การสร้างเกมให้เรียนรู้ได้

Game By Machine Learning

นายระพีพันธ์ รันทกิจ

นายสุชาติ ชยางศุ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

การสร้างเกมให้เรียนรู้ได้

Game By Machine Learning

โดย

นายระพีพันธ์ รันทกิจ

นายสุชาติ ชยางศุ

อาจารย์ที่ปรึกษา

อ.เกียรติกูล เจียรนัยธนะกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสร้างเกมให้เรียนรู้ได้

Game By Machine Learning

ผู้จัดทำ

1. นายระพีพันธ์ รันทกิจ รหัสประจำตัว 42010290
2. นายสุชาติ ชยางศุ รหัสประจำตัว 42010385

อาจารย์ที่ปรึกษา

(อ.เกียรติกูล เจียรนัยธนะกิจ)

การสร้างเกมให้เรียนรู้ได้

นายระพีพันธ์ รันทกิจ 42010290
นายสุชาติ ชยางศุ 42010385
อาจารย์เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

เนื่องจากในปัจจุบันการพัฒนาเกมได้พยายามทำให้มีความสามารถในการเล่นที่เหมาะสมกับผู้เล่น โดยส่วนมากจะพัฒนาโดยใช้กฎตายตัว (Rule Base) ซึ่งมีความสามารถจำกัด ไม่สามารถปรับปรุงกลยุทธ์ (Tactic) ในการเล่นได้ ดังนั้นการเพิ่มความสามารถในการเรียนรู้เข้าไปในเกม เป็นสิ่งที่จำเป็นเพื่อที่จะสามารถเล่นตอบสนองกับผู้เล่นได้ดีขึ้น

ในปริญญานิพนธ์ฉบับนี้ได้นำวิธีแมชชีนเลิร์นนิ่ง (Machine Learning) โดยเลือกใช้วิธีนิวรอลเน็ตเวิร์ก (Neural Network) มาประยุกต์เพื่อนำมาเรียนรู้เฉพาะในการส่งลูกบอลในเกมฟุตบอล เนื่องจากว่าฟุตบอลเป็นเกมที่มีข้อมูลที่พิจารณาค่อนข้างมาก ซึ่งพัฒนาด้วยวิธีการสร้างกฎตายตัวได้ยาก และสำหรับเหตุผลที่ทำการเรียนรู้เฉพาะการส่งลูกบอลนั้น ก็เพื่อให้การทดลองผลการเรียนรู้ได้ง่าย

Game By Machine Learning

Rapeepun	Runtakij	42010290
Suchart	Chayangsu	42010385
Kietkul	Jearanaitanakij	Advisor

ABSTRACT

Nowadays, game development focuses on making game having appropriate abilities to players. The most widely-used technique is Rule based technique which limitation is unable to adapt its tactics. Consequently, providing learning ability in the game in order to response to players' needs would be wise.

In this thesis, we applied Machine Learning technique with Neural Network methodology to build a system that is able to learn merely ball distribution in football game. Because football game has numbers of parameters to consider, developing it using rule based technique would be highly difficult. The reason why we focused merely learning ball distribution is for the ease of learning experiment.

กิตติกรรมประกาศ

ในการทำปริญญานิพนธ์ฉบับนี้คณะผู้จัดทำขอกราบขอบพระคุณบิดามารดาที่ช่วยอบรมสั่งสอน เลี้ยงดูและส่งเสริมด้านการศึกษาหาความรู้ต่างๆ รวมถึงกำลังใจอันที่หาที่เปรียบมิได้จนกระทั่งช่วยให้คณะผู้จัดจบการศึกษาสำเร็จด้วยดี

ขอขอบพระคุณอาจารย์เกียรติกูล เจียรนัยธนะกิจ ที่คอยให้คำปรึกษาและคำแนะนำต่างๆ เกี่ยวกับปริญญานิพนธ์นี้ ทำให้ปริญญานิพนธ์ครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี ตลอดจนคณาจารย์ทุกท่านผู้ประสิทธิประสาทวิชาให้แก่คณะผู้จัดทำทุกท่าน

ขอบคุณพี่ เพื่อนๆ ทุกคนที่เอื้อเฟื้อน้ำใจในการแก้ไขปัญหาที่เกิดขึ้น และความคิดดีๆ ที่ประกอบขึ้นมาเป็นปริญญานิพนธ์นี้สำเร็จขึ้นมา

นายระพีพันธ์ รันทกิจ

นายสุชาติ ชยางศุ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันการพัฒนาเกมได้พยายามที่จะทำให้เกมมีความสามารถที่เหมาะสมกับผู้เล่น เพื่อความสนุกสนานของผู้เล่น ซึ่งโดยส่วนมากจะพัฒนาโดยใช้กฎตายตัว โดยสร้างกฎหลายชุดเพื่อที่จะให้มีความสามารถหลายระดับ แต่ความสามารถแค่นี้ไม่เพียงพอสำหรับแต่ละผู้เล่นเห็นได้การพัฒนาให้มีเกมออนไลน์ (Online Game) เพื่อเลี่ยงไปเล่นกับผู้เล่นคนอื่นแทน วิธีแก้ไขปัญหาเหล่านั้นคือ ต้องหาวิธีทำให้เกมสามารถที่จะปรับเปลี่ยนพฤติกรรมของมันเองได้

โครงงานนี้จึงได้นำวิธีการของแมชชีนเลินนิ่ง ซึ่งเป็นส่วนหนึ่งของปัญญาประดิษฐ์ (Artificial Intelligence) มาประยุกต์ใช้ในเกม โดยสามารถใช้เทคนิคเรียนรู้ (Learning) ได้แก่ นิวรอลเน็ตเวิร์ก แผนภาพต้นไม้สำหรับการตัดสินใจ (Decision Tree Learning) และอื่นๆ สำหรับในโครงงานนี้เลือกใช้แบ็คพรอพาเกชันนิวรอลเน็ตเวิร์กซึ่งเป็นนิวรอลเน็ตเวิร์กประเภทหนึ่งเป็นเครื่องมือในการปรับเปลี่ยนพฤติกรรมของการเล่นเกม

ในโครงงานได้นำวิธีแมชชีนเลินนิ่ง มาประยุกต์ใช้ในเกมฟุตบอล เนื่องจากเป็นเกมที่มีความไม่แน่นอน และไม่สามารถที่จะเขียนกฎให้ครอบคลุมเกมได้ และยังเป็นเกมที่สามารถที่จะเรียนรู้จากผู้เล่นได้ โดยนำวิธีการนี้มาใช้ในการเรียนรู้เฉพาะการส่งลูกบอล เพื่อจำกัดขอบเขตในการทดลอง และสามารถเห็นผลลัพธ์ได้อย่างเด่นชัด

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาทฤษฎี และหลักการของแมชชีนเลิร์นนิ่ง
2. เพื่อศึกษาทฤษฎี และหลักการของนิวรอลเน็ตเวิร์ก
3. เพื่อออกแบบและพัฒนาเกมฟุตบอลโดยใช้หลักการเขียนโปรแกรมเชิงวัตถุ
4. เพื่อนำทฤษฎีนิวรอลเน็ตเวิร์ก มาใช้ในแก้ปัญหาการส่งลูกบอล รวมทั้งการหารูปแบบ และวิธีการที่เหมาะสมในการนำนิวรอลเน็ตเวิร์ก ไปใช้ในการแก้ปัญหาดังกล่าว

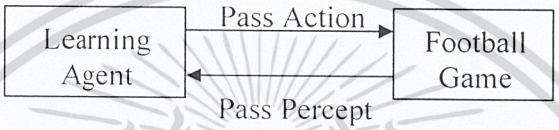
1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการเขียนเกมฟุตบอลโดยนำเอาหลักการเรียนรู้โดยใช้หลักของนิวรอลเน็ตเวิร์ก ซึ่งสามารถที่จะเรียนรู้ (Learning) โดยนำข้อมูลใช้ในการเทรนนิวรอลเน็ตเวิร์ก คือ ตำแหน่งของนักฟุตบอลทุกตัวที่อยู่ในสนามขณะที่เกิดการส่งลูกฟุตบอล ทิศทางการส่งบอลของผู้เล่น และผลการส่งลูกบอล ซึ่งจะเป็นตัวแปรที่ช่วยในการตัดสินใจว่าการส่งบอลในครั้งนั้นดีหรือไม่

โครงงานนี้เราสร้างเลือกพัฒนาเกมฟุตบอล เนื่องจากเป็นเกมที่พัฒนาด้วยวิธีกำหนดกฎได้ยาก จึงนำมาพัฒนาโดยใช้หลักการออกแบบเชิงวัตถุ เพื่อได้โปรแกรมที่ง่ายในการแก้ไข และนำกลับมาใช้ใหม่ได้ โดยที่เกมใช้เอนจินต์ที่เป็นที่นิยมในการเขียนเกมปัจจุบัน คือ ไดเรกเอ็กซ์ โดยใช้คอมโพเน็นต์

ไดเรกดรอว์ (DirectDraw) ไดเรกซาวน์ (DirectSound) และไดเรกอินพุท (DirectInput) เนื่องจากมีความนิยมแพร่หลาย สามารถพัฒนาได้ง่าย โดยไม่ต้องจัดการติดต่อกับอุปกรณ์ฮาร์ดแวร์เอง จึงลดข้อกำจัดการติดต่อกับอุปกรณ์ฮาร์ดแวร์ที่มีหลากหลายผลิตภัณฑ์ด้วยกัน

ในการประยุกต์การเรียนรู้จะเรียนรู้เฉพาะการส่งลูกบอล โดยที่ส่วนเอเจนต์ (Agent) จะมีการรับเพอเซป (Percept) ซึ่งเป็นตัวแปรต่างๆ ที่เราต้องการศึกษาจากเกม เพื่อที่จะตัดสินใจในการส่งลูกบอล และการเรียนรู้นั้นเราจะเรียนรู้จากการเก็บข้อมูลจากการเล่นครั้งก่อนๆ โดยไม่ได้ทำการเทรน (Train) แบบอินเตอร์แอ็กทีฟ เพราะว่าวิธีการเทรนนิวรอลเน็ตเวิร์คค่อนข้างใช้เวลานานมาก ไม่เหมาะสมที่เทรน ขณะที่กำลังเล่นอยู่ ดังนั้นหลังการแข่งขันจบในแต่ละรอบก็ ค่อยทำการเทรนและเก็บข้อมูลไว้ เพื่อนำข้อมูลที่เราได้ทำการเทรนไว้แล้วไปใช้ในการตัดสินใจของผู้เล่นในการแข่งขันครั้งถัดไป ซึ่งจะทำให้เปอร์เซ็นต์การส่งบอลในครั้งถัดไปดีขึ้น



รูปที่ 1-1 การตัดสินใจ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เกี่ยวกับปัญญาประดิษฐ์ โดยเฉพาะด้านแมชชีนเลิร์นนิ่ง
2. ได้รับความเข้าใจเกี่ยวกับประโยชน์ และ หลักการทำงานของนิวรอลเน็ตเวิร์ก
3. สามารถประยุกต์ใช้ นิวรอลเน็ตเวิร์ก ในการเรียนรู้ได้
4. ได้รับความรู้ ความสามารถในการพัฒนาเกม โดยใช้ ไดเรกเอ็กซ์
5. ได้รับความรู้ ความสามารถในการออกแบบโปรแกรมเชิงวัตถุ
6. สามารถวิเคราะห์และออกแบบโครงสร้างข้อมูลที่นำมาใช้ในการพัฒนาเกม

1.5 วิธีการดำเนินงาน

การดำเนินงานจะเริ่มด้วยการศึกษาทฤษฎีการเรียนรู้แบบต่างๆ ศึกษากระบวนการเรียนรู้ การนำไปประยุกต์ใช้ และเทคนิคในการพัฒนา

เมื่อศึกษาทฤษฎีที่ต้องใช้ครบถ้วนแล้วก็เริ่มพัฒนาเกมโดยทำการออกแบบและพัฒนาโดยใช้หลักการเชิงวัตถุ แล้วทำการออกแบบประยุกต์การเรียนรู้กับเกมฟุตบอล โดยมีขั้นตอนดังนี้

1. ทำการออกแบบและกำหนดโครงสร้างของเกม รวมทั้งออกแบบรูปแบบควบคุมการเคลื่อนที่ของนักฟุตบอลและการส่งบอล
2. ทำการพัฒนาเกมฟุตบอล
3. รวบรวมข้อมูลการส่งลูกบอล
4. ทดลองหาโทโปโลยีของนิวรอลเน็ตเวิร์กที่เหมาะสม
5. ทำการปรับปรุงแก้ไขตัวแปรที่ใช้ในการเทรน ให้ได้ผลลัพธ์ที่เหมาะสม

บทที่ 2
นิวรอลเน็ตเวิร์ก

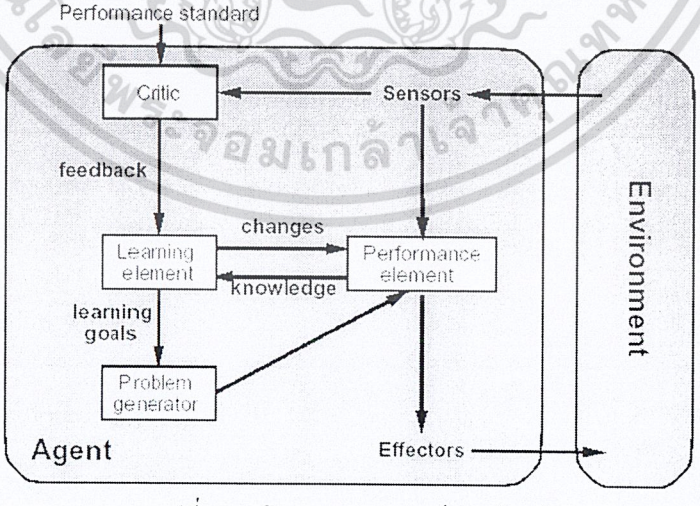
2.1 การเรียนรู้ (Learning)

โดยปกติโปรแกรมที่พัฒนาขึ้นนั้น ไม่สามารถที่จะทำงานได้เมื่อไปอยู่ในสภาพแวดล้อมที่แตกต่างกันออกไป และความสามารถของโปรแกรมเหล่านั้นจะขึ้นอยู่กับผู้พัฒนาที่ได้โปรแกรมนั้น ๆ ทำให้ไม่สามารถที่จะทำงานในสภาพแวดล้อมที่เปลี่ยนแปลงไปจากเดิมได้ ดังนั้นการเรียนรู้จึงเป็นการประยุกต์ที่ทำให้โปรแกรมเหล่านั้นสามารถที่จะเรียนรู้ และเปลี่ยนแปลงพฤติกรรมตามสิ่งแวดล้อมที่เปลี่ยนแปลงไป ซึ่งสามารถที่จะประยุกต์ใช้เพื่อทำการแยกแยะรูปแบบ และวิเคราะห์ข้อมูล ทำให้โปรแกรมมีความสามารถเพิ่มขึ้นในการทำงานในสภาวะแวดล้อมที่แตกต่างไปจากเดิม

การเรียนรู้เบื้องต้นของเอเจนต์ จะแบ่งออกได้เป็น 4 ส่วน คือ

1. ส่วนเลือกการกระทำ (Performance Element) เป็นส่วนที่เลือกการกระทำจากความรู้ที่มีอยู่เพื่อตอบสนองต่อสิ่งแวดล้อม
2. ส่วนวิเคราะห์การกระทำ (Critic) เป็นส่วนที่ทำการวิเคราะห์การกระทำต่อสิ่งแวดล้อมว่าได้ผลดีหรือผลเสีย
3. ส่วนเรียนรู้ (Learning Element) เป็นส่วนที่ทำการสร้างความรู้ที่มีโดยรับรู้จากส่วนวิเคราะห์การกระทำ เพื่อที่จะปรับปรุงความรู้ที่มี และเมื่อรู้ว่ามีข้อมูลไม่เพียงพอก็สามารถที่จะส่งสิ่งที่ต้องการไปยังส่วนสร้างปัญหา
4. ส่วนสร้างปัญหา (Problem Generation) ซึ่งเป็นส่วนที่สร้างสรรค์การกระทำใหม่ ๆ ซึ่งไม่มีความรู้มาก่อนทำให้สามารถที่จะหาความรู้ที่ดีกว่าหรือถูกต้องกว่า

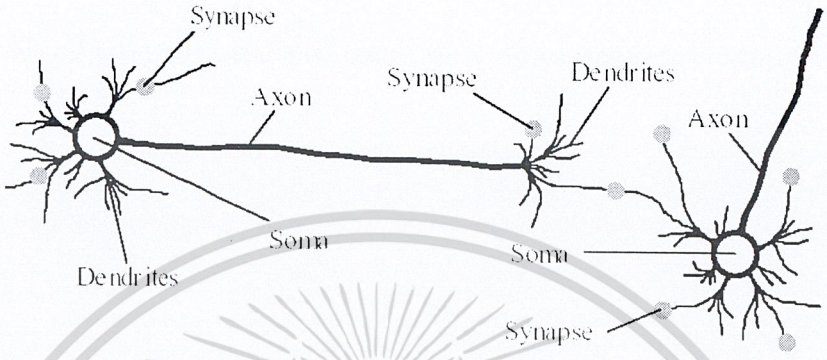
ซึ่งแสดงให้เห็นความสัมพันธ์ดังรูปที่ 2 - 1



รูปที่ 2 – 1 โมเดลของเอเจนต์การเรียนรู้

2.2 นิวรอลเน็ตเวิร์ก (Neural Network)

