จะทำการเก็บผลลัพธ์ของการประมาณมูลค่าซอฟต์แวร์ของแต่ละตัวแบบไว้เพื่อนำเข้าสู่กระบวนการรวมคำตอบด้วยขั้นตอนวิธีกลยุทธ์เชิงวิวัฒนาการ (ES) ต่อไป

3.3 กระบวนการลูกผสมนิวรอนเน็ตเวิร์ค *(Ensemble Neural Network)*

เนื่องจากขั้นตอนวิธีที่นำเสนอเป็นขั้นตอนวิธีลูกผสมของโครงข่ายประสาทเทียมจำนวน 4 ตัวแบบ ดังนั้น ในขั้นตอนของการนำคำตอบมารวมกัน จะเป็นไปในลักษณะของการนำผลลัพธ์ที่ได้จากแต่ละตัวแบบไปคำนวณค่าในฟังก์ชัน โดยรูปแบบของฟังก์ชันแสดงดังสมการที่ (8)

$$f(x) = ax_1 + bx_2 + cx_3 + dx_4 \qquad (8)$$

โดยที่ x_1, x_2, x_3, x_4 คือ ผลลัพธ์ของการประมาณมูลค่าซอฟต์แวร์ที่ได้จากโครงข่ายประสาทเทียมที่ 1, 2, 3 และ 4 ตามลำดับ

a, b, c, d คือ ค่าสัมประสิทธิ์ของฟังก์ชัน

จากสมการที่ (8) ผลลัพธ์ที่ได้จากแต่ละโครงข่ายประสาทเทียมจะถูกรวมเข้าด้วยกันเป็นค่าเดียว และค่าสัมประสิทธิ์ที่ปรากฏในฟังก์ชันจะเป็นตัวช่วยปรับปรุงคำตอบของการประมาณมูลค่าซอฟต์แวร์ให้ใกล้เคียงกับค่าจริงมากยิ่งขึ้น โดยงานวิจัยนี้จะใช้ขั้นตอนวิธีกลยุทธ์เชิงวิวัฒนาการมาช่วยในการหาค่าสัมประสิทธิ์ของฟังก์ชันที่เหมาะสมที่สุด มีรายละเอียด ดังนี้

เริ่มต้นการทำงานของกลยุทธ์เชิงวิวัฒนาการ ด้วยการสุ่มค่าจำนวนจริงทั้งหมด 4 ค่า ตามจำนวนสัมประสิทธิ์ที่ปรากฏในสมการที่ (8) และ สุ่มค่าส่วนเบี่ยงเบนมาตรฐาน 1 ค่า เพื่อเป็นตัวแทนคำตอบเริ่มต้น ตัวอย่างแสดงดังรูปที่ 3-2

1.24	2.18	0.95	1.56	0.50
a	b	c	d	σ

รูปที่ 3-2 การกำหนดค่าเริ่มต้นสำหรับกลยุทธ์เชิงวิวัฒนาการ

ทำการประเมินค่าความเหมาะสมของคำตอบของกลยุทธ์เชิงวิวัฒนาการ ดังนี้

คำนวณหาผลลัพธ์ของการประมาณมูลค่าซอฟต์แวร์โดยการแทนค่าตัวเลขต่างๆในสมการที่ 8 ตัวอย่างเช่น สมมุติให้ผลลัพธ์ที่ได้จากโครงข่ายประสาทเทียมทั้ง 4 เป็นดังนี้ 54, 86, 64 และ 72 จะได้ผลลัพธ์เป็น 427.56 ซึ่งคำนวณได้จาก ((1.24 x 54) + (2.18 x 86) + (0.95 x 64) + (1.56 x 72))

เปรียบเทียบผลลัพธ์ที่ได้กับค่าความพยายามจริงของข้อมูลนำเข้านี้ ค่าความแตกต่างที่ได้ คือ ค่าความผิดพลาดของการประมาณ

ดำเนินการตามข้อ 1 และ 2 จนครบทั้งชุดข้อมูลนำเข้าที่เตรียมไว้สำหรับกระบวนการปรับปรุงคำตอบ และคำนวณค่า MMSE ซึ่งคำนวณได้ดังสมการที่ 9 ค่าที่ได้คือค่าความเหมาะสมของคำตอบนี้

$$MMRE = \frac{1}{N}\sum_{i}^{N}\frac{|Actual\ Effort_i - Predicted\ Effort_i|}{Actual\ Effort_i} \qquad (9)$$

สร้างคำตอบใหม่ด้วยการสุ่มค่าจากการกระจายแบบปกติที่มีค่าส่วนเบี่ยงเบนมาตรฐานเท่ากับ σ นั่นคือ $Z \sim N(0,\sigma)$ จากนั้นนำค่าที่สุ่มได้นี้สำหรับแต่ละตำแหน่ง บวกกับค่าในแต่ละตำแหน่งของคำตอบเริ่มต้นทั้ง 4 ตัว ได้เป็นคำตอบชุดใหม่ จากนั้นทำการปรับปรุงค่าส่วนเบี่ยงเบนมาตรฐานใหม่ตามกฎความสำเร็จ 1/5

สำหรับการปรับค่าส่วนเบี่ยงเบนมาตรฐานตามกฎของความสำเร็จ 1/5 (Anne Auger, 2009) ได้เสนอวิธีการปรับค่าส่วนเบี่ยงเบนมาตรฐานไว้ดังสมการที่ 10

$$\sigma = \begin{cases} 1.5 \times \sigma & if(f(\tilde{x}_n) < f(x_n)) \\ \sigma & if(f(\tilde{x}_n) < f(x_n)) \\ 1.5^{-1/4} \times \sigma & if(f(\tilde{x}_n) < f(x_n)) \end{cases} \qquad (10)$$

โดย σ คือ ค่าส่วนเบี่ยงเบนมาตรฐาน

$f(\tilde{x}_n)$ คือ ค่าความเหมาะสมของคำตอบใหม่

$f(x_n)$ คือ ค่าความเหมาะสมของคำตอบเริ่มต้น

ตัวอย่างการปรับปรุงคำตอบเริ่มต้นไปเป็นคำตอบใหม่ด้วยเทคนิคการกลายพันธุ์แบบเกาส์ แสดงดังรูปที่ 3-3

	a	b	c	d
คำตอบเริ่มต้น	1.24	2.18	0.95	1.56
$Z \sim N(0,\sigma)$	0.11	-0.23	0.07	-0.44
คำตอบใหม่	1.35	1.95	1.02	0.79

รูปที่ 3-3 ตัวอย่างการปรับปรุงคำตอบเริ่มต้นไปเป็นคำตอบใหม่ด้วยเทคนิคการกลายพันธุ์แบบเกาส์

คำนวณค่าความเหมาะสมของคำตอบใหม่ที่ได้จากการกลายพันธุ์แบบเกาส์ในขั้นตอนที่ 3 คัดเลือกคำตอบที่ดีกว่า โดยการเปรียบเทียบค่าความเหมาะสมระหว่างคำตอบเริ่มต้น กับ คำตอบใหม่ที่ได้จากการกลายพันธุ์แบบเกาส์ นั่นคือ ถ้าค่าความเหมาะสมของคำตอบไหนมีค่าน้อยกว่า คำตอบนั้นจะผ่านการคัดเลือกไปเป็นคำตอบตั้งต้นในการปรับปรุงคำตอบในรอบถัดไป (เนื่องจากค่าความเหมาะสมในที่นี้คือค่าความผิดพลาดในการประมาณมูลค่า ดังนั้น ค่ายิ่งน้อยยิ่งแสดงว่าสามารถประมาณค่าได้ใกล้เคียงค่าจริง)

ดำเนินการตั้งแต่ข้อ 3 – 6 จนกว่าจะครบตามจำนวนรอบของการปรับปรุงคำตอบที่กำหนด ซึ่งในงานวิจัยนี้กำหนดไว้ที่ 10 รอบ