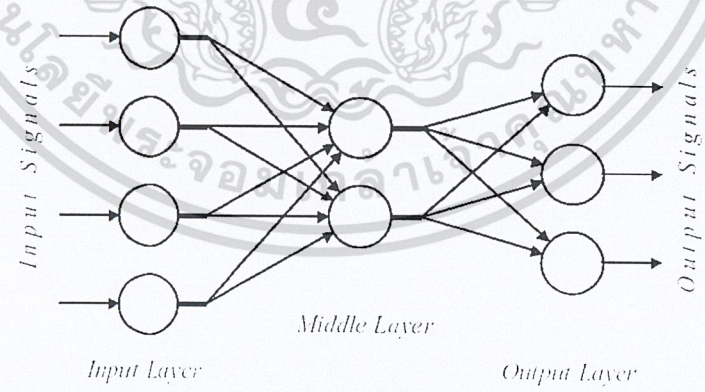
นิวรอลเน็ตเวิร์ก เป็นวิธีการเรียนรู้ที่จำลองระบบประสาทของมนุษย์ ดังรูปที่ 2 - 2 มาเป็นแบบจำลองแบบโครงข่ายประกอบด้วยยูนิต แต่ละยูนิตจะรับอินพุตหลาย ๆ ตัวเข้ามาประมวลผลได้เป็นเอาต์ -พุตค่าหนึ่ง ซึ่งจะนำเอาต์พุตเหล่านั้นส่งต่อไปยังยูนิตต่าง ๆ ดังจะเห็นได้จากรูปที่ 2 - 3



รูปที่ 2 - 2 โครงข่ายของสมองของมนุษย์

Biological Neural Network	*Artificial Neural Network*
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

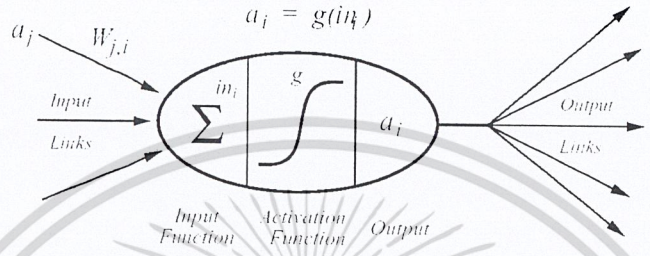
รูปที่ 2 - 3 เปรียบเทียบส่วนประกอบระหว่างระบบประสาทของมนุษย์กับที่จำลองขึ้นมาเอง



รูปที่ 2 - 4 โครงข่ายนิวรอลเน็ตเวิร์ก

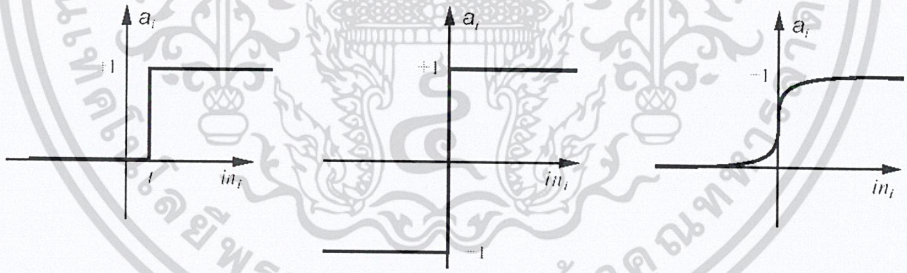
นิวรอลเน็ตเวิร์ก จะเป็นโครงข่ายของยูนิต ซึ่งแต่ละยูนิตจะรวมอินพุตต่าง ๆ เข้าไว้ด้วยกันเป็นเอาต์พุต โดยมีการคำนวณสองส่วน

ส่วนแรกคือ อินพุตฟังก์ชัน (Input Function) จะทำการรวมอินพุตต่าง ๆ เข้าไว้ด้วยกันเป็นค่าเดียว จากรูปที่ 2 - 4 จะเห็นได้ว่าอินพุตแต่ละตัวจะมีค่าถ่วงน้ำหนัก (Weight) สำหรับของแต่ละอินพุตเอง โดยอินพุตฟังก์ชันที่นิยมใช้ คือผลรวมค่าถ่วงน้ำหนัก (Weighted Sum) ซึ่งอินพุตแต่ละตัวจะถูกคูณด้วยค่าถ่วงน้ำหนักของแต่ละอินพุตเอง จากนั้นจึงนำผลคูณของแต่ละอินพุตที่ได้มาบวกกัน



รูปที่ 2 - 5 ยูนิตของนิวรอลเน็ตเวิร์ก

ส่วนที่สอง คือแอ็กทิเวชันฟังก์ชัน (Activation Function) ซึ่งจะทำหน้าที่แปลงค่าที่ได้จากอินพุตฟังก์ชันไปเป็นเอาต์พุตของแต่ละยูนิต โดยมีค่าเทรดโฮล (Thread Hole) ซึ่งเป็นตัวกำหนดจุดเริ่มต้นของแอ็กทิเวชันฟังก์ชัน ตัวอย่างแอ็กทิเวชันฟังก์ชันที่นิยมใช้ได้แก่ ซิกมอยด์ฟังก์ชัน (Sigmoid Function) และไฮเพอร์โบลิกแทนเจนท์ฟังก์ชัน (Hyperbolic Tangent Function) ดังแสดงในรูปที่ 2 - 5 และรูปที่ 2 - 6



รูปที่ 2 - 6 ตัวอย่างฟังก์ชันเบื้องต้น

$$Y^{tanh} = \frac{2a}{1+e^{-bX}} - a$$

รูปที่ 2 - 7 สมการ Hyperbolic Tangent Function

โครงข่ายนิวรอลเน็ตเวิร์ก จะเรียงยูนิตเป็นเลเยอร์ซึ่งมี 3 เลเยอร์

1. อินพุตเลเยอร์ (Input Layer) คือเลเยอร์ทางซ้ายสุดที่ต่อเชื่อมเข้ากับทุก ๆ อินพุต โดยแต่ละยูนิตในอินพุตเลเยอร์จะต่ออยู่กับแหล่งข้อมูลเพียงหนึ่งแหล่งเท่านั้น ซึ่งอินพุตเลเยอร์ไม่ได้ทำการประมวลผลใด ๆ และแต่ละอินพุตยูนิตจะนำค่าอินพุตไปแทนเป็นค่าเอาต์พุต

2. เอาต์พุตเลเยอร์ (Output Layer) คือเลเยอร์ขวาสุดที่ส่งค่าของเอาต์พุตไปยังทุก ๆ ยูนิตในเอาต์พุตเลเยอร์จะเชื่อมต่อไปยังทุกยูนิตในฮิดเดนเลเยอร์ โดยเอาต์พุตของเอาต์พุตยูนิตจะเป็นเอาต์พุตของนิวรอลเน็ตเวิร์ก

3. ฮิดเดนเลเยอร์ (Hidden Layer) คือเลเยอร์ที่อยู่ระหว่างอินพุตเลเยอร์และเอาต์พุตเลเยอร์ แต่ละ ยูนิตในฮิดเดนเลเยอร์จะเชื่อมต่อไปยังทุก ๆ ยูนิตในเลเยอร์ถัดไป ฮิดเดนยูนิตจะคำนวณหาค่าเอาต์พุตของมันโดยใช้แอ็กทิเวชันฟังก์ชัน โดยนิวรอลเน็ตเวิร์กสามารถมีฮิดเดนเลเยอร์กี่เลเยอร์ก็ได้ แต่โดยทั่วไปฮิดเดนเลเยอร์ควรมีเพียง 1 เลเยอร์ก็เพียงพอแล้ว

โทโพโลยี (Topology) ของนิวรอลเน็ตเวิร์กเบื้องต้น จะมีอยู่ 2 ชนิด ได้แก่

1. เพอเซปตรอนนิวรอลเน็ตเวิร์ก (Perceptron Neural Network) เป็นนิวรอลเน็ตเวิร์กที่มีฮิดเดนเลเยอร์เพียงชั้นเดียว และมีเพียงแค่ยูนิตเดียว โดยการปรับปรุงค่าถ่วงน้ำหนักไปในทางเดียวไม่มีการส่งค่าย้อนกลับมาปรับปรุงค่าถ่วงน้ำหนัก ความสามารถของเพอเซปตรอนนิวรอลเน็ตเวิร์ก ทำได้แค่เพียงแยกแยะประเภทของอินพุตเป็น 2 กลุ่มเท่านั้น

2. มัลติเลเยอร์นิวรอลเน็ตเวิร์ก (Multilayer Neural Network) เป็นนิวรอลเน็ตเวิร์กที่มีฮิดเดนเลเยอร์หลายชั้นปกติจะมี 1 - 4 ชั้น และมีการปรับค่าถ่วงน้ำหนักไปทางเดียว ไม่มีการปรับปรุงค่าถ่วงน้ำหนักย้อนกลับ

ขั้นตอนในการเทรนนิวรอลเน็ตเวิร์ก มีดังต่อไปนี้ คือ

1. การกำหนดค่าเริ่มต้น (Initialisation) เป็นขั้นตอนที่กำหนดค่าถ่วงน้ำหนัก และค่าเทรดโฮลของเน็ตเวิร์ก ซึ่งเป็นค่าที่สุ่มเลือกมาด้วยการกระจายตัวแบบสม่ำเสมอ (Uniform Distribution) ในช่วงแคบ ๆ ซึ่งนิวรอลเน็ตเวิร์กในแต่ละโทโพโลยีก็จะมีช่วงต่างกัน

2. การกระตุ้น (Activation) เป็นขั้นตอนในการคำนวณค่าเอาต์พุตของแต่ละยูนิต

3. การเทรนค่าถ่วงน้ำหนัก (Weight Training) เป็นขั้นที่ทำการปรับปรุงค่าถ่วงน้ำหนักและค่าเทรดโฮลจากค่าความผิดพลาดของเอาต์พุต

4. การทำซ้ำ (Iteration) เป็นการกลับไปทำซ้ำในขั้นตอนที่ 2 จนกระทั่งได้เอาต์พุตที่มีค่าความผิดพลาดที่สามารถยอมรับได้

ข้อดีของนิวรอลเน็ตเวิร์ก

1. สามารถจัดการกับปัญหาได้หลายประเภท เช่น ปัญหาการทำนายค่า การแยกประเภท และการแบ่งกลุ่ม เป็นต้น
2. ให้ผลลัพธ์ที่ดีได้แม้ว่าจะถูกนำไปใช้ในโดเมนของปัญหาที่ซับซ้อน
3. จัดการได้ทั้งตัวแปรที่มีค่าต่อเนื่องและไม่ต่อเนื่อง
4. มีแพ็กเกจสำเร็จรูปให้เลือกใช้มากมาย

ข้อเสียของนิวรอลเน็ตเวิร์ก

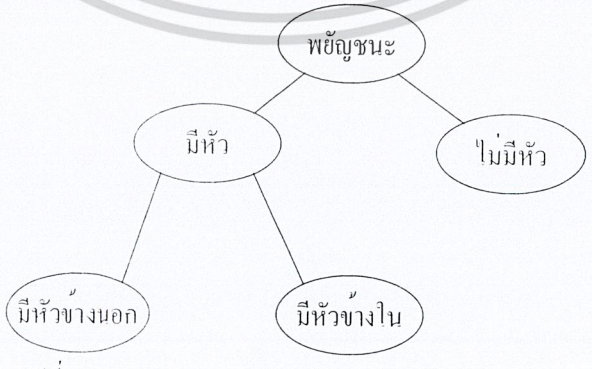
1. อินพุตและเอาต์พุตทุกตัวจะต้องถูกปรับให้อยู่ในช่วง [0, 1]
2. ไม่สามารถอธิบายที่มาของผลลัพธ์ได้
3. ผลลัพธ์ที่ได้อาจจะไม่ใช่ค่าที่ดีที่สุด

ค่าที่จำเป็นต้องเลือกเพื่อสร้างนิวรอลเน็ตเวิร์ก

1. จำนวนของฮิดเดนเลเยอร์
2. ขนาดของฮิดเดนเลเยอร์
3. ช่วงในการสุ่มค่าถ่วงน้ำหนักเริ่มต้น
4. ค่าความผิดพลาดที่ยอมรับได้

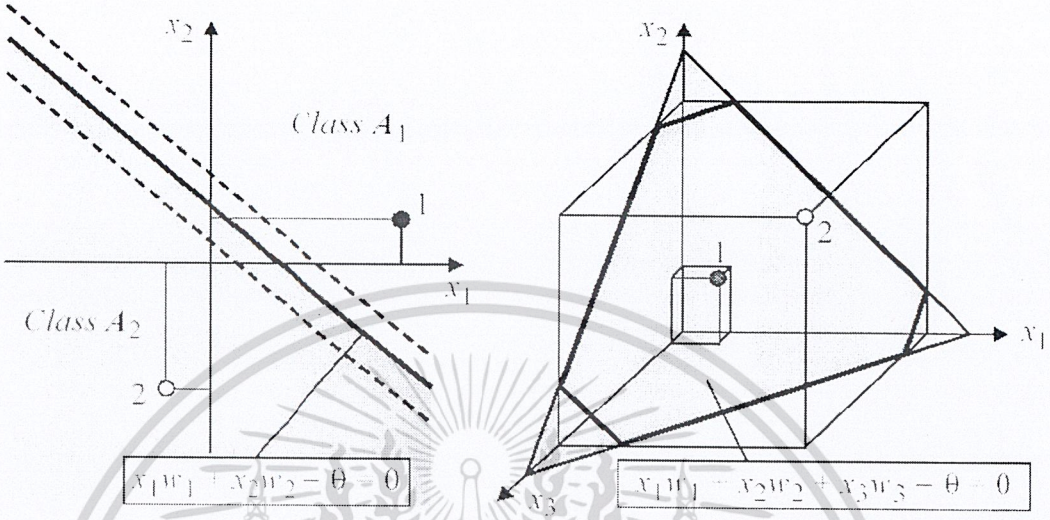
2.3 เพอเซปตรอนนิวรอลเน็ตเวิร์ก (Perceptron Neural Network)

เป็นนิวรอลเน็ตเวิร์กที่มีเพียงแค่เลเยอร์และยูนิตเดียว การปรับปรุงค่าถ่วงน้ำหนักไปในทางเดียว ไม่มีการส่งค่ายอนกลับมาปรับปรุงค่าถ่วงน้ำหนัก โดยความสามารถของเพอเซปตรอนนิวรอลเน็ตเวิร์ก คือสามารถที่จะแบ่งแยกแยะอินพุตออกเป็นประเภทต่าง ๆ ซึ่งหากมีอินพุตอยู่ เช่น $X_1, X_2, \ldots, X_n$ ดังนั้น หากใช้เพอเซปตรอนก็จะสามารถแบ่งแยกอินพุตออกเป็น 2 ข้างไปเรื่อย ๆ โดยจะเกิดรูปเหลี่ยมมีจำนวนด้านเท่ากับจำนวนอินพุต ซึ่งจะมีเส้นแบ่งแยกอินพุตออกเป็น 2 ฝั่ง ซึ่งเรียกว่า ไฮเพอร์เพลน (Hyperplane) ซึ่งเขียนอยู่ในรูปของฟังก์ชัน $\sum X_i W_i - \theta = 0$ ตัวอย่างเช่น การแบ่งแยกลักษณะรูปแบบของพยัญชนะ



รูปที่ 2 - 8 แสดงการแบ่งแยกพยัญชนะ โดยใช้เพอเซปตรอน

จากรูปมีการแบ่งแยกรูปแบบของพยัญชนะ โดยใช้ลักษณะที่แตกต่างกันของพยัญชนะเป็นเกณฑ์ โดยสามารถแบ่งแยกออกได้เป็น 2 ประเภทมีลักษณะคล้ายกับไบนารีทรี (Binary Tree) ซึ่งจะแสดงการแบ่งช่วงของอินพุตดังแสดงในรูปที่ 2 – 9



(a) Two-input perceptron. (b) Three-input perceptron.

รูปที่ 2 - 9 เส้นแบ่งอินพุตในเพอเซปตรอน (a) 2 input perceptron (b) 3 input perceptron

2.3.1 ขั้นตอนทำงานของเพอเซปตรอน

หลักการทำงานของเพอเซปตรอนจะพยายามลดความแตกต่างระหว่างค่าเอาต์พุตที่คาดหวังกับค่าเอาต์พุตจริง นั่นก็คือ พยายามจะทำการเทรนค่าถ่วงน้ำหนักไปเรื่อย ๆ จนกว่าจะได้ค่าเอาต์พุตที่ใกล้เคียงหรือเท่ากับเอาต์พุตจริง โดยจะเริ่มต้นจากการกำหนดค่าถ่วงน้ำหนักเริ่มต้นให้อยู่ในช่วง [- 0.5, 0.5] หลังจากนั้นก็จะทำการเทรน โดยการเทรนในแต่ละรอบก็จะมีค่าความผิดพลาดซึ่งสามารถคำนวณได้จากสูตร $e(p) = Y_d(p) - Y(p)$ ซึ่ง $p = 1, 2, 3, \ldots$ โดยที่ $Y_d(p)$ คือ เอาต์พุตที่คาดหวัง $Y(p)$ คือ เอาต์พุตจริงที่ออกมา p^{th} คือ ลำดับรอบของการเทรน และ $e(p)$ คือค่าความผิดพลาดในแต่ละรอบของการเทรน โดยที่หาก $e(p)$ มีค่าเป็นบวก แสดงว่าจะต้องเพิ่มค่าของ $Y(p)$ แต่หาก $e(p)$ มีค่าเป็นลบ แสดงว่าจะต้องทำการลดค่าของ $Y(p)$ และในกรณีที่ $X_i(p)$ มีค่าเป็นบวก และมีการเพิ่มขึ้นของค่าถ่วงน้ำหนักจะส่งผลให้ $Y(p)$ มีค่าเพิ่มขึ้น แต่ในทางกลับกันหาก $X_i(p)$ มีค่าเป็นลบ และมีการเพิ่มขึ้นของค่าถ่วงน้ำหนักจะส่งผลให้ $Y(p)$ มีค่าลดลง โดยค่าถ่วงน้ำหนักในรอบถัดไปสามารถคำนวณได้จากสูตร

$$W_i(p+1) = W_i(p) + \propto * X_i(p) * e(p)$$

โดยที่ $\propto$ คือค่าคงที่ของการเรียนรู้ (Learning Rate) ซึ่งค่านี้จะเหมาะสมสำหรับแต่ละปัญหา โดยจะมีการสรุปขั้นตอนการเทรนดังต่อไปนี้

1. ทำการสุ่มค่าถ่วงน้ำหนักและค่าของเทรดโฮลให้อยู่ในช่วง [- 0.5, 0.5]
2. คำนวณหาค่าเอาท์พุตจริงได้จากสูตร $Y(p) = \text{step}\,[\sum X_i(p)\, W_i(p) - \theta\,]$
3. ทำการปรับปรุงค่าถ่วงน้ำหนักได้จากสูตร

$$W_i(p+1) = W_i(p) + \alpha * X_i(p) * e(p)$$

โดยที่ $\Delta W_i(p) = \alpha * X_i(p) * e(p)$

4. ทำซ้ำโดย กลับไปทำขั้นตอนที่ 2 ไปจนกว่าจะได้ค่าถ่วงน้ำหนักที่ไม่ค่อยจะมีการเปลี่ยนแปลงแล้วจึงหยุด หรือค่าความผิดพลาดเท่ากับศูนย์ก็จะทำให้ $\alpha * X_i(p) * e(p) = 0$

เพราะฉะนั้นจะทำให้ $W_i(p+1) = W_i(p)$ คือ ค่าถ่วงน้ำหนักมีค่าคงที่ไม่เปลี่ยนแปลง

2.3.2 ตัวอย่างการนำเพอเซปตรอนไปใช้งาน

ตัวอย่างนี้คือการนำเพอเซปตรอนไปใช้ในการแก้ปัญหา AND, OR, XOR

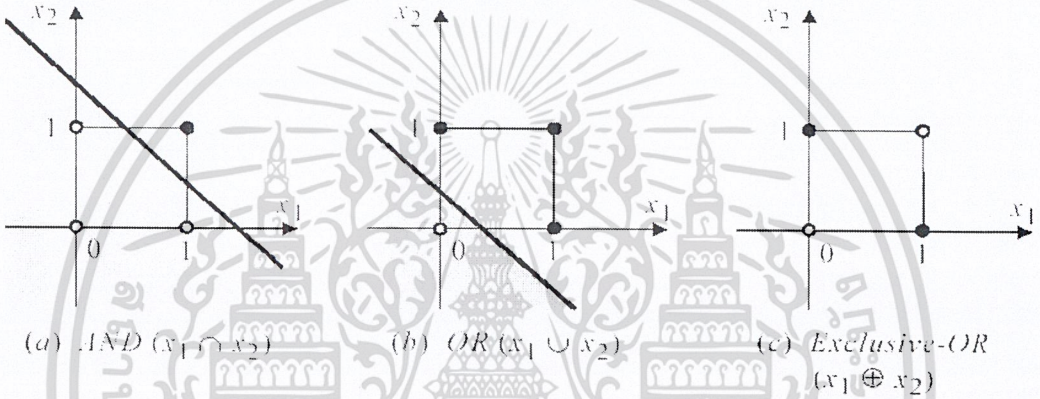
Epoch	Inputs		Desired output	Initial weights		Actual output	Error	Final weights	
	x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$

รูปที่ 2 - 10 ตารางการเทรนโอเปอเรชัน AND

จากรูปเป็นตัวอย่างแสดงรูปแบบการเทรนโอเปอเรชัน AND ซึ่งสังเกตได้ว่า หลังจากทำการเทรนในรอบสุดท้ายแล้ว ค่าความผิดพลาดเท่ากับศูนย์ แล้วได้ค่าของเอาต์พุตจริงที่ได้มีค่าเท่ากับ เอาต์พุตที่

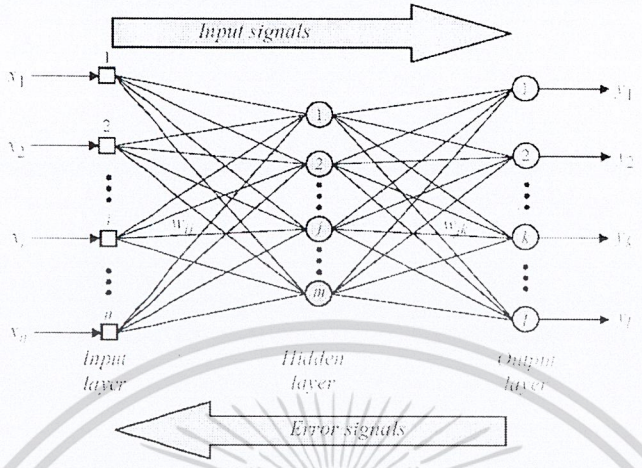
คาดหวังไว้ ซึ่งการเทรนโอเปอเรชัน AND จะมีความคล้ายคลึงกันกับโอเปอเรชันกับ OR เพราะหากสังเกตกราฟที่ได้จากรูปข้างล่างแล้ว โอเปอเรชัน AND และ OR จะได้ผลลัพธ์ของการแบ่งกราฟของเอาต์พุตออกเป็น 2 ส่วน ซึ่งแสดงว่าสามารถใช้เพอเซปตรอนนิวรอลเน็ตเวิร์กในการเทรนโอเปอเรชัน AND และ OR ได้ แต่หากจะนำไปใช้ในการเทรนโอเปอเรชัน XOR จะไม่สามารถที่จะทำได้เนื่องจากผลลัพธ์ของเอาต์พุตที่ได้จะไม่สามารถแบ่งอินพุต 2 ส่วนได้ เนื่องจากที่ถูกต้องแล้วต้องแบ่งอินพุตออกเป็น 3 ส่วน จึงไม่สามารถใช้เพอเซปตรอนมาทำการเทรนได้ เพราะเพอเซปตรอนสามารถใช้ในการแบ่งแยกอินพุตออกเพียงแค่ 2 ส่วนเท่านั้น คือสามารถใช้ได้เส้นตรงเพียงแค่เส้นเดียวในการแบ่งแยกอินพุตเท่านั้น ดังนั้นหากจะแก้ปัญหาการเทรนของ โอเปอเรชัน XOR จะต้องใช้แบ็กพรอพาเกชันมาใช้แก้ปัญหาแทนการใช้เพอเซปตรอน เนื่องจากต้องใช้เส้นตรงมากกว่า 1 เส้นมาแบ่งแยกอินพุต



รูปที่ 2 - 11 แสดงการใช้เส้นแบ่งกราฟช่วงของอินพุต

2.4 แบ็กพรอพาเกชัน (Back-Propagation Neural Network)

เป็นมัลติเลเยอร์นิวรอลเน็ตเวิร์กชนิดหนึ่งที่ได้รับความนิยมมาก ซึ่งเป็นนิวรอลเน็ตเวิร์กที่มีปรับค่าถ่วงน้ำหนักแบบย้อนกลับ หลังจากผ่านกระบวนการแอ็กทิเวชันดังรูปที่ 2 - 7



รูปที่ 2 - 12 Back - Propagation Neural Network

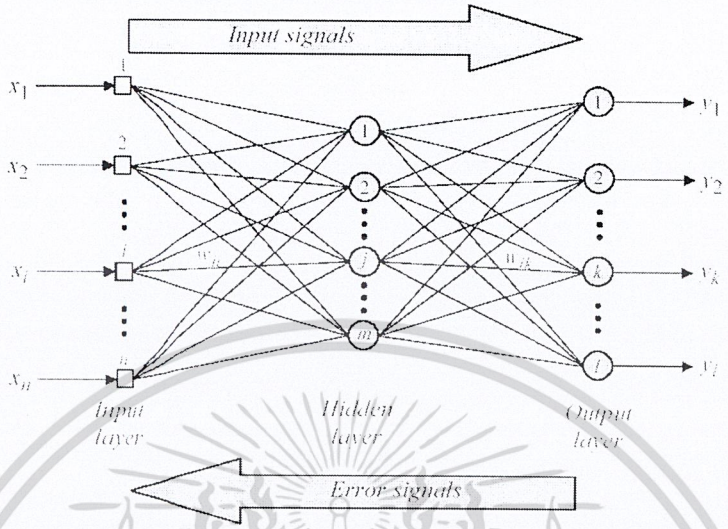
แบ็กพรอพาเกชันจะมีโครงสร้างของนิวรอลตามรูปที่ 2 - 12 โดยจะแบ่งออกเป็นลักษณะของเลเยอร์ได้ทั้งหมด 3 เลเยอร์ คือ อินพุตเลเยอร์ ฮิดเดนเลเยอร์ และเอาต์พุตเลเยอร์ โดยในแต่ละเลเยอร์นั้นก็จะมีฟังก์ชันการทำงานที่แตกต่างกันไปในแต่ละเลเยอร์ ซึ่งเริ่มจากอินพุตเลเยอร์จะทำหน้าที่รับค่าของอินพุตมาจากภายนอก แล้วจะจัดการส่งต่อไปให้ ฮิดเดนเลเยอร์ต่อไป ซึ่งฮิดเดนเลเยอร์จะทำการซ่อนเอาต์พุตที่เกิดจากการคำนวณเอาไว้ ซึ่งเราไม่สามารถที่จะรู้ได้ โดยในฮิดเดนเลเยอร์เองก็อาจประกอบด้วยเลเยอร์อีกหลาย ๆ เลเยอร์ ซึ่งโดยทั่วไปแล้วภายในฮิดเดนเลเยอร์ จะมีเลเยอร์อยู่ 3 เลเยอร์เท่านั้น หากใช้เลเยอร์มากกว่านี้ก็ต้องมีการคำนวณเพิ่มขึ้นมากเกินไปตามไปด้วย ส่วนเอาต์พุตเลเยอร์จะทำหน้าที่ในการรับอินพุตมาจากฮิดเดนเลเยอร์คำนวณหาค่าเอาต์พุตแล้วส่งออกมา

2.4.1 วิธีการเรียนรู้ของแบ็กพรอพาเกชัน

แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กจะใช้วิธีการเรียนรู้ที่เรียกว่า Back-propagation ซึ่งวิธีจะมีการคล้าย ๆ กับเพอเซปตรอน โดยจะมีการปรับปรุงค่าถ่วงน้ำหนักหากค่าของเอาต์พุตจริงที่ออกมามีค่าไม่เท่ากับเอาต์พุตที่คาดหวังไว้ เพื่อทำให้ค่าของความผิดพลาดลดลงนั่นเอง โดยเฉพาะในเพอเซปตรอนนั้นจะมีค่าของน้ำหนักเพียง 1 ตัวเท่านั้นสำหรับต่อ 1 อินพุต และเอาต์พุต แต่ในแบ็กพรอพาเกชัน จะมีค่าถ่วงน้ำหนักอยู่หลายค่าที่ใช้คำนวณเพื่อส่งค่าออกไปยังหลายเอาต์พุต โดยในแบ็กพรอพาเกชันนี้จะมีขั้นตอนในการทำงานหลักอยู่ 2 ขั้นตอน คือขั้นตอนแรกเริ่มจากอินพุตเลเยอร์ ฮิดเดนเลเยอร์ไปเรื่อย ๆ จนถึงเอาต์พุตเลเยอร์ จากนั้นก็จะทำการหาค่าผิดพลาดโดยเปรียบเทียบระหว่างเอาต์พุตที่คำนวณได้จริงกับเอาต์พุตที่คาดหวังไว้ แล้วนำค่าความผิดพลาดนี้มาพิจารณาปรับปรุงค่าถ่วงน้ำหนักย้อนกลับมายังอินพุตเลเยอร์ ซึ่งก็จะขึ้นอยู่กับค่าความผิดพลาด โดยสูตรที่ใช้หาเอาต์พุตก็มีลักษณะคล้าย ๆ เพอเซปตรอน นั่นคือ

$$X = \sum x_i w_i - \theta$$

ส่วนฟังก์ชันการกระตุ้นที่นำมาใช้ คือซิกมอยด์ฟังก์ชัน ซึ่งเท่ากับ $Y^{sigmoid} = 1 / (1 + e^{-X})$ ฟังก์ชันจะมีผลทำให้ค่าของเอาต์พุตจะอยู่ในช่วง [0, 1] เท่านั้น



รูปที่ 2 - 13 แบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กที่มี 3 เลเยอร์

จากรูปที่ 2 - 13 จะสังเกตได้ว่า $X_1, X_2, \ldots, X_n$ คือค่าของอินพุตที่ผ่านเข้าในอินพุตเลเยอร์ $Y_1, Y_2, \ldots, Y_n$ คือค่าของเอาต์พุตที่ออกทางเอาต์พุตเลเยอร์ ส่วน W_{ij} คือ ค่าของน้ำหนักที่เชื่อมระหว่างนิวรอล i ไปยังนิวรอล j และ W_{jk} คือ ค่าของน้ำหนักที่เชื่อมระหว่างนิวรอล i ไปยังนิวรอล k

2.4.2 ขั้นตอนการเทรนของแบ็กพรอพาเกชัน

1. เริ่มต้นจากกำหนดค่าถ่วงน้ำหนัก และค่าเทรดโฮลของเน็ตเวิร์ก ซึ่งเป็นค่าที่สุ่มเลือกมาด้วยการกระจายตัวสม่ำเสมอ ในช่วงแคบ ๆ ที่อยู่ในช่วง $(- 2.4 / F_i, 2.4 / F_i)$ โดยที่ F_i คือ จำนวนอินพุตในชั้นอินพุตเลเยอร์ของนิวรอลเน็ตเวิร์ก

2. ทำการกระตุ้นนิวรอลโดยเริ่มจากอินพุตในชั้นอินพุตเลเยอร์ ไปจนถึง เอาต์พุตที่คาดหวังในชั้น เอาต์พุตเลเยอร์โดยเริ่มทำตามขั้นตอนต่อไปนี้

2.1 คำนวณหาเอาต์พุตจริงออกมาจากนิวรอลในชั้น ฮิดเดนเลเยอร์ จากสมการ

$$y_j(p) = \text{sigmoid} [\sum x_i(p) * w_{ij}(p) - \theta_j]$$

2.2 คำนวณหาเอาต์พุตจริงออกมาจากนิวรอลในชั้น เอาต์พุตเลเยอร์ จากสมการ

$$y_k(p) = \text{sigmoid} [\sum x_j(p) * w_{jk}(p) - \theta_k]$$

3. ทำการปรับค่าถ่วงน้ำหนักโดย ทำการปรับจากค่าความผิดพลาดในชั้นเอาต์พุตเลเยอร์ แล้วทำการปรับมาจนถึงฮิดเดนเลเยอร์ โดยเริ่มจากขั้นตอนต่อไปนี้

3.1 คำนวณหาค่า Error Gradient ในชั้นเอาต์พุตเลเยอร์

$$\delta_k(p) = y_k(p) * [1 - y_k(p)] * e_k(p)$$

โดยที่ $e_k(p) = y_{d.k}(p) - y_k(p)$

จากนั้นก็คำนวณค่าถ่วงน้ำหนักที่เปลี่ยนไป

$\Delta W_{jk}(p) = \alpha * y_j(p) * \delta_k(p)$

ปรับเปลี่ยนค่าถ่วงน้ำหนักก่อนเอาต์พุตเลเยอร์ในรอบถัดไป

$W_{jk}(p + 1) = W_{jk}(p) + \Delta W_{jk}(p)$

3.2 คำนวณหาค่า Error Gradient ในชั้นฮิดเดนเลเยอร์

$\delta_j(p) = y_j(p) * [1 - y_j(p)] * \sum \delta_k(p) * W_{jk}(p)$

จากนั้นก็คำนวณค่าถ่วงน้ำหนักที่เปลี่ยนไป

$\Delta W_{ij}(p) = \alpha * y_i(p) * \delta_j(p)$

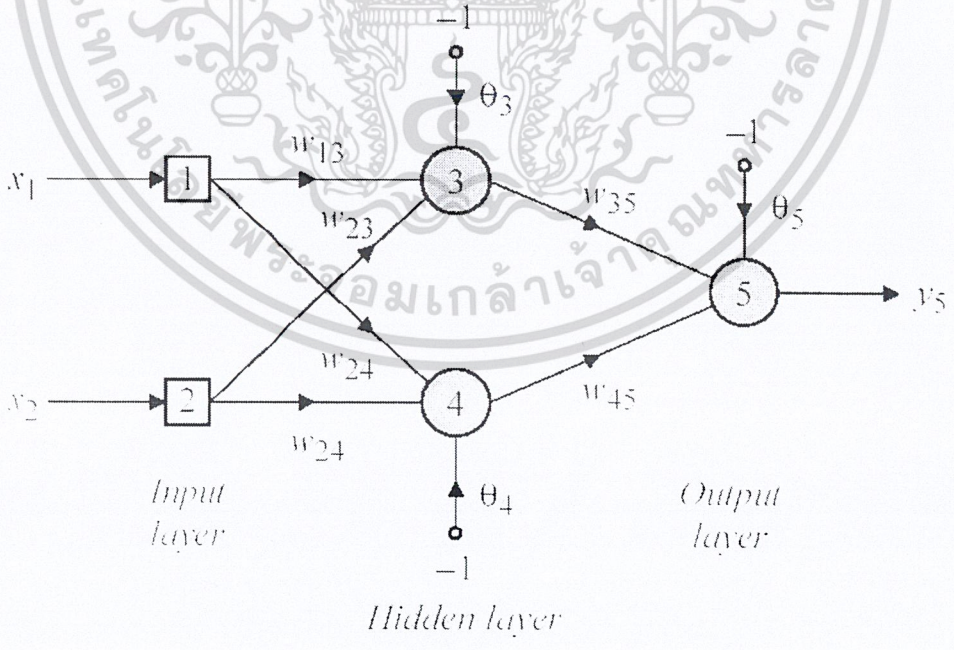
ปรับเปลี่ยนค่าถ่วงน้ำหนักก่อนเอาต์พุตเลเยอร์ในรอบถัดไป