บทที่ 4 ผลการทดลอง

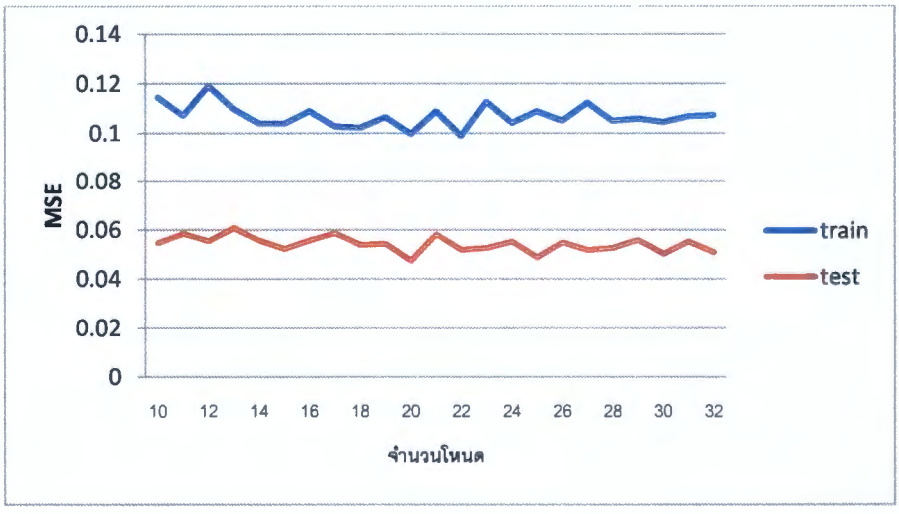
จากการดำเนินงานที่ได้ออกแบบไว้ในบทที่ 3 และผู้วิจัยได้รวบรวมข้อมูลจากฐานข้อมูลมาตรฐานในการพยากรณ์ราคาซอฟต์แวร์ของ NASA โดยข้อมูลที่รวบรวมมานี้เป็นข้อมูล 16 มิติ จำนวน 2 ชุดข้อมูลทดสอบ นั่นคือ ข้อมูลทดสอบชุดที่ 1 จำนวน 60 ชุด และ ข้อมูลทดสอบชุดที่ 2 จำนวน 93 ชุด รายละเอียดของผลการทดลองทั้งหมด มีดังนี้

4.1 ผลการศึกษาการปรับเปลี่ยนพารามิเตอร์ต่างๆของโครงข่ายประสาทเทียม

จากการศึกษาการเปลี่ยนแปลงของค่าพารามิเตอร์ต่างๆ ที่มีผลต่อการเรียนรู้ของโครงข่ายประสาทเทียมที่เหมาะสมสำหรับการประมาณมูลค่าซอฟต์แวร์ 4 ตัวแบบ ได้แก่ โครงข่ายประสาทเทียมที่ประกอบด้วย 1 ชั้นซ่อน จำนวน 2 ตัวแบบ และ โครงข่ายประสาทเทียมที่ประกอบด้วย 2 ชั้นซ่อน จำนวน 2 ตัวแบบ โดยใช้ข้อมูลชุดที่ 1 ในการทดลอง ซึ่งมีรายละเอียดดังนี้

4.1.1 ผลการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อน

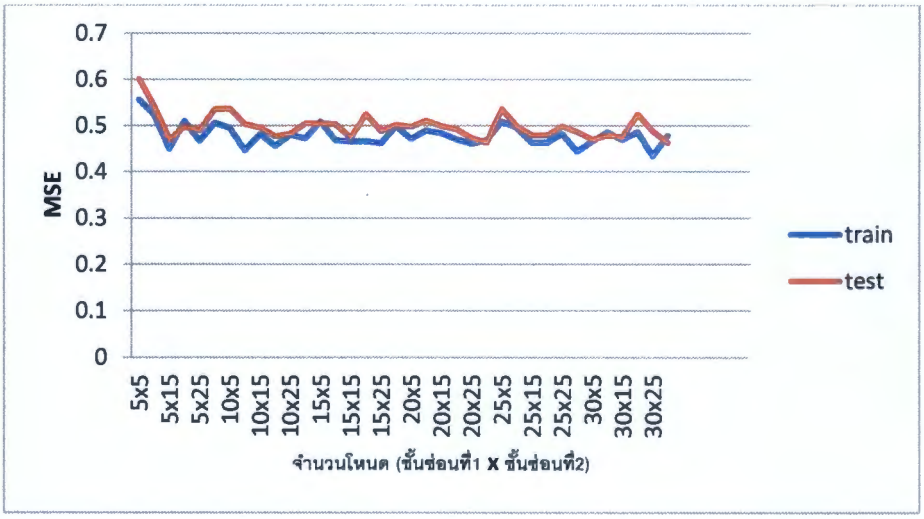
การศึกษาการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อนกรณีโครงข่ายประสาทเทียมแบบ 1 ชั้นซ่อน โดยการเปลี่ยนแปลงจำนวนโหนดตั้งแต่ 10 – 32 พบว่าจำนวนโหนดในชั้นซ่อนมีผลกระทบต่อประสิทธิภาพของโครงข่ายประสาทเทียม ผลการทดลองที่ได้แสดงดังรูปที่ 4-1



รูปที่ 4-1 ผลการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อนกรณีโครงข่ายประสาทเทียมแบบ 1 ชั้นซ่อน

จากรูปที่ 4-1 พบว่า จำนวนโหนดที่เหมาะสมซึ่งให้ค่า MSE ต่ำสุด 2 อันดับแรกโดยพิจารณาภาพรวมทั้งในขั้นตอนของการเรียนรู้ และ การทดสอบ คือ 20 โหนด และ 30 โหนด โดยมีค่า MSE ในขั้นตอนของการเรียนรู้เท่ากับ 0.099605 และ 0.104289 ตามลำดับ และ ค่า MSE ในขั้นตอนของการสอนเท่ากับ 0.047459 และ 0.050147 ตามลำดับ

สำหรับการศึกษาการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อนกรณีโครงข่ายประเภท 2 ชั้นซ่อน โดยการเปลี่ยนแปลงจำนวนโหนดในแต่ละชั้นซ่อนจาก 5 โหนด และเพิ่มขึ้นจำนวนขึ้นครั้งละ 5 โหนดไปเรื่อย ๆ จนถึง 30 โหนด ผลการทดลองแสดงดังรูปที่ 4-2



รูปที่ 4-2 ผลการเปลี่ยนแปลงจำนวนโหนดในชั้นซ่อนกรณีโครงข่ายประสาทเทียมแบบ 2 ชั้นซ่อน

จากรูปที่ 4-2 พบว่า จำนวนโหนดที่เหมาะสมซึ่งให้ค่า MSE ต่ำสุด 2 อันดับแรกโดยดูภาพรวมทั้งในขั้นตอนของการเรียนรู้ และ การทดสอบ คือ

- โครงสร้าง 20x30 (ชั้นซ่อนที่ 1 จำนวน 20 โหนด และ ชั้นซ่อนที่ 2 จำนวน 30 โหนด) โดยมีค่า MSE ในขั้นตอนของการเรียนรู้ และ ขั้นตอนการทดสอบ เท่ากับ 0.468062 และ 0.462599 ตามลำดับ
- โครงสร้าง 25x15 (ชั้นซ่อนที่ 1 จำนวน 25 โหนด และ ชั้นซ่อนที่ 2 จำนวน 15 โหนด) โดยมีค่า MSE ในขั้นตอนของการเรียนรู้ และ ขั้นตอนการทดสอบ เท่ากับ 0.462832 และ 0.478664 ตามลำดับ

4.1.2 ผลการเปลี่ยนแปลงอัตราการเรียนรู้

เนื่องจากในกระบวนการเรียนรู้ของโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ ค่าของอัตราการเรียนรู้มีผลกระทบต่อการหาค่าน้ำหนักที่เหมาะสม งานวิจัยนี้จึงได้ทำการเปลี่ยนแปลงค่าอัตราการเรียนรู้ในช่วง 0.1 – 0.9 ผลการศึกษาสำหรับโครงข่ายประสาทเทียมแบบ 1 ชั้นซ่อนแสดงดังตารางที่ 4-1

ตารางที่ 4-1 ผลการเปลี่ยนแปลงค่าอัตราการเรียนของโครงข่ายประสาทเทียมที่มี 1 ชั้นซ่อน

อัตราการเรียนรู้	โครงข่าย 1 (16x20x1)		โครงข่าย 2 (16X30X1)	
	การเรียนรู้	การทดสอบ	การเรียนรู้	การทดสอบ
0.1	0.099605	0.047459	0.104289	0.050147
0.2	0.097136	0.047029	0.099111	0.049837
0.3	0.096642	0.04722	0.098766	0.049813
0.4	0.096259	0.047403	0.098606	0.049788
0.5	0.095903	0.047608	0.098565	0.049732
0.6	0.095641	0.047812	0.098613	0.049736
0.7	0.095458	0.048042	0.098755	0.049807
0.8	0.095368	0.048255	0.098934	0.049943
0.9	0.095314	0.048514	0.099193	0.050093

จากตารางที่ 4-1 แสดงให้เห็นว่า ค่าอัตราการเรียนรู้ที่เหมาะสมสำหรับโครงข่ายประสาทเทียม แบบ 1 ชั้นซ่อน คือ

- โครงข่ายประสาทเทียมที่มีสถาปัตยกรรมแบบ 16x20x1 ควรกำหนดค่าอัตราการเรียนรู้เท่ากับ 0.2 จึงจะให้ค่า MSE ในภาพรวมทั้งขั้นตอนการเรียนรู้ และ ขั้นตอนการทดสอบต่ำสุด นั่นคือ มีค่า MSE เท่ากับ 0.097136 และ 0.047029 ตามลำดับ

- โครงข่ายประสาทเทียมที่มีสถาปัตยกรรมแบบ 16x30x1 ควรกำหนดค่าอัตราการเรียนรู้เท่ากับ 0.6 โดยให้ค่า MSE ในขั้นตอนของการเรียนรู้ และขั้นตอนการทดสอบต่ำสุด นั่นคือ 0.098613 และ 0.049736 ตามลำดับ

ในส่วนของการศึกษาค่าการเปลี่ยนแปลงอัตราการเรียนรู้สำหรับโครงข่ายประสาทเทียมแบบ 2 ชั้นซ่อนก็ทำในลักษณะเดียวกัน นั่นคือ ทำการเปลี่ยนแปลงค่าอัตราการเรียนรู้ในช่วง 0.1 – 0.5 ผลการศึกษาที่ได้แสดงดังตารางที่ 4-2

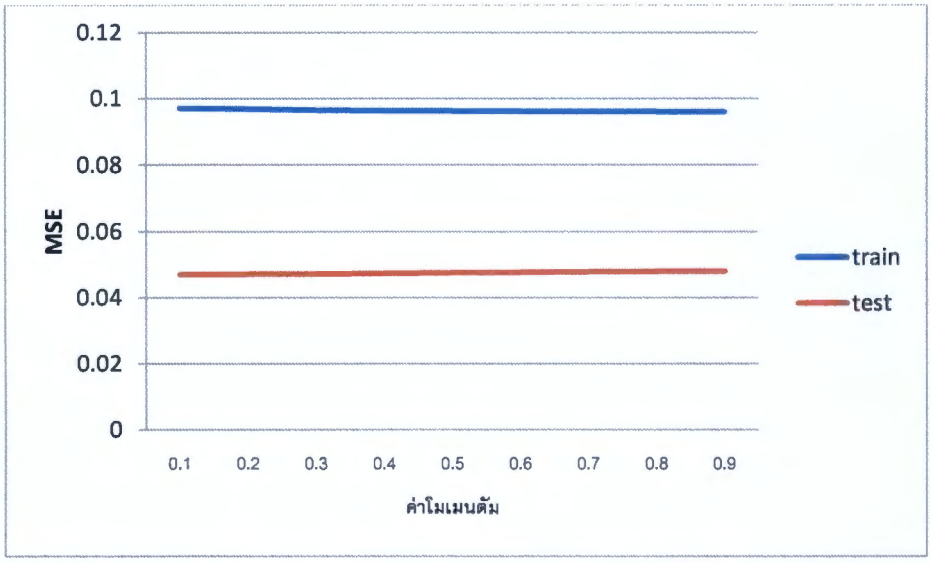
ตารางที่ 4-2 ผลการเปลี่ยนแปลงค่าอัตราการเรียนของโครงข่ายประสาทเทียมที่มี 2 ชั้นซ่อน

อัตราการเรียนรู้	โครงข่าย 3 (16x20x30x1)		โครงข่าย 4 (16X25X15x1)	
	การเรียนรู้	การทดสอบ	การเรียนรู้	การทดสอบ
0.1	0.099605	0.047459	0.104289	0.050147
0.2	0.097136	0.047029	0.099111	0.049837
0.3	0.096642	0.04722	0.098766	0.049813
0.4	0.096259	0.047403	0.098606	0.049788
0.5	0.095903	0.047608	0.098565	0.049732

จากตารางที่ 4-2 แสดงให้เห็นว่า ค่าอัตราการเรียนรู้ที่เหมาะสมของโครงข่ายประสาทเทียมที่มี 2 ชั้นซ่อน ของทั้ง 2 โครงข่าย คือ 0.1 โดยให้ค่า MSE สำหรับโครงข่ายประสาทเทียมที่มีสถาปัตยกรรมเป็น 16x20x30x1 ในขั้นตอนการเรียนรู้ และ ขั้นตอนการสอน เป็น 0.099605 และ 0.047459 ตามลำดับ ส่วนโครงข่ายประสาทเทียมที่มีสถาปัตยกรรมเป็น 16x25x15x1 ผลการศึกษาได้ค่า MSE ในขั้นตอนการเรียนรู้ และ ขั้นตอนการทดสอบ คือ 0.104289 และ 0.050147 ตามลำดับ

4.1.3 ผลการเปลี่ยนแปลงค่าโมเมนตัม

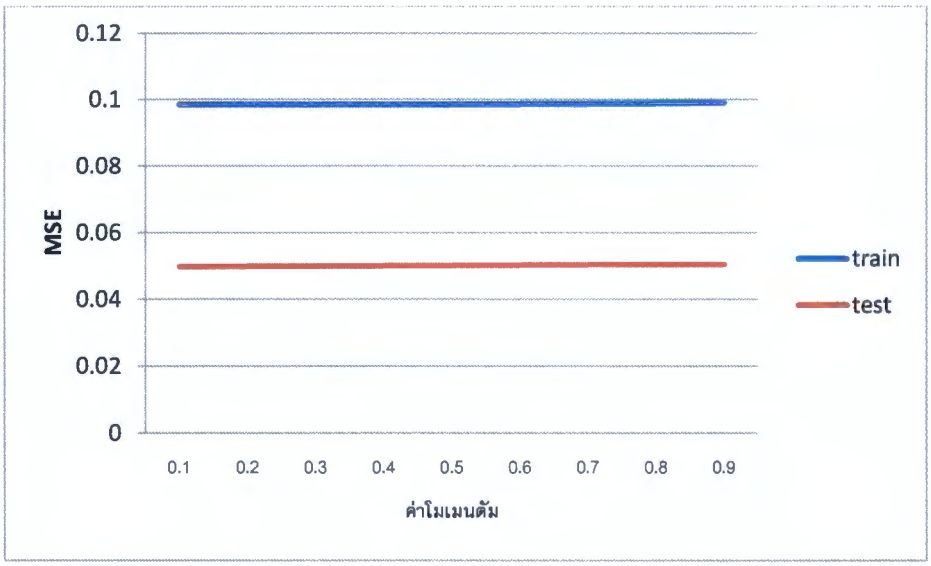
การกำหนดค่าโมเมนตัมในโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ มีวัตถุประสงค์เพื่อช่วยในการปรับปรุงค่าน้ำหนักของกระบวนการเรียนรู้ในรอบถัดๆไป ดังนั้น เพื่อให้โครงข่ายประสาทเทียมลู่เข้าสู่ค่าน้ำหนักที่เหมาะสมได้รวดเร็วขึ้น จึงได้ทำการทดสอบเปลี่ยนแปลงค่าโมเมนตัมตั้งแต่ 0.1 – 0.9 สำหรับโครงข่ายประสาทเทียมที่มี 1 ชั้นซ่อน โดยผลการศึกษาของโครงข่ายประสาทเทียมที่มีสถาปัตยกรรมเป็น 16x20x1 แสดงดังรูปที่ 4-3



รูปที่ 4-3 ผลการทดสอบค่าโมเมนตัมสำหรับโครงข่ายประสาทเทียม 16x20x1

จากผลการศึกษาที่แสดงในรูปที่ 4-3 พบว่า ค่าโมเมนตัมในช่วง 0.1 – 0.9 ให้ค่าความคลาดเคลื่อน (MSE) ที่ใกล้เคียงกัน ดังนั้น ในงานวิจัยนี้จึงกำหนดค่าโมเมนตัมสำหรับโครงข่ายประสาทเทียมที่มีสถาปัตยกรรมเป็น 16x20x1 ไว้เท่ากับ 0.1