$W_{ij}(p + 1) = W_{ij}(p) + \Delta W_{ij}(p)$

4. กลับไปทำซ้ำในขั้นตอนที่ 2 ไปเรื่อย ๆ จนกว่าจะได้ค่าความผิดพลาดที่ยอมรับได้ หรือว่าทำซ้ำจนกว่าค่าถ่วงน้ำหนักไม่มีการเปลี่ยนแปลงนั่นเอง

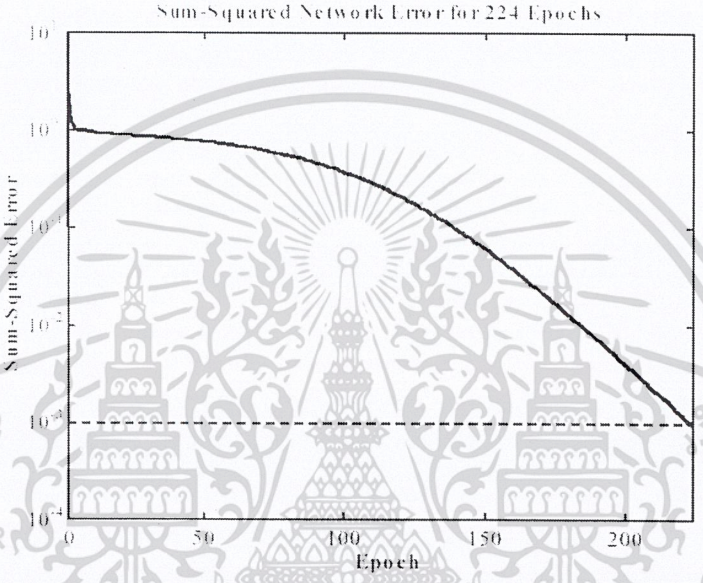
2.4.3 ตัวอย่างการนำแบ็กพรอพาเกชันไปใช้งาน

จากตัวอย่างการใช้งานของเพอเซปตรอน เคยกล่าวไว้ว่าสามารถใช้เพอเซปตรอนมาใช้ในการแก้ปัญหาโอเปอเรชัน AND และ OR ได้แต่ไม่สามารถแก้ XOR ได้ จึงได้นำแบ็กพรอพาเกชันมาใช้ในการแก้ปัญหาแทน โดยสามารถออกแบบโครงสร้างของนิวรอลตามโครงสร้างข้างล่าง



รูปที่ 2 - 14 โครงสร้างนิวรอลของโอเปอเรชัน XOR

รูปแบบโครงสร้างของนิวรอลไม่สามารถจะบอกได้ว่าดีหรือไม่ เพราะการออกแบบโครงสร้างของนิวรอลเป็นเรื่องที่ยาก ซึ่งเป็นศาสตร์ทางศิลป์มากกว่าศาสตร์ทางวิศวกรรม จากรูป 2 - 14 เป็นโครงสร้างของนิวรอลที่ใช้แก้ปัญหาของโอเปอเรชัน XOR โดยใช้ฮิดเดนเลเยอร์เพียงเลเยอร์เดียวเท่านั้น และมีการกำหนดค่าผลรวมของกำลังสองของค่าความผิดพลาดให้มีค่าเท่ากับ 0.001 โดยค่านี้จะเป็นค่าที่บอกถึงประสิทธิภาพของการเรียนรู้ว่าเรียนรู้ได้เร็วแค่ไหน ซึ่งจากแก้ปัญหาของโอเปอเรชัน XOR โดยทำการเทรนไปเรื่อย ๆ ก็จะได้กราฟของการเรียนรู้ดังข้างล่าง ซึ่งในการแก้ปัญหานี้ต้องทำการเทรนทั้งหมดจำนวน 204 รอบดังรูป 2 – 15



รูปที่ 2 - 15 กราฟแสดงการเรียนรู้ของการเทรนโอเปอเรชัน XOR

2.5 การปรับปรุงให้สามารถเรียนรู้ได้เร็วขึ้น

วิธีที่จะปรับปรุงเพื่อทำให้เรียนรู้ได้รวดเร็วมีขั้นตอนอยู่ 2 ขั้นตอนคือ

1. เปลี่ยนฟังก์ชันกระตุ้นที่จากเดิมเราจะใช้ซิกมอยด์ฟังก์ชัน ให้เปลี่ยนไปเป็นฟังก์ชันไฮเพอร์โบลิกแทนเจนท์ฟังก์ชันแทน ซึ่งสามารถเขียนให้อยู่ในรูปแบบสมการดังนี้

$$Y^{\tan h} = 2a \ / (1 + e^{-bX}) - a$$

โดยที่ a และ b เป็นค่าคงที่มีค่าเท่ากับ 1.716 และ 0.667 ตามลำดับ

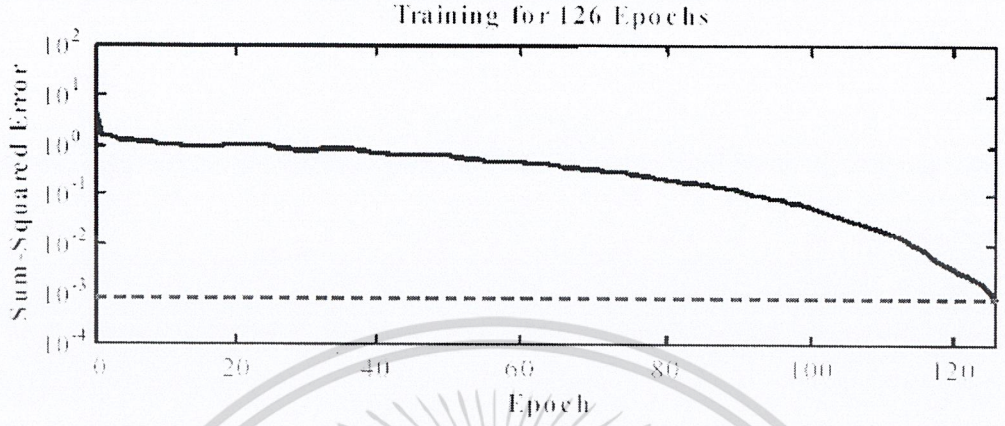
2. เปลี่ยนแปลงวิธีการคำนวณ ค่าถ่วงน้ำหนักที่เปลี่ยนไป โดยมีการเพิ่มโมเมนตัมเทอม (Momemtum Term) ไปคำนวณด้วย โดยโมเมนตัมเทอม คือ

$$\beta * \Delta W_{jk}(p-1)$$

ดังนั้นสมการใหม่จึงอยู่ในรูป

$$\Delta W_{ik}(p) = \beta * \Delta W_{jk}(p-1) + \alpha * y_i(p) * \delta_k(p)$$

เรียกว่า Genelised Delta Rule โดยที่ β จะอยู่ในช่วง [0, 1) และหาก $\beta = 0$ แสดงว่าโมเมนตัมเทอมก็จะไม่มีทำให้สมการเหมือนกับสมการคำนวณค่าถ่วงน้ำหนักที่เปลี่ยนไปตัวเดิมนั่นเอง



รูปที่ 2 - 16 กราฟปรับปรุงความเร็วเรียนรู้ของการเทรนโอเปอเรชัน XOR

บทที่ 3
ไดเรกดรอว์

3.1 รู้จักกับไดเรกดรอว์

ไดเรกดรอว์เป็นคอมโพเน็นต์ของไดเรกเอ็กซ์ที่ทำให้ผู้เขียนโปรแกรมสามารถจัดการหน่วยความจำแสดงผล และฮาร์ดแวร์ Blitter (ส่วนประกอบฮาร์ดแวร์ที่อยู่ภายในการ์ดแสดงผล ทำหน้าที่ในการ Blit หรือการถ่ายโอนบล็อกข้อมูล) ได้โดยตรง และสนับสนุนการทำ Flipping โดยจะกล่าวรายละเอียดต่อไปในบทนี้ นอกจากนี้ไดเรกดรอว์ยังคงสภาพการทำงานของแอพพลิเคชันบนระบบปฏิบัติการวินโดวส์ให้เข้ากันได้กับไดรเวอร์ของอุปกรณ์ต่าง ๆ ด้วย

ไดเรกดรอว์เป็นอินเทอร์เฟซที่ทำให้สามารถติดต่อกับอุปกรณ์แสดงผลได้โดยตรงในขณะเดียวกันยังรักษาการทำงานให้เข้ากับอินเทอร์เฟซหลักหรือจีดีไอ (GDI : Graphic Device Interface) ซึ่งใช้ในการติดต่ออุปกรณ์กราฟิกบนระบบปฏิบัติการวินโดวส์ แต่จีดีไอไม่ใช่เอพีไอสำหรับกราฟิกในระดับสูง ไดเรกดรอว์จะทำให้เกมและซอฟต์แวร์ในระบบย่อยของวินโดวส์ เช่น แพ็กเกจกราฟิก 3 มิติทั้งหลาย เครื่องแปลงสัญญาณวีดีโอดิจิตอล มีอิสระไม่ขึ้นกับอุปกรณ์บนเครื่องคอมพิวเตอร์

ไดเรกดรอว์ทำงานท่ามกลางความหลากหลายของฮาร์ดแวร์แสดงผล ที่เป็นได้ตั้งแต่จอภาพ เอสวีจีเอ (SVGA) แบบธรรมดาจนถึงฮาร์ดแวร์ระดับสูงที่สนับสนุนการทำ Clipping การทำ Stretching (การ Blit รูปไปยังปลายทางซึ่งมีมิติต่างออกไป) และรูปแบบสีที่ไม่ใช่ RGB อินเทอร์เฟซถูกออกแบบมาทำให้แอพพลิเคชันสามารถประเมินความสามารถของฮาร์ดแวร์แล้วจึงนำคุณลักษณะที่ได้รับการสนับสนุนจากฮาร์ดแวร์ตัวเร่งมาใช้ ส่วนคุณลักษณะที่ไม่ฮาร์ดแวร์ไม่สามารถทำได้จะถูกจำลองขึ้นมาโดยไดเรกเอ็กซ์

ไดเรกดรอว์ทำให้มีอิสระในการเข้าถึงหน่วยความจำแสดงผลโดยวิธีที่ไม่ขึ้นกับอุปกรณ์ ที่สำคัญเราใช้ไดเรกดรอว์ในการจัดการหน่วยความจำแสดงผล โดยที่แอพพลิเคชันต้องรู้จักเพียงข้อจำกัดของอุปกรณ์พื้นฐานบางชนิดที่เป็นมาตรฐานในการใช้งานฮาร์ดแวร์ เช่น รูปแบบสี RGB และ YUV และระยะห่างระหว่างแอดเดรสเริ่มต้นที่แสดงถึงเส้นบนจอภาพแต่ละเส้น โดยที่ผู้เขียนโปรแกรมไม่จำเป็นต้องเรียกใช้โพรซีเจอร์พิเศษเพื่อจัดการกับ Blitter หรือรีจีสเตอร์พาเลตสี เมื่อใช้ไดเรกดรอว์จะทำให้สามารถใช้หน่วยความจำแสดงผลได้อย่างง่าย รวมทั้งยังใช้ประโยชน์จากการทำ Blitting และความสามารถในการแปลงสีให้เข้ากับฮาร์ดแวร์แสดงผล โดยไม่ขึ้นกับฮาร์ดแวร์ส่วนใดส่วนหนึ่งโดยเฉพาะ

ไดเรกดรอว์ทำให้เกิดเกมกราฟิกชั้นนำบนเครื่องคอมพิวเตอร์ระบบวินโดวส์ 95 และวินโดวส์รุ่นหลัง รวมทั้งวินโดวส์ NT เวอร์ชัน 4.0 หรือวินโดวส์ 2000

3.2 เหตุผลในการนำไดเรกดรอว์มาใช้ในการพัฒนาเกม

ไดเรกดรอว์ เป็นคอมโพเน็นต์ที่ทำให้การพัฒนาโปรแกรมกราฟิกบนวินโดวส์ทำได้อย่างมีประสิทธิภาพ

- มีอินเทอร์เฟซสำหรับใช้งานฮาร์ดแวร์แสดงผลได้โดยตรง ส่งผลให้แอพพลิเคชันแสดงผลได้ดีที่สุด เมื่อโปรแกรมทำงานในระดับ Hardware Abstraction Layer หรือ HAL ของไดเรกดรอว์
- ไดเรกดรอว์สามารถประเมินความสามารถของฮาร์ดแวร์และใช้คุณลักษณะพิเศษต่าง ๆ ของฮาร์ดแวร์เมื่อทำได้ ตัวอย่างเช่น หากการ์ดจอบนเครื่องสนับสนุนฮาร์ดแวร์ที่ทำ Blitting ไดเรกดรอว์ก็จะมอบหน้าที่การ Blit ให้กับการ์ดจอ ซึ่งจะทำให้การแสดงผลมีประสิทธิภาพดียิ่งขึ้น นอกจากนี้ไดเรกดรอว์ยังมีระดับการทำงาน Hardware Emulation Layer ซึ่งจะสนับสนุนคุณลักษณะต่าง ๆ เมื่อฮาร์ดแวร์ไม่สามารถทำได้
- ไดเรกดรอว์ทำงานภายใต้ระบบปฏิบัติการวินโดวส์ จึงได้รับประโยชน์จากการอ้างอิงแอดเดรสของหน่วยความจำขนาด 32 บิตและโมเดลของหน่วยความจำแบบแฟลตซึ่งระบบปฏิบัติการเตรียมให้ ไดเรกดรอว์จะแสดงหน่วยความจำแสดงผลและหน่วยความจำของระบบเป็นแหล่งเก็บข้อมูลรูปแบบบล็อกขนาดใหญ่ ไม่ใช่เซ็กเมนต์ขนาดเล็ก ๆ คนที่เคยใช้การอ้างอิงแอดเดรสแบบเซ็กเมนต์หรือแบบออฟเซต จะรู้คุณค่าของโมเดลหน่วยความจำแบบแฟลตนี้ได้อย่างรวดเร็ว
- ไดเรกดรอว์ทำให้การสลับหน้าจอหรือที่เรียกว่า Page Flipping ที่มีบัฟเฟอร์สำรองหรือ Back Buffer หลายอันในแอพพลิเคชันโหมดเต็มจอสามารถทำได้โดยง่าย ดูรายละเอียดเพิ่มเติมได้จากหัวข้อ Page Flipping และ Back Buffering
- สนับสนุนการทำ Clipping ในแอพพลิเคชันทั้งในโหมดวินโดว์และโหมดเต็มจอ
- สนับสนุน Z-Buffer สามมิติ
- สนับสนุนโอเวอร์เลย์ของฮาร์ดแวร์ตัวช่วยโดยการจัดลำดับระดับความลึกหรือ Z - Ordering
- สามารถเข้าถึงฮาร์ดแวร์ที่ทำหน้าที่ Image - Stretching
- เข้าถึงพื้นที่หน่วยความจำของอุปกรณ์แสดงผลในระดับมาตรฐานและระดับที่สูงขึ้นได้ในเวลาเดียวกัน
- คุณลักษณะอื่น ๆ รวมถึงพาเลตแบบไดนามิคและแบบคัสตอม การเข้าถึงฮาร์ดแวร์เฉพาะอย่างและการสับเปลี่ยนโหมดความละเอียดหรือเรโซลูชัน

เมื่อรวมคุณลักษณะเหล่านี้ไว้ด้วยกัน จะเห็นได้ว่าเราสามารถสร้างแอพพลิเคชันที่มีประสิทธิภาพดียิ่งกว่าแอพพลิเคชันมาตรฐานที่มีรากฐานมาจากจีดีไอบนระบบปฏิบัติการวินโดวส์ทั่วไป หรือแม้แต่แอพพลิเคชันบนระบบปฏิบัติการดอสก็ตาม

3.3 ความเข้าใจพื้นฐานทางด้านกราฟิก

เนื้อหาในส่วนนี้จะอธิบายความรู้พื้นฐานที่จำเป็นต้องใช้ในการเขียนโปรแกรมด้านกราฟิกโดยใช้ไดเรกดรอว์ ซึ่งมีดังต่อไปนี้

- Device - Independent Bitmaps

- Drawing Surfaces
- Blitting
- Page Flipping และ Back Buffering
- รู้จักกับ Rectangles

3.3.1 Device - Independent Bitmaps

ระบบปฏิบัติการวินโดวส์ รวมถึงไดเรกเอ็กซ์ใช้ Device - Independent Bitmap หรือ DIB เป็นรูปแบบไฟล์กราฟิกของตนเอง DIB เป็นไฟล์ซึ่งบรรจุข้อมูลที่อธิบาย ขนาดของรูปภาพ จำนวนสีที่ใช้ ค่าที่แสดงถึงสีเหล่านั้น และข้อมูลของแต่ละ Pixel นอกจากนี้ DIB ยังประกอบด้วยค่าตัวแปรที่ใช้ไม่บ่อยนัก เช่นข้อมูลเกี่ยวกับการบีบอัดไฟล์ สีที่สำคัญบางสี (หากไม่ได้ใช้ทั้งหมดทุกสี) และขนาดจริงของรูปภาพ (ในกรณีที่ต้องการพิมพ์ภาพนั้น) โดยปกติไฟล์ DIB จะอยู่ในรูป .bmp หรือบางครั้งอาจใช้ .dib