ผลการศึกษาการเปลี่ยนแปลงค่าโมเมนตัมของโครงข่ายประสาทเทียมที่มีสถาปัตยกรรมเป็น 16x30x1 ทำในลักษณะเดียวกัน นั่นคือ ทำการเปลี่ยนแปลงค่าโมเมนตัมในช่วง 0.1- 0.9 ผลลัพธ์ที่ได้แสดงดังรูปที่ 4-4 ซึ่งผลลัพธ์นี้สอดคล้องกับการศึกษาก่อนหน้า คือ ค่าความคลาดเคลื่อนที่ได้ไม่แตกต่างกันมากนั้น ดังนั้น จึงกำหนดค่าโมเมนตัมสำหรับสถาปัตยกรรมนี้เป็น 0.1



รูปที่ 4-4 ผลการทดสอบค่าโมเมนตัมสำหรับโครงข่ายประสาทเทียม 16x30x1

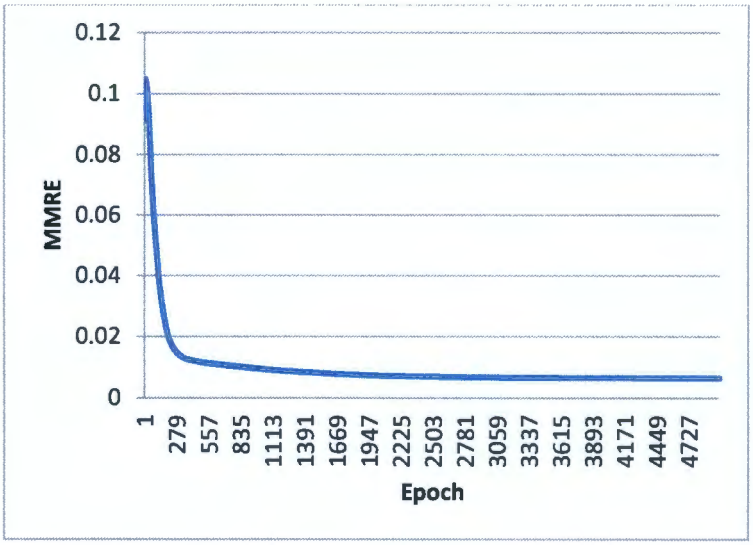
4.2 ผลการพยากรณ์มูลค่าซอฟต์แวร์ของแต่ละตัวแบบ

จากการศึกษาค่าพารามิเตอร์ต่างๆเพื่อหาโครงสร้างที่เหมาะสมสำหรับการประมาณมูลค่าซอฟต์แวร์ในหัวข้อที่ผ่านมา ในหัวข้อนี้ เราจะนำตัวแบบทั้ง 4 ไปทำการทดสอบกับชุดข้อมูลทดสอบที่เตรียมไว้ทั้ง 2 ชุด นั่นคือ ชุดทดสอบที่ 1 จำนวน 60 ชุดข้อมูล และ ชุดทดสอบที่ 2 จำนวน 93 ชุดข้อมูล

การประเมินประสิทธิภาพการประเมินมูลค่าซอฟต์แวร์ของแต่ละโครงข่ายประสาทเทียมในงานวิจัยนี้ประเมินจากค่าเฉลี่ยความคลาดเคลื่อนสัมพัทธ์ (Mean Magnitude of Relative Error: MMRE) ซึ่งคำนวณได้จากสมการที่ 9 หากค่า MMRE มีค่าสูง แสดงว่าร้อยละของความคลาดเคลื่อนระหว่างราคาซอฟต์แวร์จริง กับ ราคาซอฟต์แวร์ที่ได้จากการพยากรณ์ นั่นหมายความว่า ค่า MMRE ยิ่งน้อย หมายถึง ความแม่นยำที่สูง

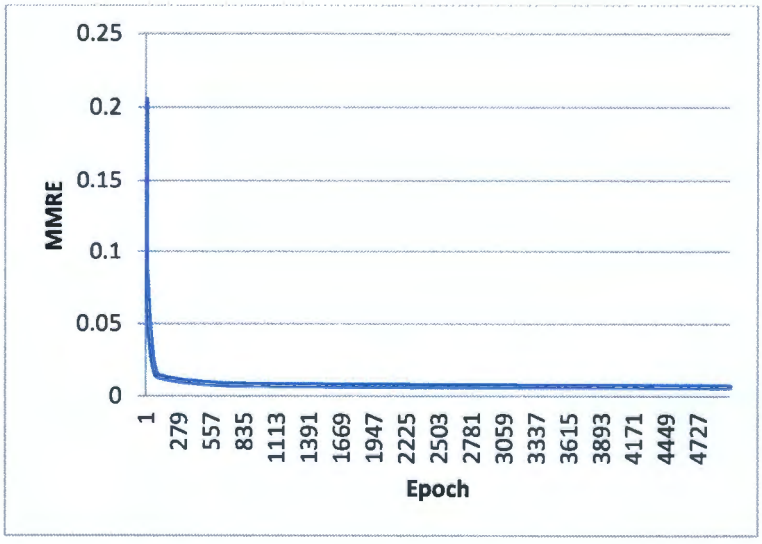
4.2.1 ผลการทดสอบกับชุดข้อมูลจำนวน 60

ในการทดลองจะทำการแบ่งข้อมูลออกเป็นชุดย่อยๆ ทั้งหมด 5 ชุด โดยผลลัพธ์ของค่าความคลาดเคลื่อนในกระบวนการเรียนรู้ทั้งหมด 5000 รอบ ของโครงข่ายประสาทเทียมทั้ง 4 ตัวแบบ นั่นคือ 16x20x1, 16x30x1, 16x20x30x1 และ 16x25x15x1 แสดงดังรูปที่ 4-5, 4-6, 4-7 และ 4-8 ตามลำดับ โดยค่า MMRE ที่ปรากฏจะเป็นค่าเฉลี่ยของผลการเรียนรู้ข้อมูลทั้ง 5 ชุดย่อย



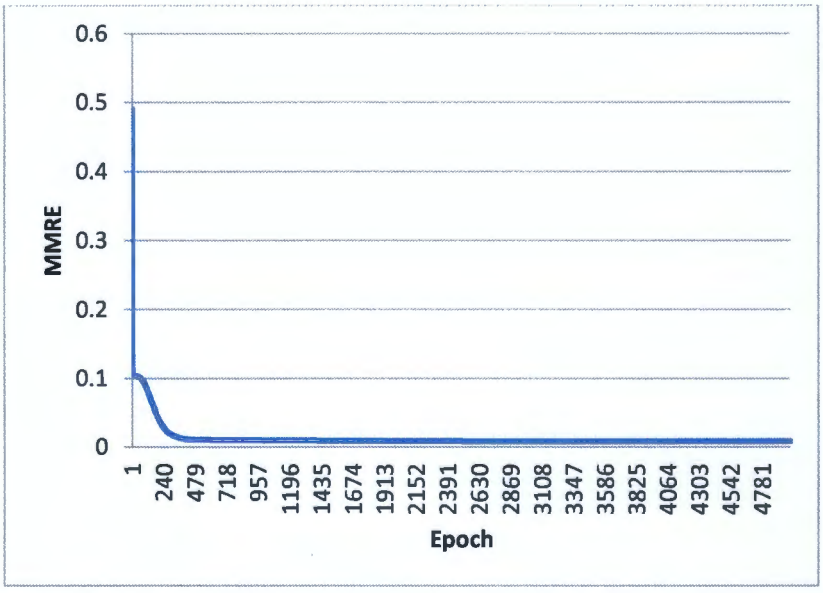
รูปที่ 4-5 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x20x1 สำหรับชุดข้อมูลทดสอบที่ 1

จากรูปที่ 4-5 แสดงให้เห็นว่าโครงข่ายประสาทเทียมที่มีโครงสร้าง 16x20x1 เริ่มต้นกระบวนการเรียนรู้โดยมีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.091263 จากนั้นค่าความคลาดเคลื่อนที่ได้จากกระบวนการเรียนรู้ของโครงข่ายนี้จะลดลงเรื่อยๆ จนกระทั่งประมาณรอบการเรียนรู้ที่ 3500 ค่าความคลาดเคลื่อนเริ่มลดลงเพียงเล็กน้อย และเมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนของโครงข่ายประสาทเทียมนี้มีค่าเท่ากับ 0.006441



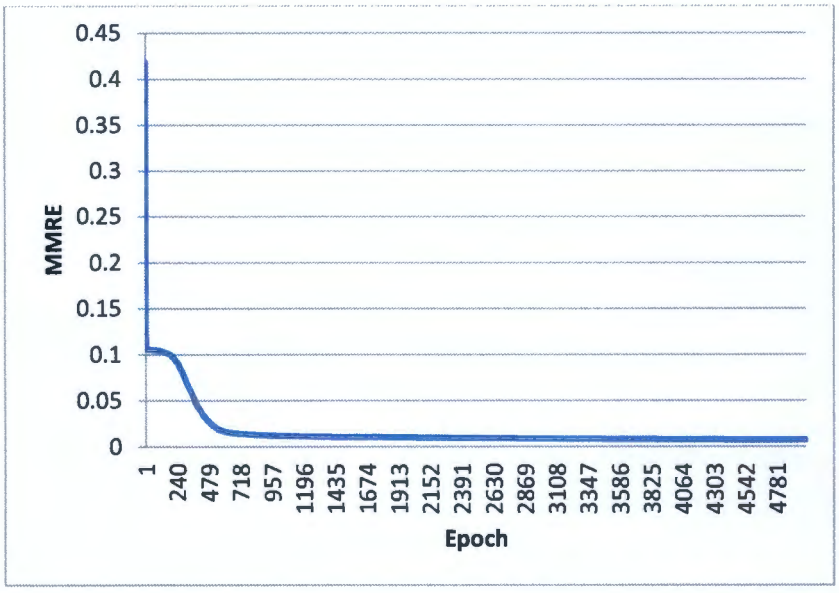
รูปที่ 4-6 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x30x1 สำหรับชุดข้อมูลทดสอบที่ 1

จากรูปที่ 4-6 แสดงให้เห็นว่าโครงข่ายประสาทเทียมที่มีโครงสร้าง 16x30x1 เริ่มต้นกระบวนการเรียนรู้โดยมีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.205353 ซึ่งเป็นค่าที่ค่อนข้างสูงเมื่อเทียบกับการเริ่มต้นกระบวนการเรียนรู้ของตัวแบบก่อนหน้า (16x20x1) แต่เมื่อผ่านการปรับปรุงค่าน้ำหนักไปได้ระยะหนึ่งประมาณรอบการเรียนรู้ที่ 718 ค่าความคลาดเคลื่อนเริ่มมีการเปลี่ยนแปลงลดลงเพียงเล็กน้อย และสิ้นสุดกระบวนการเรียนรู้ด้วยค่าความคลาดเคลื่อนเท่ากับ 0.006452



รูปที่ 4-7 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x20x30x1 สำหรับชุดข้อมูลทดสอบที่ 1

จากรูปที่ 4-7 โครงข่ายประสาทเทียมนี้เป็นแบบ 2 ชั้นซ่อน มีโครงสร้างเป็น 16x20x30x1 ตอนเริ่มต้นของกระบวนการเรียนรู้มีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.491566 และเมื่อผ่านการปรับปรุงค่าน้ำหนักไปได้ระยะหนึ่งประมาณรอบการเรียนรู้ที่ 240 ค่าความคลาดเคลื่อนเริ่มมีการเปลี่ยนแปลงลดลงเพียงเล็กน้อย และเริ่มคงที่เมื่อรอบการเรียนรู้ที่ 2152 และสิ้นสุดกระบวนการเรียนรู้ด้วยค่าความคลาดเคลื่อนเท่ากับ 0.0074

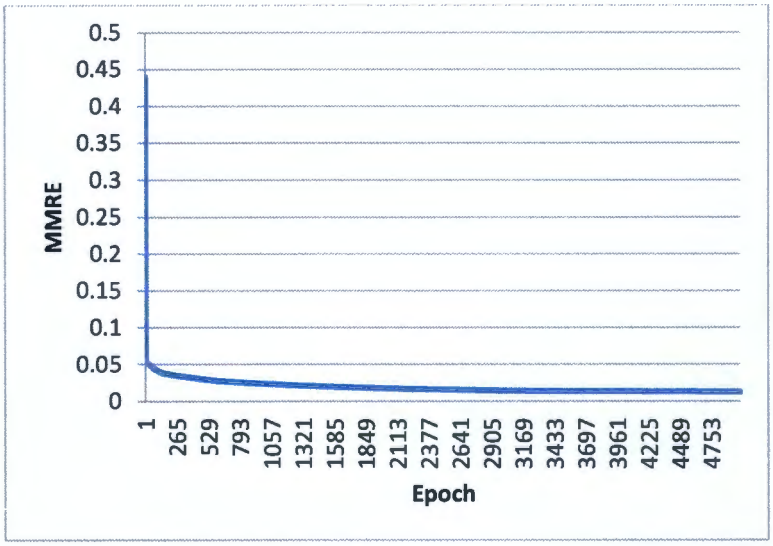


รูปที่ 4-8 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x25x15x1 สำหรับชุดข้อมูลทดสอบที่ 1

จากรูปที่ 4-8 โครงข่ายประสาทเทียมนี้เป็นแบบ 2 ชั้นซ่อนเช่นเดียวกับโครงข่ายประสาทเทียมในรูปที่ 4-7 โดยโครงสร้างของสถาปัตยกรรมเป็น 16x25x15x1 สำหรับตัวแบบนี้มีค่าความคลาดเคลื่อน (MMRE) ตอนเริ่มต้นของกระบวนการเรียนรู้ เท่ากับ 0.418636 และเริ่มมีการเปลี่ยนแปลงค่าความคลาดเคลื่อนอย่างเห็นได้ชัดในรอบการเรียนรู้ต้นๆ จากนั้นค่อยปรับปรุงค่าน้ำหนักอันมีผลให้ค่าความคลาดเคลื่อนของการพยากรณ์ลดลงเรื่อยๆ จนเริ่มคงที่ประมาณรอบการเรียนรู้ที่ 2152 และสิ้นสุดกระบวนการเรียนรู้ด้วยค่าความคลาดเคลื่อนเท่ากับ 0.007249

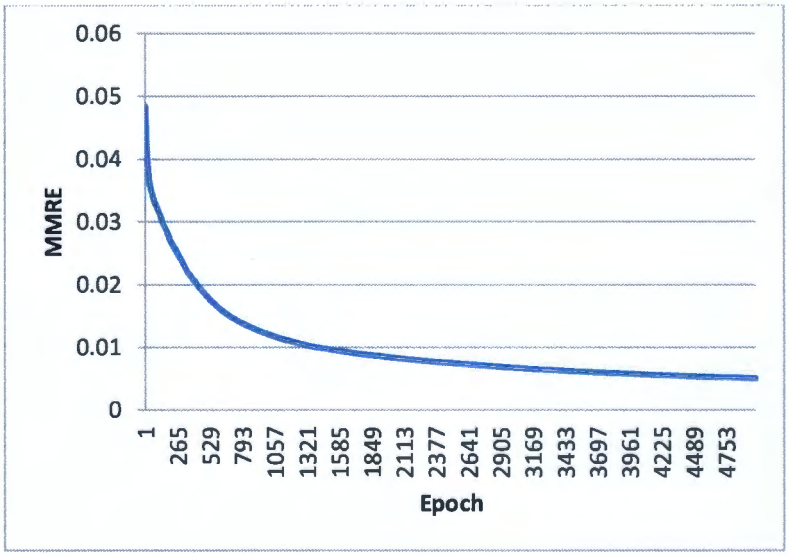
4.2.2 ผลการทดสอบกับชุดข้อมูลจำนวน 93

สำหรับการทดลองในส่วนนี้จะทำในลักษณะเดียวกันกับกระบวนการเรียนรู้กับชุดข้อมูลทดสอบก่อนหน้า โดยผลลัพธ์ของค่าความคลาดเคลื่อนในกระบวนการเรียนรู้ทั้งหมด 5000 รอบ ของโครงข่ายประสาทเทียมทั้ง 4 ตัวแบบ นั่นคือ 16x20x1, 16x30x1, 16x20x30x1 และ 16x25x15x1 แสดงดังรูปที่ 4-9, 4-10, 4-11 และ 4-12 ตามลำดับ โดยค่า MMRE ที่ปรากฏจะเป็นค่าเฉลี่ยของผลการเรียนรู้ข้อมูลทั้ง 5 ชุดย่อย