เนื่องจาก DIB ใช้กันอย่างแพร่หลายในการเขียนโปรแกรมบนวินโดวส์ Platform SDK จึงบรรจุฟังก์ชันมากมายไว้เรียบร้อยแล้ว เราสามารถนำมาใช้กับไดเรกเอ็กซ์ได้เลย ตัวอย่างเช่น การเขียนเกมในโครงงานนี้ก็ใช้ไฟล์รูป .bmp มาใช้ในการตกแต่งพื้นผิวของวัตถุ 3 มิติภายในเกมเช่นกัน

3.3.2 Drawing Surfaces

ไดเรกดรอว์เซอร์เฟส (Drawing Surface) รับข้อมูลวีดีโอและถูกแสดงออกทางหน้าจอเป็นรูปภาพในขั้นสุดท้าย ในโปรแกรมวินโดวส์ส่วนใหญ่มักจะเข้าถึงไดเรกดรอว์เซอร์เฟส โดยใช้ฟังก์ชันของ Win32 เช่น *GetDC* ซึ่งย่อมาจาก Get Device Context หลังจากนั้นจึงจะสามารถวาดภาพลงบนหน้าจอได้ อย่างไรก็ตาม ส่วนต่าง ๆ ภายในระบบสามารถใช้ฟังก์ชันกราฟิกของ Win32 โดยผ่านทางอินเทอร์เฟซของอุปกรณ์กราฟิกที่เรียกว่าจีดีไอ ซึ่งเป็นคอมโพเน็นต์หนึ่งของระบบที่ทำให้แอพพลิเคชันต่าง ๆ บนวินโดวส์มาตรฐานสามารถแสดงผลทางหน้าจอได้

ข้อเสียของจีดีไอก็คือไม่ได้ออกแบบมาสำหรับซอฟต์แวร์มัลติมีเดียที่มีประสิทธิภาพสูง แต่ออกแบบมาสำหรับแอพพลิเคชันทางธุรกิจ เช่น เวิร์ดโปรเซสเซอร์และแอพพลิเคชันสเปรดชีตต่าง ๆ จีดีไอยังสนับสนุนเพียงการเข้าถึงวีดีโอบัฟเฟอร์ภายในหน่วยความจำของระบบ ไม่ใช่หน่วยความจำวีดีโอ และยังไม่สามารถดึงลักษณะเด่นที่มีในการ์ดวีดีโอบางชนิดมาใช้ให้เกิดประโยชน์ได้ โดยสรุปจีดีไอเหมาะสำหรับแอพพลิเคชันทางธุรกิจมากที่สุด แต่อาจมีการแสดงผลไม่เร็วพอสำหรับซอฟต์แวร์มัลติมีเดียหรือเกมต่าง ๆ

ในขณะเดียวกันไดเรกดรอว์นั้นจะทำให้เราสามารถวาดเซอร์เฟส ซึ่งแสดงถึงข้อมูลภายในหน่วยความจำวีดีโอ จึงหมายความว่าเมื่อเราสามารถใช้ไดเรกดรอว์ เราจึงสามารถเขียนข้อมูลลงบนหน่วยความจำบนการ์ดวีดีโอได้โดยตรง ส่งผลทำให้กระบวนการทางกราฟิกมีความเร็วสูงขึ้นมาก โดยเซอร์เฟสเหล่านี้จะถูกแสดงในรูปของชุดของบล็อกข้อมูลภายในหน่วยความจำ ซึ่งการเข้าถึงแอดเดรสของข้อมูลเหล่านี้สามารถทำได้โดยง่าย

เซอร์เฟสเป็นส่วนประกอบที่สำคัญมากของไดเรกดรอว์มีหน้าที่ในการจัดการกับรูปภาพและการประมวลผลต่าง ๆ

3.3.3 Blitting

คำว่า Blit เป็นคำย่อจาก “Bit Block Transfer” ซึ่งเป็นกระบวนการในการถ่ายโอนบล็อกข้อมูลจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งภายในหน่วยความจำ การ Blit มักจะใช้ในการแสดงการเคลื่อนไหวช้า

3.3.4 Page Flipping และ Back Buffering

การสลับภาพบนหน้าจอหรือการทำ Page Flipping ถือเป็นกุญแจสำคัญในการทำมัลติมีเดียภาพเคลื่อนไหว และซอฟต์แวร์เกม การทำ Page Flipping โดยซอฟต์แวร์เปรียบเสมือนการทำภาพเคลื่อนไหวจากแผ่นกระดาษ โดยกระดาษแต่ละหน้าจะมีรูปภาพที่ค่อย ๆ เปลี่ยนแปลงไปทีละเล็กน้อย ดังนั้นเมื่อเราพลิกแผ่นกระดาษเหล่านั้นอย่างรวดเร็วรูปที่เห็นจะเหมือนมีการเคลื่อนไหวด้วย

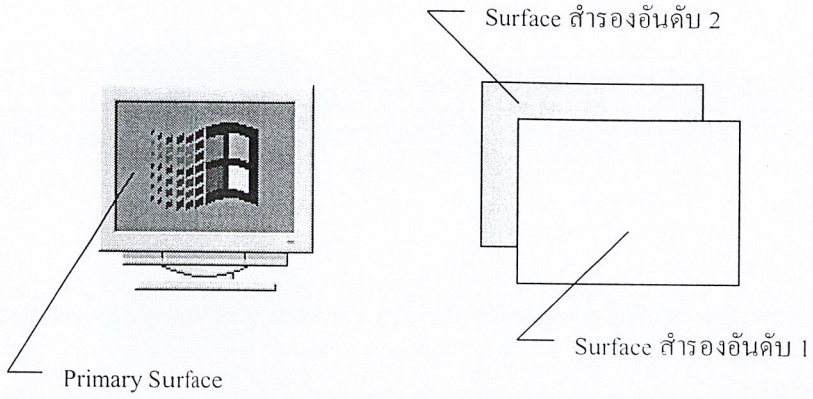
การใช้ซอฟต์แวร์ในการทำ Page Flipping จะคล้ายกับกระบวนการเช่นนี้มาก กล่าวคือ เริ่มต้นเราจะจัดเรียงไดเรกดรอว์เซอร์เฟสไว้ชุดหนึ่ง โดยออกแบบจะมีลำดับการแสดงบนหน้าจอเช่นเดียวกับชุดรูปภาพบนแผ่นกระดาษที่จะถูกพลิกเป็นลำดับถัด ๆ กันไป พื้นผิวหรือเซอร์เฟสแรกเรียกว่า เซอร์เฟสหลัก หรือ Primary Surface และ เซอร์เฟสลำดับต่อ ๆ มาเรียกว่า Back Buffer แอพพลิเคชันของเราจะเขียนหรือเปลี่ยนแปลงข้อมูลลงใน Back Buffer แล้วจึงทำการสลับกับเซอร์เฟสหลักทำให้ภาพใน Back Buffer เดิมแสดงออกสู่หน้าจอ ในขณะที่ระบบกำลังแสดงรูปภาพออกทางหน้าจอ ซอฟต์แวร์ก็จะทำการเขียนลงบน Back Buffer อีกครั้ง กระบวนการเช่นนี้จะดำเนินไปเรื่อย ๆ ในขณะที่ภาพกำลังเคลื่อนไหว ส่งผลให้เราสามารถทำภาพเคลื่อนไหวได้อย่างรวดเร็วและมีประสิทธิภาพ

ไดเรกดรอว์นั้นให้เราสามารถกำหนดกรรมวิธีในการทำ Page Flipping ได้โดยง่าย โดยสามารถกำหนดให้เป็นได้ตั้งแต่แบบพื้นฐานซึ่งใช้ 2 บัฟเฟอร์ ทำงานเกี่ยวเนื่องกัน (เซอร์เฟสหลักกับอีก 1 Back Buffer) ไปจนถึงแบบที่มีจำนวน Back Buffer เพิ่มมากขึ้น ซึ่งจะซับซ้อนมากยิ่งขึ้นเช่นกัน

โดยในการสร้างเซอร์เฟสนั้น เราจะสร้างเซอร์เฟสได้หลายแบบด้วยกัน แต่ที่ใช้กันบ่อย ๆ ก็จะสร้างไว้ 2 เซอร์เฟสดังนี้

1. เซอร์เฟสหลัก เป็นเซอร์เฟสที่กำลังแสดงอยู่บนหน้าจอปัจจุบัน
2. Other Surface หรือ Back Surface (เซอร์เฟสสำรอง) เป็นเซอร์เฟสที่เตรียมไว้สำหรับแสดงต่อไป

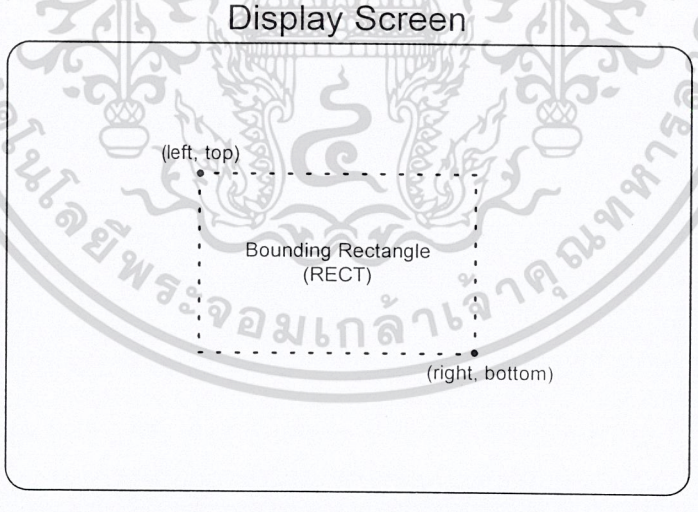
โดยจะเลื่อนขึ้นไปอีก 1 ขั้น คือ ถ้าเป็นเซอร์เฟสสำรองอันดับ 1 ก็จะเลื่อนขึ้นไปเป็นเซอร์เฟสหลักในลำดับต่อไป ส่วนเซอร์เฟสสำรองอันดับ 2 ก็จะเลื่อนไปเป็นเซอร์เฟสสำรองอันดับ 1 และ เซอร์เฟสหลักก็จะเลื่อนกลับมาเป็นเซอร์เฟสสำรองอันดับที่ 2 ในกรณีที่เราสร้างเซอร์เฟสไว้ทั้งหมดด้วยกัน 3 เซอร์เฟส ดังแสดงในรูปที่ 3 - 1



รูปที่ 3 - 1 Complex Surface

3.3.5 รู้จักกับ Rectangle

จากแนวคิดของไดเรกดรอว์และการเขียนโปรแกรมบนวินโดวส์ วัตถุต่าง ๆ บนหน้าจอจะถูกอ้างอิงในรูปกรอบสี่เหลี่ยมผืนผ้า โดยขอบของกรอบสี่เหลี่ยมผืนผ้ามักจะขนานกับขอบของหน้าจอ ดังนั้นสี่เหลี่ยมผืนผ้าดังกล่าวสามารถอธิบายได้โดยจุด 2 จุด คือมุมบน - ซ้าย และมุมล่าง - ขวา โดยแอพพลิเคชันส่วนใหญ่จะใช้โครงสร้างข้อมูล RECT ในการเก็บข้อมูลเกี่ยวกับกรอบสี่เหลี่ยมนี้ไว้ใช้เมื่อมีการ Blitting บนหน้าจอหรือการตรวจสอบการชนกันของวัตถุต่าง ๆ



รูปที่ 3 - 2 ตัวอย่างกรอบสี่เหลี่ยมผืนผ้า

ไดเรกดรอว์มีฟังก์ชัน Blitting ที่ใช้กรอบสี่เหลี่ยมในการอ้างอิงวัตถุต่าง ๆ เนื่องจากเหตุผลทางด้านประสิทธิภาพ เสถียรภาพ และความง่ายในการใช้งาน อย่างไรก็ตาม เราสามารถสร้างกระบวนการ Blit ในรูปแบบอื่นที่ไม่ใช่กรอบสี่เหลี่ยมได้เช่นกัน

3.4 สถาปัตยกรรมของไดเรกดรอว์

เนื้อหาในส่วนนี้ประกอบด้วย ข้อมูลทั่วไปเกี่ยวกับความสัมพันธ์ระหว่างคอมโพเน็นต์ไดเรกดรอว์กับคอมโพเน็นต์อื่น ๆ ของไดเรกเอ็กซ์ ระบบปฏิบัติการ และฮาร์ดแวร์ภายในระบบ

3.4.1 โครงสร้างโดยทั่วไปของไดเรกดรอว์

ซอฟต์แวร์ด้านมัลติมีเดียต้องการการแสดงผลกราฟิกในระดับสูง เมื่อเปรียบเทียบกับการใช้จีดีไอ แล้วนับได้ว่าไดเรกดรอว์ทำให้แอพพลิเคชันด้านกราฟิกบนวินโดวส์มีประสิทธิภาพและความเร็วในระดับที่สูงขึ้นกว่าเดิมมาก ในขณะเดียวกันยังคงความเป็นอิสระไม่ขึ้นกับอุปกรณ์ไว้ด้วย ไดเรกดรอว์มีเครื่องมือที่ช่วยในการแสดงงานหลักทางด้านกราฟิกต่าง ๆ ซึ่งได้แก่

- การใช้เซอร์เฟสแสดงผลหลาย ๆ เซอร์เฟส
- การเข้าถึงหน่วยความจำวีดีโอได้โดยตรง
- การทำ Page Flipping
- การจัดการพาเลต
- การทำ Clipping

นอกจากนี้ไดเรกดรอว์ยังทำให้เราสามารถดึงความสามารถของฮาร์ดแวร์แสดงผลมาใช้ในขณะรันไทม์และนำเสนอการแสดงผลที่ดีที่สุดเท่าที่จะเป็นไปได้จากความสามารถของฮาร์ดแวร์ที่อยู่บนเครื่องคอมพิวเตอร์ของผู้ใช้ซอฟต์แวร์

ในขณะที่ทำงานร่วมกับคอมโพเน็นต์อื่น ๆ ไดเรกดรอว์จะใช้ฮาร์ดแวร์ให้เกิดประโยชน์สูงสุดเท่าที่จะเป็นไปได้ และสามารถทำการจำลองทางซอฟต์แวร์สำหรับคุณลักษณะที่ดีที่สุดในกรณีที่ฮาร์ดแวร์ไม่สามารถทำได้ ความอิสระไม่ขึ้นกับอุปกรณ์เกิดขึ้นได้เมื่อใช้การทำงานระดับ Hardware Abstraction Layer หรือ HAL ซึ่งจะกล่าวรายระเอียดต่อไป

คอมโพเน็นต์ไดเรกดรอว์จัดหาบริการผ่านทางอินเทอร์เฟซที่มีแนวคิดมาจาก COM ขณะที่กำลังทำโครงงานชิ้นนี้ไดเรกดรอว์มีอินเทอร์เฟซดังนี้ IDirectDraw4, IDirectDrawSurface4, IdirectDrawPalette, IDirectDrawClipper และ IDirectDrawVideoPort

1. DirectDraw Objects มีหน้าที่หลักในการจัดการกับระบบการแสดงผล และใช้ในการติดต่อกับส่วนอื่น ๆ ในลำดับถัดไป

2. IDirectDrawSurface Objects ใช้ในการจัดการกับรูปภาพหรือข้อความที่จะทำการแสดงที่หน้าจอ โดยมีฟังก์ชันและเมธอดต่าง ๆ สนับสนุน

3. IDirectDrawPalette Objects จะช่วยในการจัดการกับพาเลต เพื่อที่จะใช้ในการแสดงโหมดสี

ของรูปภาพที่จะทำการแสดงบนจอภาพ

4. IDirectDrawClipper Objects ช่วยในการป้องกันไม่ให้รูปที่จะแสดง ออกจากขอบเขตที่กำหนดไว้ให้ ซึ่งจะใช้ในการสร้างเกมในโหมดวินโดว์

5. IDirectDrawVideoPort มีเมธอดที่ใช้ในการส่งผ่านข้อมูลวีดีโอสด ๆ จากพอร์ตวีดีโอของฮาร์ดแวร์ไปยังเซอร์เฟสของไดเรกดรอว์

3.4.2 Hardware Abstraction Layer (HAL)

ไดเรกดรอว์ทำให้แอพพลิเคชันที่สร้างขึ้นมีอิสระไม่ขึ้นกับฮาร์ดแวร์โดยการทำงานในระดับ Hardware Abstraction Layer หรือ HAL ซึ่งเป็นอินเทอร์เฟซโดยเฉพาะของอุปกรณ์ ที่ได้จากผู้ผลิตอุปกรณ์แต่ละชนิด โดยไดเรกดรอว์จะนำมาใช้ในการทำงานกับฮาร์ดแวร์แสดงผลโดยตรง แอพพลิเคชันไม่ได้ติดต่อกับ HAL แต่จะติดต่อกับโครงสร้างในระดับต่ำกว่าที่ HAL นำเสนอให้ หรือกล่าวได้ว่า ไดเรกดรอว์เป็นผู้เปิดทางให้แอพพลิเคชันได้ใช้ชุดของอินเทอร์เฟซและเมธอดที่เสถียรภาพในการแสดงผลกราฟิก ผู้ผลิตอุปกรณ์สร้าง HAL เป็นโค้ดในรูปแบบ 16 บิตรวมกับรูปแบบ 32 บิตสำหรับระบบวินโดวส์ แต่สำหรับวินโดวส์ NT นั้น HAL จะถูกใช้ในรูปแบบโค้ด 32 บิต ในที่นี้ HAL อาจเป็นส่วนหนึ่งในไดรเวอร์แสดงผล หรือ DLL ซึ่งจะติดต่อกับไดรเวอร์แสดงผลผ่านอินเทอร์เฟซที่สร้างขึ้นมาเป็นพิเศษโดยผู้ผลิตไดรเวอร์