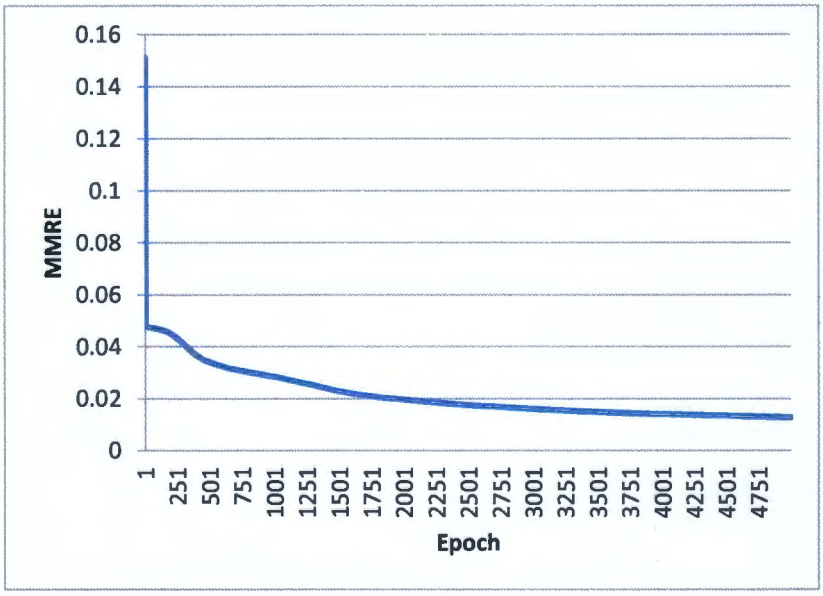
รูปที่ 4-9 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x20x1 สำหรับชุดข้อมูลทดสอบที่ 2

จากรูปที่ 4-9 แสดงให้เห็นว่าโครงข่ายประสาทเทียมที่มีโครงสร้าง 16x20x1 เริ่มต้นกระบวนการเรียนรู้โดยมีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.4402 จากนั้นค่าความคลาดเคลื่อนที่ได้จากกระบวนการเรียนรู้ของโครงข่ายนี้จะลดลงเรื่อยๆ จนกระทั่งประมาณรอบการเรียนรู้ที่ 2869 ค่าความคลาดเคลื่อนเริ่มลดลงเพียงเล็กน้อย และเมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนของโครงข่ายประสาทเทียมนี้มีค่าเท่ากับ 0.012232



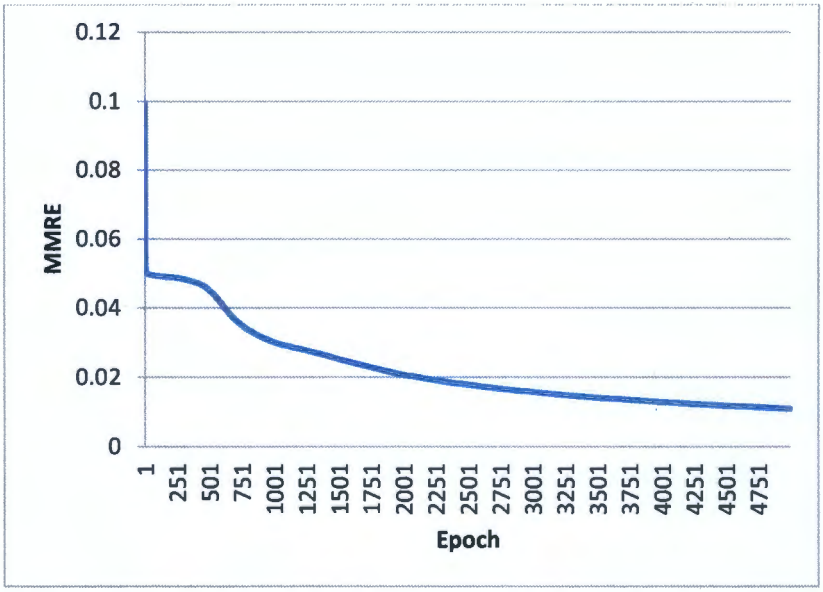
รูปที่ 4-10 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x30x1 สำหรับชุดข้อมูลทดสอบที่ 2

จากรูปที่ 4-10 แสดงให้เห็นว่าโครงข่ายประสาทเทียมที่มีโครงสร้าง 16x30x1 เริ่มต้นกระบวนการเรียนรู้โดยมีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.048488 ซึ่งใกล้เคียงกับค่าความคลาดเคลื่อนในตอนเริ่มต้นของตัวแบบก่อนหน้า (16x20x1) และเมื่อผ่านการปรับปรุงค่าน้ำหนักไปเรื่อยๆค่าความคลาดเคลื่อนค่อยๆลดลงตามลำดับ จนเมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนของตัวแบบนี้มีค่าเท่ากับ 0.004975



รูปที่ 4-11 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x20x30x1 สำหรับชุดข้อมูลทดสอบที่ 2

จากรูปที่ 4-11 โครงข่ายประสาทเทียมนี้เป็นแบบ 2 ชั้นซ่อน มีโครงสร้างเป็น 16x20x30x1 ตอนเริ่มต้นของกระบวนการเรียนรู้มีค่าความคลาดเคลื่อน (MMRE) เท่ากับ 0.151269 ซึ่งน้อยกว่าค่าความคลาดเคลื่อนเริ่มต้นของ 2 ตัวแบบที่ผ่านมา เริ่มมีการเปลี่ยนแปลงที่ชัดเจนในรอบการเรียนรู้ที่ 251 และ ค่าความคลาดเคลื่อนค่อยๆลดลงเรื่อยๆ โดยเมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนของตัวแบบนี้มีเท่ากับ 0.012663



รูปที่ 4-12 ค่าความคลาดเคลื่อน (MMRE) ในกระบวนการเรียนรู้ของตัวแบบ 16x25x15X1 สำหรับชุดข้อมูลทดสอบที่ 2

จากรูปที่ 4-12 โครงข่ายประสาทเทียมนี้เป็นแบบ 2 ชั้นซ่อน มีโครงสร้างของสถาปัตยกรรมเป็น 16x25x15x1 สำหรับตัวแบบนี้มีค่าความคลาดเคลื่อน (MMRE) ตอนเริ่มต้นของกระบวนการเรียนรู้ เท่ากับ 0.099553 ซึ่งถือว่าต่ำที่สุดในจำนวนตัวแบบทั้ง 4 ที่ทดสอบกับชุดข้อมูลทดสอบที่ 2 และเริ่มมีการเปลี่ยนแปลงค่าความคลาดเคลื่อนอย่างเห็นได้ชัดในรอบการเรียนรู้ประมาณ 500 รอบ จากนั้นค่าความคลาดเคลื่อนของการพยากรณ์ลดลงเรื่อยๆ ตามลำดับ จนกระทั่งสิ้นสุดกระบวนการเรียนรู้มีค่าความคลาดเคลื่อนเท่ากับ 0.010825

4.3 ผลการเรียนรู้ของกระบวนการลูกผสมนิวรอนเน็ตเวิร์ค

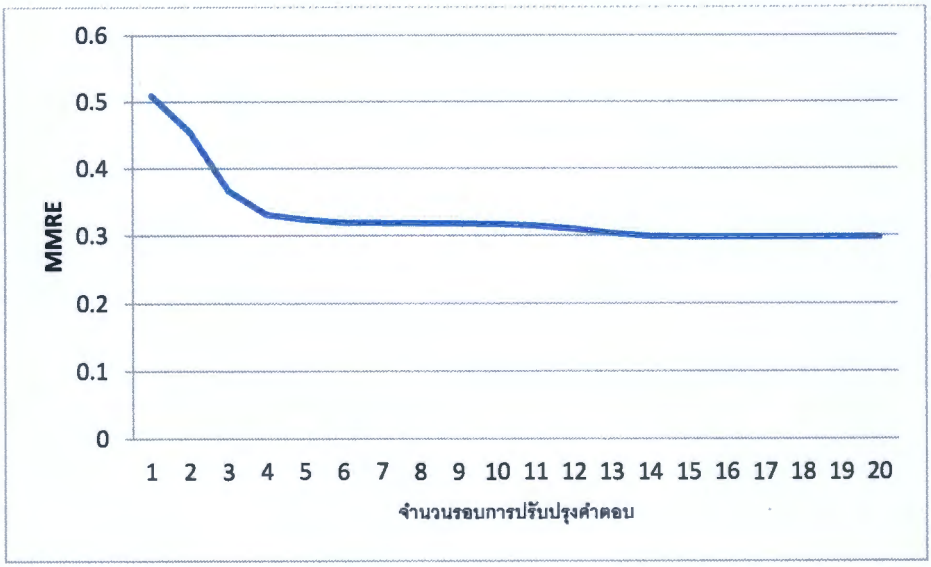
หลังจากโครงข่ายประสาทเทียมทั้ง 4 ตัวแบบได้ผ่านกระบวนการเรียนรู้ตามเงื่อนไขที่กำหนดแล้ว ในขั้นตอนนี้จะเป็นการนำผลลัพธ์ของการพยากรณ์จากทั้ง 4 ตัวแบบมาใช้ในกระบวนการหาค่าสัมประสิทธิ์ของฟังก์ชันพยากรณ์ด้วยขั้นตอนวิธีกลยุทธ์เชิงวิวัฒนาการ (ES)

ผลการดำเนินการในขั้นตอนนี้ ได้ดำเนินการกับ 2 ชุดข้อมูลทดสอบดังที่ได้อธิบายไปแล้วในตอนต้น รายละเอียดของผลการทดลองทั้งหมด มีดังนี้

4.3.1 ผลการเรียนรู้กับชุดข้อมูลจำนวน 60

ผลการเรียนรู้ของขั้นตอนวิธีกลยุทธ์เชิงวิวัฒนาการในการปรับปรุงคำตอบเพื่อหาสัมประสิทธิ์ที่เหมาะสมของฟังก์ชันในการพยากรณ์ราคาซอฟต์แวร์สำหรับชุดข้อมูลทดสอบที่ 1 ซึ่งแบ่งข้อมูล

ทดสอบออกเป็น 5 กลุ่มย่อย แสดงดังรูปที่ 4-13 โดยค่าความเคลื่อนที่แสดงดังรูปเป็นค่าความคลาดเคลื่อนเฉลี่ยจาก 5 ชุดข้อมูลย่อย

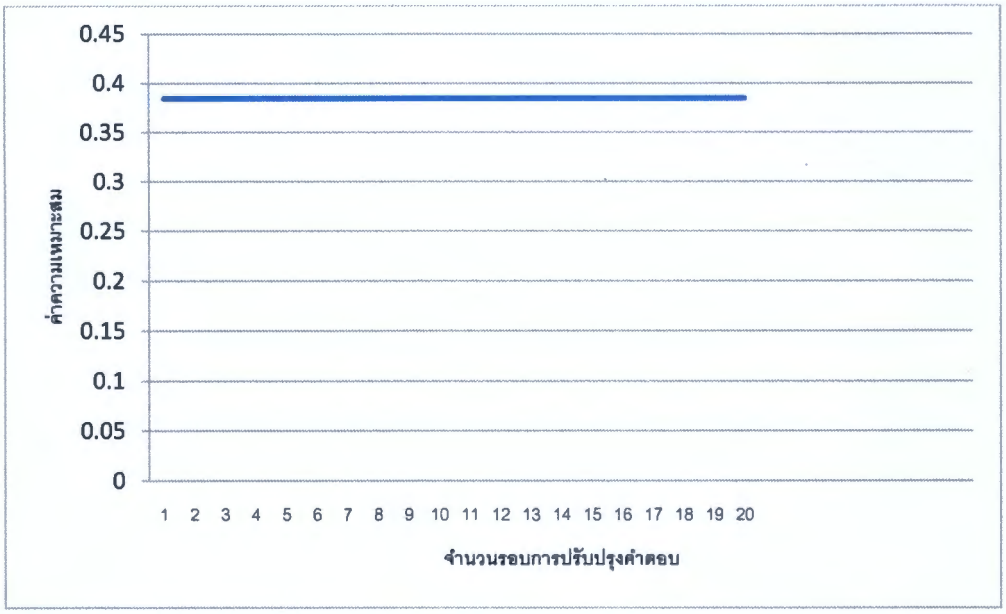


รูปที่ 4-13 ผลการเรียนรู้ของกระบวนการลูกผสมนิวรอนเน็ตเวิร์คสำหรับชุดข้อมูลที่ 1

จากผลการเรียนรู้ของกระบวนการลูกผสมนิวรอนเน็ตเวิร์คดังแสดงในรูปที่ 4-13 พบว่า ในตอนเริ่มต้นค่าความคลาดเคลื่อนที่ได้จากฟังก์ชันพยากรณ์มีค่าเท่ากับ 0.509666 และเมื่อผ่านกระบวนการปรับปรุงคำตอบด้วยกลยุทธ์เชิงวิวัฒนาการ มีผลทำให้ค่าความคลาดเคลื่อนลดลงเรื่อยๆ และเริ่มคงที่เมื่อรอบการปรับปรุงคำตอบที่ 13 และลดลงเพียงเล็กน้อย เมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนสำหรับชุดข้อมูลทดสอบนี้มีค่าเท่ากับ 0.297491

สำหรับสัมประสิทธิ์ของฟังก์ชันการพยากรณ์ทั้ง 4 ตัวแปร (a, b, c, d) ดังสมการที่ 8 ซึ่งได้จากกระบวนการลูกผสมนิวรอนเน็ตเวิร์คนี้ มีค่าเท่ากับ 0.5793, 0.276743, -0.804319 และ 0.690073 ตามลำดับ

4.3.2 ผลการทดสอบกับชุดข้อมูลจำนวน 93



รูปที่ 4-13 ผลการเรียนรู้ของกระบวนการลูกผสมนิวรอนเน็ตเวิร์คสำหรับชุดข้อมูลที่ 2

จากผลการเรียนรู้ของกระบวนการลูกผสมนิวรอนเน็ตเวิร์คดังแสดงในรูปที่ 4-14 พบว่า ค่าความคลาดเคลื่อนที่ได้จากฟังก์ชันพยากรณ์มีค่าเท่ากับ 0.38436 ตั้งแต่รอบการปรับปรุงคำตอบแรกจนสิ้นสุดกระบวนการเรียนรู้ แสดงให้เห็นว่ากลยุทธ์เชิงวิวัฒนาการสามารถหาค่าสัมประสิทธิ์ที่ค่อนข้างเหมาะสมตั้งแต่ในตอนต้นของกระบวนการเรียนรู้ และ ไม่สามารถที่จะหาค่าสัมประสิทธิ์ของฟังก์ชันที่ให้ค่าความคลาดเคลื่อนต่ำลงได้ แต่อย่างไรก็ตามเมื่อสิ้นสุดกระบวนการเรียนรู้ค่าความคลาดเคลื่อนที่ได้ยังคงให้ผลลัพธ์เป็นที่น่าพอใจ

สำหรับสัมประสิทธิ์ของฟังก์ชันการพยากรณ์ทั้ง 4 ตัวแปร (a, b, c, d) ดังสมการที่ 8 ซึ่งได้จากกระบวนการลูกผสมนิวรอนเน็ตเวิร์คสำหรับชุดข้อมูลทดสอบที่ 2 มีค่าเท่ากับ 0.11863, 0.246856, 0.104795 และ 0.623359 ตามลำดับ

4.4 สรุปผลการพยากรณ์ราคาซอฟต์แวร์ของวิธีการที่งานวิจัยนำเสนอ

จากผลการดำเนินงานทั้งหมด สามารถสรุปผลการพยากรณ์ราคาซอฟต์แวร์ด้วยกระบวนการลูกผสมนิวรอนเน็ตเวิร์ค (Ensemble Neural Network) แสดงดังตารางที่ 4-3

ตารางที่ 4-3 ผลการพยากรณ์ราคาซอฟต์แวร์ด้วยกระบวนการลูกผสมนิวรอนเน็ตเวิร์ค

ข้อมูลทดสอบ	วิธีการพยากรณ์	ค่าความคลาดเคลื่อน (MMRE)
NASA'60	COCOMO	0.342957
	16x20x1	0.50947
	16x30x1	0.554956
	16x20x30x1	0.570011
	16x25x15x1	0.580848
	Proposed Method	0.297491
NASA'93	COCOMO	0.542562
	16x20x1	1.59763
	16x30x1	1.577509
	16x20x30x1	1.795303
	16x25x15x1	1.709375
	Proposed Method	0.38436
Average	COCOMO	0.442759
	16x20x1	1.05355
	16x30x1	1.066233
	16x20x30x1	1.182657
	16x25x15x1	1.145112
	Proposed Method	0.340926