HAL นั้นถูกสร้างขึ้นมาโดยผู้ผลิตชิป ผู้สร้างบอร์ด หรือผู้ผลิตอุปกรณ์นั้น ๆ โดย HAL เป็นเพียงโค้ดที่สร้างขึ้นเพื่อทำให้เกิดความอิสระจากอุปกรณ์ และไม่ได้แสดงผลการจำลองใด ๆ หากมีฟังก์ชันการทำงานใดที่ไม่สามารถแสดงได้โดยฮาร์ดแวร์ HAL ก็จะไม่รายงานฟังก์ชันนั้นเป็นความสามารถของฮาร์ดแวร์

3.4.3 Software Emulation

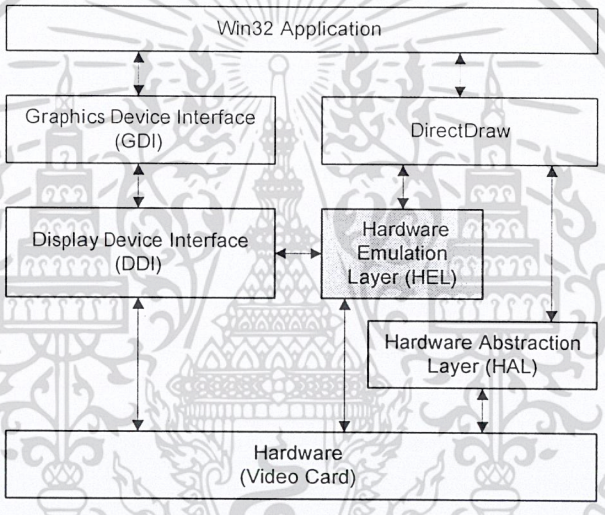
เมื่อฮาร์ดแวร์ไม่สนับสนุนคุณลักษณะผ่านทาง Hardware Abstraction Layer หรือ HAL ไดเรกดรอว์จะทำการจำลองคุณลักษณะนั้นขึ้นมา โดยฟังก์ชันที่ทำการจำลองนี้ทำได้ผ่านทาง Hardware Emulation Layer หรือ HEL โดย HEL มีการทำงานเช่นเดียวกับ HAL คือจะเสนอความสามารถของตนให้ไดเรกดรอว์และแอพพลิเคชันก็จะไม่ทำงานกับ HEL โดยตรง ผลลัพธ์ที่ได้จึงเหมือนกับแอพพลิเคชันได้รับการสนับสนุนสำหรับคุณลักษณะที่สำคัญ ๆ เกือบทุกชนิด ไม่ว่าคุณสมบัติดังกล่าวจะจะสนับสนุนโดยฮาร์ดแวร์หรือผ่านทาง HEL ก็ตาม

แน่นอนว่าการจำลองโดยซอฟต์แวร์ไม่สามารถเทียบเท่าการแสดงผลที่ได้จากคุณลักษณะต่าง ๆ ของฮาร์ดแวร์โดยตรงได้ เราสามารถร้องขอคุณลักษณะต่าง ๆ ที่ฮาร์ดแวร์สนับสนุนได้โดยใช้เมธอดของไดเรกดรอว์ (IDirectDraw :: GetCaps) ซึ่งจะตรวจสอบพิจารณาคุณลักษณะเหล่านี้ระหว่างช่วงกำหนดค่าเริ่มต้นของแอพพลิเคชัน เราสามารถปรับเปลี่ยนค่าตัวแปรต่าง ๆ ที่จะส่งผลให้การแสดงผลเป็นไปอย่างเหมาะสมที่สุดภายใต้ฮาร์ดแวร์ที่มีประสิทธิภาพแตกต่างกันไป

ส่วนใหญ่แล้ว เมื่อใช้คุณลักษณะที่สนับสนุนนโดยฮาร์ดแวร์ร่วมกับการจำลองสามารถส่งผลให้แสดงผลได้ช้ากว่าการจำลองโดยซอฟต์แวร์เพียงอย่างเดียว ตัวอย่างเช่น ถ้าหากไดรเวอร์อุปกรณ์แสดงผลสนับสนุนไดเรกดรอว์แต่ไม่สนับสนุนการทำ Stretch Blitting เราสามารถสังเกตเห็นถึงประสิทธิภาพที่สูญเสียไปเมื่อมีการ Stretching Blitting จากเซอร์เฟสบนหน่วยความจำวีดีโอ ที่เป็นเช่นนี้เพราะว่าหน่วยความจำวีดีโอมักจะช้ากว่าหน่วยความจำของระบบ ทำให้ CPU ต้องรอเมื่อมีการเข้าถึงเซอร์เฟสบนหน่วยความจำวีดีโอ ดังนั้นบางครั้งการสร้างเซอร์เฟสบนหน่วยความจำของระบบอาจเป็นสิ่งที่ดีที่สุดก็ได้

3.4.4 การทำงานรวมกันภายในระบบ

ไดอะแกรมข้างล่างแสดงถึงความสัมพันธ์ระหว่างไดเรกดรอว์, อินเทอร์เฟซของอุปกรณ์กราฟิกหรือ จีดีไอ, Hardware Abstraction Layer หรือ HAL, Hardware Emulation Layer หรือ HEL และฮาร์ดแวร์



รูปที่ 3 - 3 ความสัมพันธ์ระหว่างไดเรกดรอว์, จีดีไอ, HAL, HEL และฮาร์ดแวร์ในระบบ

จากรูปที่ 3 - 3 จะเห็นว่า ออบเจ็กต์ไดเรกดรอว์อยู่ระดับเดียวกับจีดีไอ และทั้งคู่มีการเข้าถึงฮาร์ดแวร์โดยตรงผ่านเลเยอร์หนึ่งซึ่งเป็นอิสระไม่ขึ้นกับอุปกรณ์ แต่ไดเรกดรอว์ต่างจากจีดีไอ คือจะใช้ประโยชน์จากคุณลักษณะพิเศษของฮาร์ดแวร์เมื่อใดก็ตามที่เป็นไปได้ หากฮาร์ดแวร์ไม่สนับสนุนคุณลักษณะใดไดเรกดรอว์จะพยายามจำลองคุณลักษณะนั้นขึ้นมาโดยใช้ HEL นอกจากนี้ไดเรกดรอว์สามารถนำเสนอหน่วยความจำของเซอร์เฟสในรูปแบบ Device Context ซึ่งจะทำให้เราสามารถใช้ฟังก์ชันต่าง ๆ ของจีดีไอในการทำงานกับออบเจ็กต์เซอร์เฟสได้

บทที่ 4
แนวคิดและการออกแบบ

4.1 แนวความคิดเบื้องต้น

ในโครงงานนี้เราจะเริ่มต้นจากการออกแบบ โดยจะแบ่งคลาสออกเป็น 2 ส่วน โดยในส่วนแรกจะเป็นคลาสที่จัดการเกี่ยวกับโครงสร้างของเกมอาทิเช่น กราฟฟิกการแสดงผลของเกม, รูปแบบการเคลื่อนไหวของผู้เล่น, รูปแบบการส่งบอลของผู้เล่น, กฎเกณฑ์ต่าง ๆ ภายในเกม, ส่วนที่ทำหน้าที่ติดต่อกับแมชชีนเลิร์นนิ่ง เป็นต้น และในส่วนที่ 2 คือคลาสที่จัดการเกี่ยวกับแมชชีนเลิร์นนิ่ง เช่น ส่วนเกี่ยวกับนิวรอลเน็ตเวิร์ก ส่วนที่จัดการเชื่อมระหว่างนิวรอลเน็ตเวิร์กกับตัวเกม โดยอาศัยหลักการของการเขียนโปรแกรมในเชิงวัตถุหรือ OOP (Object - Oriented Programming) เพื่อทำให้สามารถที่จะประกอบเข้าด้วยกันได้ง่ายยิ่งขึ้น

4.2 คลาสไดอะแกรมที่เกี่ยวข้องเกมฟุตบอล

รูปที่ 4 - 1 Class diagram ของเกมฟุตบอล

4.2.1 คลาส CFootballApp

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับการทำงานของแอพลิเคชันบนวินโดว์ ซึ่งประกอบด้วย ประกอบด้วยการสร้างหน้าต่าง (Create Window) รีจิสเตอร์คลาส (Register Class) แมสเสจลูป (Message Loop) เมนู (Menu) ทำการสร้างตัวแปรที่จำเป็นต่อการใช้งานผ่านไดเรกเอ็กซ์ และทำการเก็บค่าตัวเลือก (Option) ของเกม

โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CFootballApp() เมธรอดนี้จัดการเกี่ยวกับการกำหนดค่าเริ่มต้นของโปรแกรม และทำการโหลด (Load) ข้อมูลค่าตัวเลือกของเกม
- Create() เมธรอดนี้จัดการเกี่ยวกับการสร้างหน้าต่าง รีจิสเตอร์คลาส แสดงผลหน้าต่าง และทำการสร้างตัวแปรของไดเรกเอ็กซ์
- CreateDeviceObject() เมธรอดนี้จัดการเกี่ยวกับการสร้างตัวแปรของไดเรกเอ็กซ์ ที่ใช้ในเกมทั้งหมด ซึ่งประกอบด้วยตัวแปรของไดเรกดรอว์ ไดเรกซาวน์ ไดเรกอินพุต
- CreateDirectDraw() เมธรอดนี้จัดการเกี่ยวกับการสร้างตัวแปรของไดเรกดรอว์ และทำการสร้างเซอร์เฟสของทุก ๆ ออปเจ็คที่มีการแสดงผล
- CreateSurface() เมธรอดนี้จัดการเกี่ยวกับการสร้างเซอร์เฟสของทุก ๆ ออปเจ็คที่มีการแสดงผล
- CreateDirectInput() เมธรอดนี้จัดการเกี่ยวกับการสร้างตัวแปรของไดเร็กอินพุต และทำการสร้างตัวจับเวลา (Timer) เพื่อทำการตรวจสอบคีย์ที่กด
- ~CFootballApp() เมธรอดนี้จัดการเกี่ยวกับการทำลายสร้างตัวแปรของไดเรกเอ็กซ์ จัดเก็บข้อมูลค่าตัวเลือกของเกม
- FreeDeviceObject() เมธรอดนี้จัดการเกี่ยวกับการลบตัวแปรของไดเรกเอ็กซ์ ที่ใช้ในเกมทั้งหมด
- FreeDirectDraw() เมธรอดนี้จัดการเกี่ยวกับการลบตัวแปรของไดเรกดรอว์ และทำการลบเซอร์เฟสของทุก ๆ ออปเจ็คที่มีการแสดงผล
- FreeSurface() เมธรอดนี้จัดการเกี่ยวกับการลบเซอร์เฟสของทุก ๆ ออปเจ็คที่มีการแสดงผล
- FreeDirectInput() เมธรอดนี้จัดการเกี่ยวกับการลบตัวแปรของไดเร็กอินพุต และทำการทำลายตัวจับเวลา ที่ทำการกดคีย์
- GameLoadOption() เมธรอดนี้จัดการเกี่ยวกับการโหลดข้อมูลค่าตัวเลือกของเกม
- GameSaveOption() เมธรอดนี้จัดการเกี่ยวกับการจัดเก็บข้อมูลค่าตัวเลือกของเกม
- Run() เมธรอดนี้จัดการเกี่ยวกับ แมสเสจลูป ซึ่งทำหน้าที่ประจายแมสเสจ (Dispatch Message) การทำงานของโปรแกรม
- MsgProc() เมธรอดนี้จัดการทำงานตามแมสเสจต่าง ๆ ที่รับเข้ามา
- MenuNewGame() เมธรอดนี้จัดการเกี่ยวกับการสร้างตัวแปรของเกมเพื่อเริ่มเล่นเกมใหม่
- MenuLoadGame() เมธรอดนี้จัดการเกี่ยวกับการโหลดเกมเพื่อต้องการเล่นต่อได้

- MenuSaveGame() เมธรอดนี้จัดการเกี่ยวกับการจัดเก็บเกมเพื่อต้องการเล่นต่อได้
- MenuLoadNeural() เมธรอดนี้จัดการเกี่ยวกับการโหลดไฟล์นิวรอลเน็ตเวิร์ก
- MenuSaveNeural() เมธรอดนี้จัดการเกี่ยวกับการจัดเก็บไฟล์นิวรอลเน็ตเวิร์ก
- MenuGameOptionDlgProc() เมธรอดนี้จัดการทำงานตามแมสเสจต่าง ๆ ที่รับเข้ามา ของไดอะล็อก (Dialog) ตัวเลือกของเกม
- MenuTrainOptionDlgProc() เมธรอดนี้จัดการทำงานตามแมสเสจต่าง ๆ ที่รับเข้ามา ของไดอะล็อก (Dialog) ตัวเลือกของการเทรนนิวรอลเน็ตเวิร์ก
- MenuTrainNeuralDlgProc() เมธรอดนี้จัดการทำงานตามแมสเสจต่าง ๆ ที่รับเข้ามา ของไดอะล็อก (Dialog) การเทรนนิวรอลเน็ตเวิร์ก
- MenuTrainProcessThreadProc() เมธรอดนี้จัดการเกี่ยวกับการเทรนนิวรอลเน็ตเวิร์กโดยที่จะเป็นทำงานแบบเทรด(Thread)
- ProcessNextFrame() เมธรอดนี้จัดการเกี่ยวกับการเปลี่ยนแปลงออปเจ็คที่มีการแสดงผล ตามเวลาที่เปลี่ยนแปลงไป และทำการวาดลงหน้าจอ
- DisplayFrame() เมธรอดนี้จัดการเกี่ยวกับการวาดทุก ๆ เซอร์เฟสของทุก ๆ ออปเจ็คที่มีการแสดงผลลงหน้าจอ
- RestoreSurfaces() เมธรอดนี้จัดการเกี่ยวกับการวาดเซอร์เฟสของทุก ๆ ออปเจ็คใหม่ ซึ่งจะไม่สร้างเซอร์เฟสใหม่ขึ้นมาด้วย จะถูกเรียกใช้ในกรณีที่เปลี่ยนแปลงโหมดการแสดงผล

4.2.2 คลาส CDrawable

คลาสนี้เป็นคลาสที่เป็นต้นแบบของคลาสที่ต้องมีการวาดลงหน้าจอ
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CreateSurface() เมธรอดนี้จัดการเกี่ยวกับการสร้างเซอร์เฟสออปเจ็คนั้น ๆ
- Draw() เมธรอดนี้จัดการเกี่ยวกับการวาดเซอร์เฟสลงหน้าจอ
- ReleaseSurface() เมธรอดนี้จัดการเกี่ยวกับการวาดเซอร์เฟสใหม่ ซึ่งจะไม่สร้างเซอร์เฟสใหม่ขึ้นมาด้วย จะถูกเรียกใช้ในกรณีที่เปลี่ยนแปลงโหมดการแสดงผล
- Update() เมธรอดนี้จัดการเกี่ยวกับการเปลี่ยนแปลงออปเจ็ค ตามเวลาที่เปลี่ยนแปลงไป

4.2.3 คลาส CFootballMatch

คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับการทำงานหลักของเกม ซึ่งประกอบด้วยการสร้างตัวแปรของเกมฟุตบอล เช่น ลูกบอล และอื่น ๆ จัดการเกี่ยวกับสถานะของเกม แสดงผลพื้นสนาม ทำการกระจายการทำงานในขณะเล่นเกม และตรวจสอบกฎต่าง ๆ ภายในเกม เช่น ตรวจสอบการที่ลูกออกนอกสนาม ตรวจสอบหาผู้ครอบครองบอล
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- SetTeam() เมธรอดนี้จัดการเกี่ยวกับการกำหนดทีม
- SetControlTeam() เมธรอดนี้จัดการเกี่ยวกับการกำหนดทีมที่ผู้เล่นควบคุม
- SetDefendSide() เมธรอดนี้จัดการเกี่ยวกับการทีมที่ทำการรับ
- SetMatchState() เมธรอดนี้จัดการเกี่ยวกับการกำหนดสถานะของเกม
- InitGame() เมธรอดนี้จัดการเกี่ยวกับการติดตั้งค่าก่อนเล่นเริ่มเกม
- SetKickoff() เมธรอดนี้จัดการเกี่ยวกับการกำหนดตำแหน่งผู้เล่นก่อนการเตะ
- SetCorner() เมธรอดนี้จัดการเกี่ยวกับการกำหนดตำแหน่งเตะมุม
- SetFreeKick() เมธรอดนี้จัดการเกี่ยวกับการกำหนดตำแหน่งเตะ
- DeinitGame() เมธรอดนี้จัดการเกี่ยวกับการติดตั้งค่าหลังเล่นเริ่มเกม
- MatchAction() เมธรอดนี้จัดการทำงานเคลื่อนที่ ตรวจสอบกฎในการเล่นฟุตบอล และปรับปรุงตำแหน่งการเคลื่อนที่ของผู้เล่น
- CheckFootballRule() เมธรอดนี้จัดการเกี่ยวกับตรวจตำแหน่งของลูกบอล
- CheckHaveBallPlayer() เมธรอดนี้จัดการเกี่ยวกับตรวจสอบหาผู้ครอบครองบอล
- MakePassData() เมธรอดนี้จัดการเกี่ยวกับการเก็บค่าสิ่งแวดล้อมของการเตะ
- SwapDefendSide() เมธรอดนี้จัดการเกี่ยวกับการสลับทีมรุกและรับ
- SwapTeamSide() เมธรอดนี้จัดการเกี่ยวกับการสลับทีมซ้ายและขวา

4.2.4 คลาส CFootballScore

คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับการเก็บข้อมูลของเกม เช่น ข้อมูลการส่งบอล ผลการเตะ และอื่น ๆ รวมถึงการจับเวลา แสดงผลการแข่ง และสรุปผล ข้อมูลที่ทำการจัดการคือข้อมูลอัตราส่งบอลสำเร็จและ อัตราการครอบครองบอล
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- StartMatch() เมธรอดนี้จัดการเกี่ยวกับการกำหนดค่าเริ่มต้นของเกม และทำการเริ่มจับเวลา
- EndMatch() เมธรอดนี้จัดการเกี่ยวกับการเก็บข้อมูลการส่งบอล และหยุดการจับเวลา
- AddScore() เมธรอดนี้จัดการเกี่ยวกับการนับจำนวนประตูที่แต่ละทีมทำได้
- GetTime() เมธรอดนี้จะส่งค่าเวลาของเกม
- GetScore() เมธรอดนี้จะส่งจำนวนประตูของแต่ละทีม
- DoShoot() เมธรอดนี้จัดการเกี่ยวกับเก็บข้อมูลค่าสิ่งแวดล้อมปัจจุบันของการส่งครั้งนั้น ๆ
- DoResult() เมธรอดนี้จัดการเกี่ยวกับการตรวจสอบผลการส่ง และทำการคำนวณค่าสถิติของการส่ง และการครอบครองบอล
- GetPassRatio() จะทำการส่งค่าอัตราส่วนการส่งลูกบอลสำเร็จของแต่ละทีม
- GetProsessiveRatio() จะทำการส่งค่าอัตราส่วนการคลองบอลของแต่ละทีม
- SaveScoreLog() เมธรอดนี้จัดการเกี่ยวกับการเก็บผลการเตะทุก ๆ ครั้งไว้

- SaveDataLog() เมธรอดนี้จัดการเกี่ยวกับการจำนวนข้อมูลการส่งลูกทั้งหมดทุก ๆ ครั้งไว้
- LoadKickData() เมธรอดนี้จัดการเกี่ยวกับการโหลดข้อมูลการส่งลูกทั้งหมด รวมถึงผลการส่งด้วย เพื่อใช้ในการเทรน
- SaveKickData() เมธรอดนี้จัดการเกี่ยวกับการโหลดจัดเก็บข้อมูลการส่งลูก
- WriteOut() เมธรอดนี้จัดการเกี่ยวกับการสถานะของเกมทางด้านซ้ายมือ

4.2.5 คลาส CFootballBall

คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับการเคลื่อนที่ของลูกบอลโดยที่จะมีการเคลื่อนที่ทั้งในแนวราบ และแนวนอน ซึ่งจะมีส่วนที่จัดการแสดงผลอยู่ด้วย จัดการเกี่ยวกับการรับและส่งลูกบอล และการครองลูกบอลด้วย
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- GetLocation() เมธรอดนี้จัดการเกี่ยวกับการส่งค่าตำแหน่งปัจจุบันของบอล
- GetHeight() เมธรอดนี้จะส่งค่าความสูงของบอลจากพื้นสนาม
- GetControlBallPlayer() เมธรอดนี้จะส่งผู้เล่นที่ครอบครองบอลอยู่กลับ
- GetBallTeam() เมธรอดนี้จะส่งทีมที่ครอบครองบอลกลับ
- GetKicker() เมธรอดนี้จะส่งผู้เล่นที่ส่งบอลกลับ
- GetDestLocation() เมธรอดนี้จะส่งตำแหน่งที่ลูกบอลตกกลับ
- Shoot() เมธรอดนี้จัดการเกี่ยวกับการเตะบอล เพื่อที่จะให้บอลเคลื่อนที่ออกไป
- SetLocation() เมธรอดนี้จัดการเกี่ยวกับการกำหนดตำแหน่งของบอลเพื่อที่จะกำหนดให้อยู่ตำแหน่งต่าง ๆ เช่น ที่มุม ที่กลางสนาม

4.2.6 คลาส CFootballTeam

คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับทีมฟุตบอล การสร้าง และเพิ่มรวมถึงสถานะของทีม การแสดงผล และการจัดการการเคลื่อนที่ของผู้เล่น ให้เหมาะสมโดยที่จะให้คลาส CFootballTactic ช่วยในการหาจัดตำแหน่งของผู้เล่น
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- SetPlayer() เมธรอดนี้จัดการเกี่ยวกับการกำหนดผู้เล่นของทีม
- CreateDefaultTeam() เมธรอดนี้จัดการเกี่ยวกับการสร้างผู้เล่นเริ่มต้นให้ แก่ทีม
- GetCloserPlayer() เมธรอดนี้จัดการเกี่ยวกับการหาผู้เล่นของทีมที่ใกล้ตำแหน่งที่กำหนด
- TeamAction() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ผู้เล่นของทีม
- DoDefend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมไม่ได้ครอบครองบอล
- DoOffend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมได้ครอบครองบอลอยู่
- Shoot() เมธรอดนี้จัดการเกี่ยวกับการตัดสินใจในการส่งลูกบอล

- DoGoalKeeper() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของผู้รักษาประตู โดยเฉพาะเพื่อป้องกันการยิงประตูของอีกฝ่าย
- InitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมก่อนการแข่งขัน
- DeinitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมหลังการแข่งขัน

4.2.7 คลาส CFootballTeamComputer

คลาสนี้สืบทอดมาจากคลาส CFootballTeam เป็นคลาสที่จัดการเกี่ยวกับทีมฟุตบอล โดยที่มีโปรแกรมจัดการการเคลื่อนที่ และการตัดสินใจในการส่งลูกบอลเอง
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- DoDefend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมไม่ได้ครอบครองบอล
- DoOffend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมได้ครอบครองบอลอยู่
- Shoot() เมธรอดนี้จัดการเกี่ยวกับการตัดสินใจในการส่งลูกบอล
- TeamAction() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ผู้เล่นของทีม
- InitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมก่อนการแข่งขัน
- DeinitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมหลังการแข่งขัน

4.2.8 คลาส CFootballTeamControl

คลาสนี้สืบทอดมาจากคลาส CFootballTeam เป็นคลาสที่จัดการเกี่ยวกับทีมฟุตบอล ของผู้เล่นซึ่งต้องมีการจัดการการเคลื่อนที่ แต่ไม่มีส่วนการตัดสินใจในการส่งลูกบอล
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- DoDefend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมไม่ได้ครอบครองบอล
- DoOffend() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของทีมในขณะที่ทีมได้ครอบครองบอลอยู่
- Shoot() เมธรอดนี้จัดการเกี่ยวกับการตัดสินใจในการส่งลูกบอล
- TeamAction() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ผู้เล่นของทีม
- InitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมก่อนการแข่งขัน
- DeinitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมหลังการแข่งขัน
- SetControlPlayer() เมธรอดนี้จัดการเกี่ยวกับการกำหนดนักฟุตบอลที่ถูกผู้เล่นควบคุม
- GetControlPlayer() เมธรอดนี้จะส่งนักฟุตบอลที่ถูกผู้เล่นควบคุม
- SwapControlPlayer() เมธรอดนี้จัดการเกี่ยวกับการสลับตัวนักฟุตบอลที่ถูกผู้เล่นควบคุม

4.2.9 คลาส CFootballTeamLearning

คลาสนี้สืบทอดมาจากคลาส CFootballTeamComputer เป็นคลาสที่จัดการเกี่ยวกับทีมฟุตบอล โดยจะมีการจัดการเพิ่มเติมในส่วนที่เกี่ยวกับการเรียนรู้การส่งลูกบอล โดยจะมีการโอเวอร์ไรด์เมธรอดเกี่ยวกับการส่ง และเพิ่มเติมเมธรอดเกี่ยวกับการเรียนรู้

โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- InitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมก่อนการแข่งขัน
- DeinitialTeam() เมธรอดนี้จัดการเกี่ยวกับการทำงานของทีมหลังการแข่งขัน
- Shoot() เมธรอดนี้จัดการเกี่ยวกับการตัดสินใจในการส่งลูกบอล
- LoadData() เมธรอดนี้จัดการเกี่ยวกับการโหลดนิวรอลเน็ตเวิร์คขึ้นมาใช้งาน
- PreProcess() เมธรอดนี้จะทำจัดการเปลี่ยนแปลงข้อมูลการส่งให้อยู่ในรูปแบบของอินพุตของนิวรอลเน็ตเวิร์ค
- PreTraining() เมธรอดนี้จะทำจัดการเปลี่ยนแปลงข้อมูลการส่งให้เหมาะสมที่จะส่งให้นิวรอลเน็ตเวิร์ค

4.2.10 คลาส CKickData

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับเก็บข้อมูลการส่งที่เกิดขึ้นในเกม

4.2.11 คลาส CTrainData

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับเก็บข้อมูลในการเทรน

4.2.12 คลาส CFootballTactic

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับตำแหน่งของผู้เล่นในเกม เช่น ตำแหน่งตอนเริ่มต้นเกม ตำแหน่งเริ่มเตะบอล ตำแหน่งเตะมุม เป็นต้น

โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- Load() เมธรอดนี้จัดการเกี่ยวกับการโหลดตำแหน่งของผู้เล่นขึ้นมา
- SetChanging() เมธรอดนี้จัดการเกี่ยวกับการจัดตำแหน่งผู้เล่นเมื่อเริ่มเล่นเกม
- SetKickoff() เมธรอดนี้จัดการเกี่ยวกับการจัดตำแหน่งผู้เล่นเริ่มเล่นเตะบอล
- SetCorner() เมธรอดนี้จัดการเกี่ยวกับการจัดตำแหน่งผู้เล่นเมื่อเตะมุม
- SetFreeKick() เมธรอดนี้จัดการเกี่ยวกับการจัดตำแหน่งผู้เล่นเมื่อเตะฟรีคิก
- SetThrowIn() เมธรอดนี้จัดการเกี่ยวกับการจัดตำแหน่งผู้เล่นเมื่อเริ่มทุมบอล
- MovetoKickoff() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ของผู้เล่นไปยังตำแหน่งเริ่มเตะบอล

4.2.13 คลาส CFootballPlayer

คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับนักฟุตบอล ซึ่งประกอบด้วยการทำงานตามคำสั่งของทีม เช่น การเคลื่อนที่ การหยุด การหาทิศทาง เป็นต้น
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- DoRuns() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ผู้เล่น
- DoRunsTo() เมธรอดนี้จัดการเกี่ยวกับการเคลื่อนที่ไปที่ตำแหน่งใดตำแหน่งหนึ่ง
- DoStop() เมธรอดนี้จัดการเกี่ยวกับการหยุดการเคลื่อนที่
- CanTouch() เมธรอดนี้จัดการเกี่ยวกับการตรวจสอบความสามารถในการเข้าถึงตำแหน่งหนึ่ง
- CloserLoc() เมธรอดนี้จัดการเกี่ยวกับการตรวจสอบความสามารถในการเข้าถึงตำแหน่งหนึ่งในช่วงที่กำหนด
- GetDirectionTo() เมธรอดนี้จัดการเกี่ยวกับการหาทิศทางของนักฟุตบอลถึงตำแหน่งใด ๆ

4.2.14 คลาส CFootballGoalKeeper

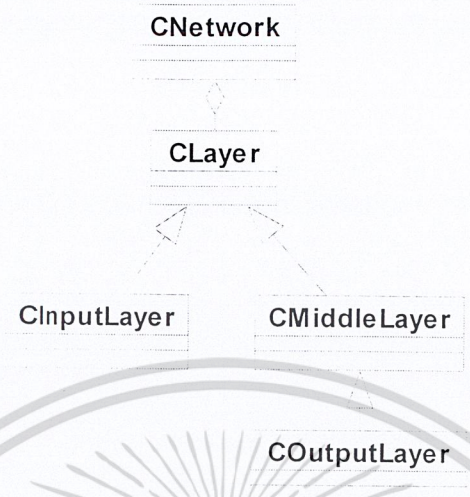
คลาสนี้สืบทอดมาจากคลาส CDrawable เป็นคลาสที่จัดการเกี่ยวกับผู้รักษาประตู
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CanTouch() เมธรอดนี้จัดการเกี่ยวกับการตรวจสอบความสามารถในการเข้าถึงตำแหน่งหนึ่ง

4.2.15 คลาส CLocation

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับการเก็บข้อมูลตำแหน่งต่าง ๆ โดยที่มีค่าเป็นจำนวนจริง และมีการจัดการเกี่ยวกับการสร้าง การเปรียบเทียบตำแหน่ง และการกลับข้างของตำแหน่งบนสนาม

4.3 คลาสไดอะแกรมที่เกี่ยวข้องกับแมชชีนเลิร์นนิ่ง



รูปที่ 4 - 2 Class diagram ของนิวรอลเน็ตเวิร์ก

4.3.1 คลาส CNetwork

คลาสนี้เป็นคลาสที่จัดการเกี่ยวกับการทำงานของนิวรอลเน็ตเวิร์กทั้งหมด เช่น การสุ่มเลือกค่าน้ำหนัก และ เทรดโฮล, การปรับปรุงค่าของน้ำหนัก (Train weight) เป็นต้น และหน้าที่อีกอย่างคือเป็นคลาสที่จัดการเกี่ยวโครงสร้างของนิวรอล โดยคลาสนี้จะประกอบด้วยคลาสเลเยอร์ต่าง ๆ เช่น CInputLayer, COutputLayer, CMiddleLayer ซึ่งคลาสนี้จะทำการสร้างโครงสร้างของนิวรอลขึ้นมาจากเลเยอร์เหล่านี้โดยเฉพาะในฮิดเดนเลเยอร์อาจจะประกอบด้วยหลายเลเยอร์ประกอบกันขึ้นมาอีกด้วย โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CreateDefaultNetwork() เมธรอดนี้จะสร้างนิวรอลเน็ตเวิร์ก ที่มี 3 เลเยอร์
- SetLayers() เมธรอดนี้จะกำหนดจำนวนเลเยอร์ของนิวรอลเน็ตเวิร์ก
- SetLayerNo() เมธรอดนี้จะกำหนดจำนวนนิวรอลของแต่ละเลเยอร์
- SetupNetwork() เมธรอดนี้จะทำการสร้าง โครงสร้างของนิวรอลขึ้น
- RandomizeWeights() เมธรอดนี้มีหน้าที่สุ่มเลือกค่าของน้ำหนัก และค่าของเทรดโฮล ของแต่ละนิวรอล
- ForwardProp() เมธรอดนี้จะการคำนวณค่าของเอาต์พุตเลเยอร์
- BackwardProp() เมธรอดนี้ก็จะทำการคำนวณค่าความผิดพลาด และค่าที่จำเป็นในการปรับปรุงค่าน้ำหนัก และเทรดโฮล
- UpdateWeights() เมธรอดนี้จะทำการเอาปรับค่าน้ำหนัก และค่าเทรดโฮล ในชั้นฮิดเดนเลเยอร์กับเอาต์พุตเลเยอร์
- UpdateMomentum() เมธรอดนี้จะทำการเก็บค่าความผิดพลาด ครั้งนี้ไว้

- GetInputLayerNode() เมธรอดนี้จะคืนค่าจำนวนของอินพุตเลเยอร์
- GetOutputLayerNode() เมธรอดนี้จะคืนค่าจำนวนของเอาต์พุตเลเยอร์
- GetLayerNo() เมธรอดนี้จะคืนค่าจำนวนของเลเยอร์ใด ๆ
- ClearPattern() เมธรอดนี้จะทำการลบข้อมูลอินพุตที่ใช้ในการเทรนทั้งหมด
- GetNumberPatts() เมธรอดนี้จะคืนค่าจำนวนของข้อมูลอินพุตในการเทรน
- SetPattern() เมธรอดนี้จะทำการเพิ่มจำนวนอิทพุทที่จะทำการเทรน
- SetInput() เมธรอดนี้จะทำการกำหนดค่าของนิวรอลของอินพุตเลเยอร์
- GetOutput() เมธรอดนี้จะทำการรับค่าของนิวรอลของเอาต์พุตเลเยอร์
- SetupPattern() เมธรอดนี้จะทำการกำหนดค่าของนิวรอลของอินพุตเลเยอร์ ด้วยอินพุต
- LoadNetwork() เมธรอดนี้จะทำการโหลดนิวรอลเน็ตเวิร์ก
- SaveNetwork() เมธรอดนี้จะทำการจัดเก็บนิวรอลเน็ตเวิร์ก

4.3.2 คลาส CInputLayer

คลาสนี้จะทำการอินเฮอร์ริทมาจากคลาส CLayer เป็นส่วนของเลเยอร์แรกสุดในนิวรอลเน็ตเวิร์ก แต่คลาสไม่สามารถนำไปใช้ได้ในทันทีเนื่องจากคลาส CLayer เป็นคลาสนามธรรมที่มีฟังก์ชันเสมือนบริสุทธิ์อยู่ ดังนั้นคลาสนี้จึงต้องทำการโอเวอร์ไรด์ฟังก์ชันนั้นเสียก่อน
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CalcOut() โอเวอร์ไรด์เฉย ๆ แต่ไม่มีการทำอะไรเหลือไว้เผื่อจะมีการทำงานทีหลัง

4.3.3 คลาส CMiddleLayer

คลาสนี้จะทำการอินเฮอร์ริทมาจากคลาส CLayer เป็นส่วนของเลเยอร์ต่อมาในนิวรอลเน็ตเวิร์ก แต่ไม่สามารถนำไปใช้งานได้ทันที ซึ่งมีเหตุผลเดียวกับคลาส CInputLayer โดยคลาสนี้จะทำการจัดการในส่วนฮิดเดนเลเยอร์ โดยต้องทำโอเวอร์ไรด์เมธรอด calculateError()ไว้
โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CalcOut() เมธรอดนี้จะทำการหาค่าเอาต์พุตของทุก ๆ นิวรอลในเลเยอร์นี้
- CalcError() เมธรอดนี้จะทำการหาค่าความผิดพลาดของนิวรอลแต่ละตัวในเลเยอร์นี้
- UpdateMomentum() เมธรอดนี้จะทำการเก็บค่าความผิดพลาด ครั้งนี้ไว้
- UpdateWeights() เมธรอดนี้จะทำการปรับค่าน้ำหนักในเลเยอร์นี้
- RandomWeights() เมธรอดนี้มีหน้าที่สุ่มเลือกค่าของน้ำหนัก และค่าของเทรดโฮล ของแต่ละนิวรอลในเลเยอร์นี้
- LoadWeights() เมธรอดนี้มีหน้าที่โหลดค่าน้ำหนัก และค่าของเทรดโฮล ของแต่ละนิวรอลในเลเยอร์นี้จากไฟล์