จากผลการทดลองดังตารางที่ 4-3 เมื่อพิจารณาในส่วนของชุดข้อมูลทดสอบที่ 1 (NASA'60) พบว่า ค่าความคลาดเคลื่อนที่ได้จากโครงข่ายประสาทเทียมทั้ง 4 ตัวแบบ มีค่าสูงกว่าค่าความคลาดเคลื่อนที่ได้จาก COCOMO แต่เมื่อนำผลลัพธ์จากโครงข่ายประสาทเทียมทั้ง 4 มารวมกันโดยคำนวณค่าด้วยฟังก์ชันการพยากรณ์ดังที่นำเสนอไปในหัวข้อ 4.3.1 พบว่า ค่าความคลาดเคลื่อนของวิธีการที่งานวิจัยนี้นำเสนอมีค่าลดลงเหลือเพียง 0.297491 และมีค่าน้อยกว่าค่าความคลาดเคลื่อนที่ได้จากวิธี COCOMO

สำหรับการทดสอบกับชุดข้อมูลทดสอบที่ 2 (NASA'93) พบว่า ค่าความคลาดเคลื่อนที่ได้จากโครงข่ายประสาทเทียมทั้ง 4 ตัวแบบจะเป็นไปในทิศทางเดียวกันกับการทดสอบกับชุดข้อมูลแรก กล่าวคือ มีค่าสูงกว่าค่าความคลาดเคลื่อนที่ได้จาก COCOMO แต่เมื่อนำผลลัพธ์จากโครงข่าย

ประสาทเทียมทั้ง 4 มารวมกันโดยคำนวณค่าด้วยฟังก์ชันการพยากรณ์ดังที่นำเสนอไปในหัวข้อ 4.3.2 พบว่า ค่าความคลาดเคลื่อนของวิธีการที่งานวิจัยนี้นำเสนอมีค่าลดลงจากค่าความคลาดเคลื่อนประมาณ 1.6 เหลือเพียง 0.38436 และมีค่าน้อยกว่าค่าความคลาดเคลื่อนที่ได้จากวิธี COCOMO

ในคอลัมน์สุดท้ายของตารางที่ 4-3 เป็นการคำนวณค่าความคลาดเคลื่อนเฉลี่ยของการดำเนินการทั้งหมด เพื่อสรุปภาพรวมให้เห็นว่าแต่ละขั้นตอนวิธีที่นำมาเปรียบเทียบ ให้ผลลัพธ์ของการพยากรณ์ราคาซอฟต์แวร์ของทุกชุดการทดสอบเป็นไปในทิศทางใด ผลลัพธ์ที่ได้แสดงให้เห็นว่าวิธีการที่นำเสนอให้ค่าความคลาดเคลื่อนเฉลี่ยของการพยากรณ์ราคาซอฟต์แวร์เท่ากับ 0.340926 ซึ่งน้อยกว่าค่าความคลาดเคลื่อนเฉลี่ยของวิธี COCOMO ซึ่งมีค่าเท่ากับ 0.442759

บทที่ 5 สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

ในงานวิจัยนี้ (ปีงบประมาณ 2555) ได้นำเสนอการพยากรณ์ราคาซอฟต์แวร์ของฐานข้อมูลมาตรฐานของ NASA โดยได้นำเสนอกระบวนการลูกผสมนิวรอนเน็ตเวิร์คซึ่งเป็นการนำขั้นตอนวิธีการรู้จำแบบแพร่ย้อนกลับจำนวน 4 ตัวแบบที่มีสถาปัตยกรรมและพารามิเตอร์แตกต่างกันมาเรียนรู้เพื่อหาผลลัพธ์ของการพยากรณ์ราคาซอฟต์แวร์ จากนั้นรวมผลลัพธ์ที่ได้ด้วยฟังก์ชันการพยากรณ์ที่ใช้กลยุทธ์เชิงวิวัฒนาการในการช่วยหาค่าสัมประสิทธิ์ที่เหมาะสม

ผลการประเมินประสิทธิภาพด้วยค่าความคลาดเคลื่อนเฉลี่ยสัมพัทธ์ (MMRE) แสดงให้เห็นว่าวิธีการที่งานวิจัยนี้นำเสนอให้ผลลัพธ์ของการพยากรณ์ที่ดีกว่าวิธีการของ COCOMO ทั้ง 2 ชุดข้อมูลทดสอบ กล่าวคือ ในชุดข้อมูล 60 วิธีการที่นำเสนอให้ค่าความเคลื่อน 0.297491 และ ค่าความคลาดเคลื่อนในชุดข้อมูล 93 มีค่าเท่ากับ 0.38436 ในขณะที่ค่าความคลาดเคลื่อนที่ได้จากวิธี COCOMO คือ 0.342957 และ 0.542562 ตามลำดับ นอกจากนี้ การเปรียบเทียบค่าความคลาดเคลื่อนเฉลี่ยที่ได้จากทั้ง 2 ชุดข้อมูลทดสอบ แสดงให้เห็นว่า ในภาพรวมวิธีการที่นำเสนอก็ยังคงให้ค่าความคลาดเคลื่อนที่ต่ำกว่าค่าความคลาดเคลื่อนเฉลี่ยของวิธี COCOMO ประมาณ 0.1

5.2 งานที่ต้องทำต่อไปในอนาคต

1. ศึกษาและพัฒนาขั้นตอนวิธีสำหรับการสกัดคุณลักษณะเด่นสำหรับการพยากรณ์มูลค่าซอฟต์แวร์ เพื่อเพิ่มประสิทธิภาพในการพยากรณ์
2. ศึกษาและพัฒนาขั้นตอนวิธีเพื่อการรู้จำแบบอื่น ๆ รวมทั้งการเรียนรู้แบบผสมสำหรับการพยากรณ์มูลค่าซอฟต์แวร์ให้มีประสิทธิภาพสูงขึ้น

บรรณานุกรม

B. Tirimula Rao, B. Sameet, G. Kiran Swathi, K. Vikram Gupta, Ch. RaviTeja, และ S.Sumana, (2009), A Novel Neural Network Approach For Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN), International Journal of Computer Science and Network Security, VOL.9 No.6, pp.126-131.

Charles.W.Therrien, (1989), *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics,* John Wiley & Sons.

D. H. Wolpert, (1992), Stacked Generalization. *Neural Networks*, 5(2), pp. 241–259.

K. Vinay Kumar, V. Ravi *, Mahil Carr และ N. Raj Kiran, (2008), Software development cost estimation using wavelet neural networks, *The Journal of System and Software*, Vol. 81, pp. 1853-1867.

L. Breiman, (1996), Bagging Predictors. *Machine Learning*, 24(2), pp. 123–140.

N. Tadayon, (2005), Neural Network Approach for Software Cost Estimation. *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, Vol.02, pp. 815 – 818.

R. E. Schapire, (1990), The Strength of Weak Learnability. *Machine Learning*, 5(2), pp. 197–227.

S. Haykin, (1998), *Neural Networks: A Comprehensive Foundation (2nd edition)*, Prentice Hall.

Y. Kultur, B. Turhan และ A. Bener (2009), Ensemble of neural networks with associative memory (ENNA) for estimating software development costs, *Knowledge-Based Systems,* Vol. 22, pp. 395-402.

Y. Freund and R. E. Schapire, (1996), Experiments with a New Boosting Algorithm. *Proceedings of the13th International Conference on Machine Learning (ICML'96).* San Francisco, CA: Morgan Kaufmann, pp.148–156.

J. D. Aron, (1969) *Estimating Resource for Large Programming Systems*, NATO Science Committee, Rome, Italy.

R.K.D. Black, R. P. Curnow, R. Katz and M. D. Gray, (1977) , BCS Software Production Data, *Final Technical Report, RADC-TR-77-116*, Boeing Computer Services, Inc.

B. W. Boehm, (1981), *Software engineering economics*, Englewood Cliffs, NJ: Prentice-Hall.

B.W. Boehm et al., (1996), "*The COCOMO 2.0 Software Cost Estimation Model*", American Programmer, pp.2-17.

G. Cantone, A. Cimitile and U. De Carlini, (1986) "*A comparison of models for software cost estimation and management of software projects*", in Computer Systems: Performance andSimulation, Elisevier Science Publishers B.V.

R. E. Park, (1988), "PRICE S: The calculation within and why", *Proceedings of ISPA Tenth Annual Conference*, Brighton, England.

N. A. Parr, (1980), "*An alternative to the Raleigh Curve Model for Software development effort*", IEEE on Software Engineering.

R. Tausworthe, (1981), *Deep Space Network Software Cost Estimation Model*, Jet Propulsion Laboratory Publication 81-7.

R. W. Wolverton, (1974), "*The cost of developing large-scale software*", IEEE Trans. Computer, pp.615-636.