- SaveWeights() เมธรอดนี้มีหน้าที่จัดเก็บค่านำหนัก และค่าของเทรดโฮล ของแต่ละนิวรอล ในเลเยอร์นี้จากไฟล์

4.3.4 คลาส COutputLayer

คลาสนี้จะทำการอินเฮอร์ริทมาจากคลาส CLayer เป็นส่วนของเลเยอร์หลังสุดที่จะส่งเอาต์พุตออกมา โดยคลาสนี้มีเมธรอดที่สำคัญดังต่อไปนี้

- CalcError() เมธรอดนี้จะทำการโอเวอร์ไรด์ฟังก์ชันที่อยู่ในคลาส CMiddleLayer หาค่าความผิดพลาดของนิวรอลแต่ละตัว เพื่อที่จะนำค่านี้ไปทำการปรับเปลี่ยนค่าน้ำหนักของค่าน้ำหนักที่อยู่เลเยอร์ต่าง ๆ

4.3.5 คลาส CLayer

คลาสนี้เป็นคลาสที่ถือเป็นต้นแบบของคลาสต่าง ๆ เช่น CInputLayer, CMiddleLayer, COutputLayer โดยจะทำการสร้างฟังก์ชันเสมือนบริสุทธิ์ไว้ คือ CalcOut () เพราะว่าการกำหนดการทำงานของเมธรอดนี้ของแต่ละเลเยอร์ทำงานไม่เหมือนกัน

บทที่ 5
วิธีการเรียนรู้ของเกมฟุตบอล

ในบทนี้จะกล่าวถึงวิธีการเรียนรู้ของเกม โดยก่อนที่จะทำให้เกมสามารถเรียนรู้ได้ก็ต้องเริ่มจากการสร้างตัวเกมขึ้นมาก่อน จากนั้นก็ต้องสร้างตัวเกมให้สมบูรณ์ โดยการคิดรูปแบบการเคลื่อนที่ของผู้เล่นแต่ละตัวในเกม ซึ่งว่าส่วนนี้เป็นส่วนที่ถือว่าสำคัญมากภายในเกม เพราะว่าหากการเคลื่อนที่ของผู้เล่นไม่สมบูรณ์ มันจะเป็นการยากที่จะทำให้ ส่วนอื่น ๆ ของเกมสมบูรณ์ตามไปด้วยอย่างแน่นอน แต่หากการเคลื่อนที่ของผู้เล่นสามรถทำให้สมบูรณ์ก็ย่อมเป็น ทำให้ส่วนอื่น ๆ ของเกมสมบูรณ์มากตามไปด้วยนั่นเอง หลังจากที่จัดการเกี่ยวกับการเคลื่อนไหวสมบูรณ์ระดับหนึ่งแล้ว จากนั้นก็จะสร้างรูปแบบการส่งบอลขึ้นมา โดยเป็นลักษณะของกฎเกณฑ์ หลังจากนั้นก็ทำการสร้างส่วนของแมชชีนเลิร์นนิ่งขึ้นมาแล้วก็ทำการทดสอบ (Train) เพื่อหาค่าถ่วงน้ำหนัก รวมทั้งอินพุตที่เหมาะสม เพื่อจะนำค่าเหล่านี้ไปใช้ในการเล่นครั้งต่อไป ซึ่งโครงงานนี้จะทำการเรียนรู้เฉพาะเกี่ยวกับการส่งบอลเท่านั้น

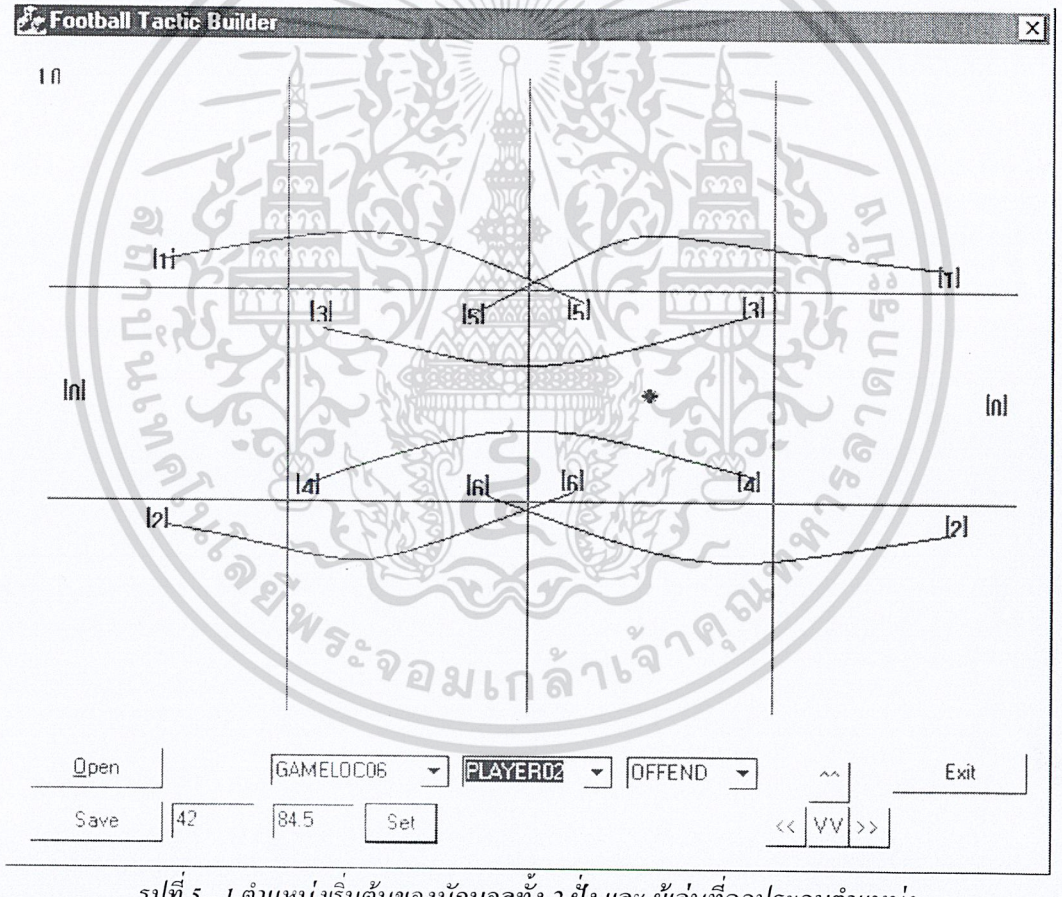
5.1 การทำงานของโปรแกรม

การทำงานของโปรแกรมจะแบ่งออกเป็น 3 ส่วนคือ ส่วนแรกคือส่วนที่เกี่ยวข้องกับการเคลื่อนที่ของนักบอล ซึ่งจะทำการตัดสินใจในการเคลื่อนที่ว่าควรจะเคลื่อนที่ไปทางใดจึงจะเหมาะสม โดยจะพิจารณาจากตำแหน่งของนักบอลของทั้ง 2 ทีม ส่วนต่อมาคือส่วนที่จะควบคุมส่งลูกบอล ซึ่งจะเป็นลักษณะของกฎเกณฑ์ที่ค่อนข้างตายตัว และส่วนสุดท้ายจะเป็นส่วนมีการนำแมชชีนเลิร์นนิ่งมาใช้ในการตัดสินใจในการส่งบอล ซึ่งการทำงานของโปรแกรมโดยรวม ในครั้งแรกก็จะเริ่มมีการเล่นเกม โดยแบ่งทีมฟุตบอลออกเป็น 2 ทีม คือ ทีมแรกคือทีมที่มีคอมพิวเตอร์ควบคุมทั้งการเคลื่อนที่และการส่งบอลทั้งคู่ ทีมที่สองคือทีมที่มีมนุษย์ควบคุมทั้งการการเคลื่อนที่ของนักบอลและการส่งบอลของนักบอล 1 คน ส่วนผู้เล่นคนอื่น ๆ จะเป็นหน้าที่ของคอมพิวเตอร์จะควบคุมการเคลื่อนที่ของนักบอลเหล่านั้นเอง ซึ่งเมื่อเริ่มแข่งขันในครั้งแรกก็จะยังก็ทำการเก็บข้อมูลตำแหน่งผู้เล่น ซึ่งในขณะนี้การวิเคราะห์การส่งบอลก็ยังคงใช้ในลักษณะของกฎเกณฑ์การส่งบอลที่ได้ค่อนข้างตายตัวแน่นอน จากนั้นเมื่อแข่งขันจบแล้วก็จะนำข้อมูลของนักบอลไปใช้ในการเทรน ซึ่งข้อมูลที่จะนำมาใช้ในการเทรนก็อาทิ เช่น ตำแหน่งของนักบอลในสนาม, ทิศทางของนักบอลในสนามขณะที่มีการส่งบอล เป็นต้น โดยหลังจากทำการเทรนเสร็จก็จะนำข้อมูลที่ได้ไปใช้แข่งขันครั้งต่อไป ซึ่งในการแข่งขันครั้งต่อไปก็จะมีการเก็บข้อมูลอีกว่าการส่งครั้งไหนส่งแล้วไม่ผิดพลาด คือไม่เสียบอลไปให้ทีมฝั่งตรงข้าม ซึ่งก็จะมีการเก็บตำแหน่งของผู้เล่นในขณะนั้นไว้ด้วย โดยจะเก็บข้อมูลอินพุต ซึ่งมีวิธีการเก็บที่ว่า หากข้อมูลนั้นมีการใช้ล่าสุดก็เก็บเอาไว้ ส่วนข้อมูลที่ไม่ได้ใช้มานานก็ลบทิ้ง (LRU : Least Recently Use) แล้วหลังจากแข่งขันจบแล้วก็จะทำการเทรนใหม่อีกครั้งเพื่อนำข้อมูลไปใช้ในครั้งต่อไป ซึ่งการวัดผลการเรียนรู้ ก็พิจารณาจากอัตราการส่งบอล คือ จะวัดผลจากอัตราการส่งบอลของทีมฝั่งที่คอมพิวเตอร์เป็นผู้ควบคุมจะมีอัตราการส่งบอลที่ค่อนข้างดีขึ้น แสดงว่า

เป็นการเรียนรู้ที่ดี แต่หากอัตราการส่งบอลเป็นไปในทางที่ตรงกันข้ามกัน แสดงว่ามีการเรียนรู้ที่ผิดพลาด ซึ่งอาจจะเกิดขึ้นได้อินพุตที่นำไปใช้ในการเทรนนั้นไม่มีผลกระทบต่ออัตราส่งบอล

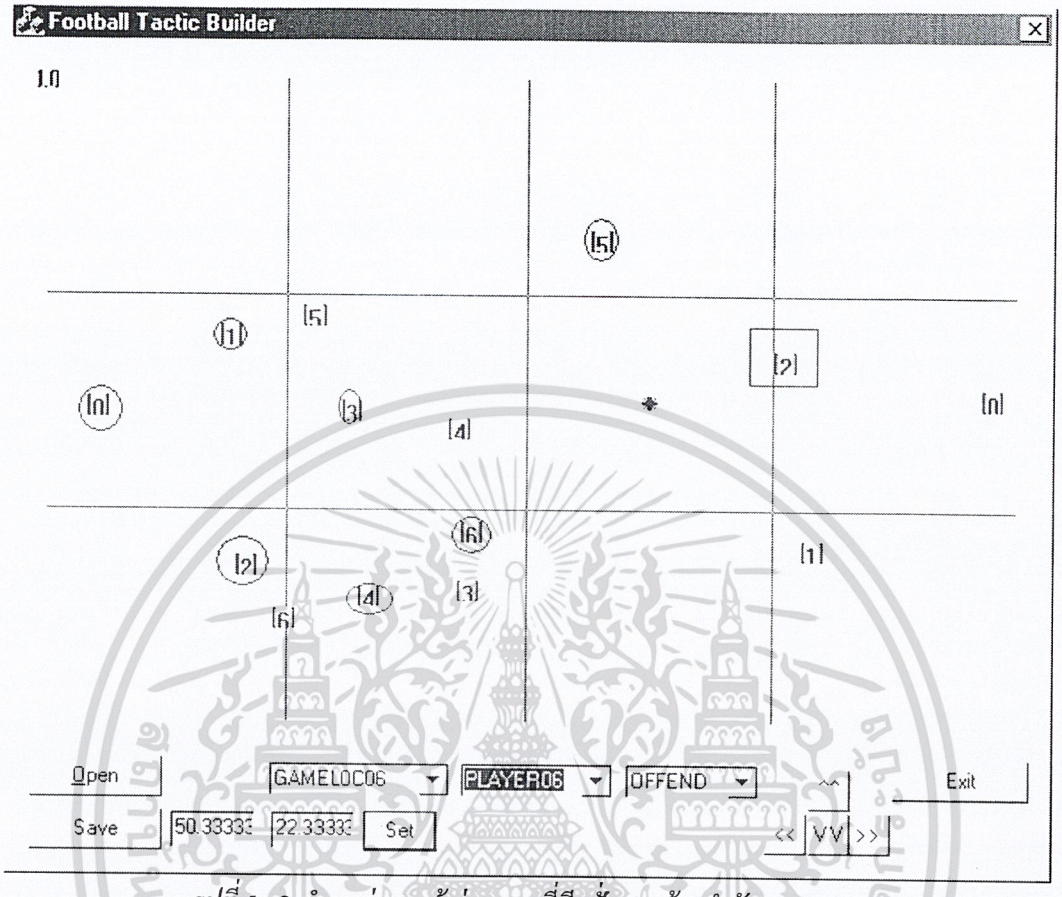
5.2 รูปแบบการเคลื่อนที่ของนักบอล

ส่วนนี้ก็จะเป็นส่วนที่ควบคุมรูปแบบการเคลื่อนที่ของนักบอลทั้ง 2 ทีม ยกเว้นมีผู้เล่นเป็นผู้ควบคุมอยู่เท่านั้นที่ ซึ่งคอมพิวเตอร์ไม่ได้ทำการควบคุมการเคลื่อนที่นักบอลคนนี้ โดยการแนวคิดการเคลื่อนที่ของนักบอลนี้ โดยก่อนอื่นจะแบ่งตำแหน่งผู้เล่นของทั้ง 2 ทีมออกเป็น 3 ตำแหน่ง คือตำแหน่งกองหน้า กองกลาง กองหลัง ซึ่งการพิจารณาที่จะวางตำแหน่งนักบอลอยู่ในตำแหน่งไหนนั้นได้มีการพัฒนาโปรแกรม เพื่อใช้สำหรับการวางแผนตำแหน่งของนักบอลในลักษณะของกฎที่ตายตัวแล้วเก็บข้อมูลส่วนนี้เพื่อนำไปใช้ในเกม โดยแบบแผนจะต้องออกเป็น 2 กรณีคือ กรณีแรกคือขณะที่ทีมเดียวกันครองบอล กรณีที่สองคือ ขณะที่ทีมฝั่งตรงข้ามเป็นผู้ครองบอลอยู่



รูปที่ 5 - 1 ตำแหน่งเริ่มต้นของนักบอลทั้ง 2 ฝั่ง และ ผู้เล่นที่ถูกประกบตำแหน่ง

5.2.1 การเคลื่อนที่ขณะผู้เล่นทีมตรงข้ามครองบอล



รูปที่ 5 - 2 ตำแหน่งของผู้เล่น ขณะที่ทีมฝั่งตรงข้ามกำลังครองบอล

จากรูปสมมติว่าผู้เล่นหมายเลข 2 ซึ่งถูกล้อมด้วยสี่เหลี่ยม กำลังครองบอลอยู่ในแดนฝั่งตรงข้าม ผู้เล่นที่วงกลมไว้ คือ ผู้เล่นที่กำลังพิจารณาและไม่ได้ครองบอล

- **ตำแหน่งกองหลัง**

ในกรณีนี้กองหลังจะทำหน้าที่ป้องกันและคอยเคลื่อนที่ติดตามกองหน้าฝั่งตรงข้าม จากรูปสังเกตได้รูปแบบการเคลื่อนที่ของผู้เล่นคือ ตำแหน่งกองหลังของทั้ง 2 ทีมก็จะคอยพิจารณาการเคลื่อนที่ของตำแหน่งกองหน้าว่ามีการเคลื่อนที่ไปในทิศทางไหน โดยหน้าที่หลักในตอนนี้ของกองหลัง คือ คอยคุมตำแหน่งให้ตำแหน่งตัวเองให้อยู่ใกล้กองหน้าที่นักบอลคนนั้นที่ตัวเองมีหน้าที่รับผิดชอบอยู่โดยจะพยายามให้ตำแหน่งตัวเองอยู่ใกล้ผู้เล่นกองหน้าคนนั้นมากกว่าผู้เล่นคนอื่น ๆ ในทีมเดียวกันมากที่สุด

จากรูปสังเกตว่าผู้เล่นหมายเลข 1 กับ 2 ซึ่งเป็นกองหลังจะคอยประกบผู้เล่นหมายเลข 5 และ 6 ซึ่งเป็นกองหน้าฝั่งตรงข้าม และผู้เล่นกองกลางหมายเลข 4 จะคอยรักษาระยะตำแหน่งให้อยู่ไม่ห่างจากหมายเลข 6 และหมายเลข 3 จะคอยรักษาระยะอยู่กลางสนามเพื่อไม่ให้กลางสนามมีพื้นว่างมากเกินไป

- **ตำแหน่งกองกลาง**

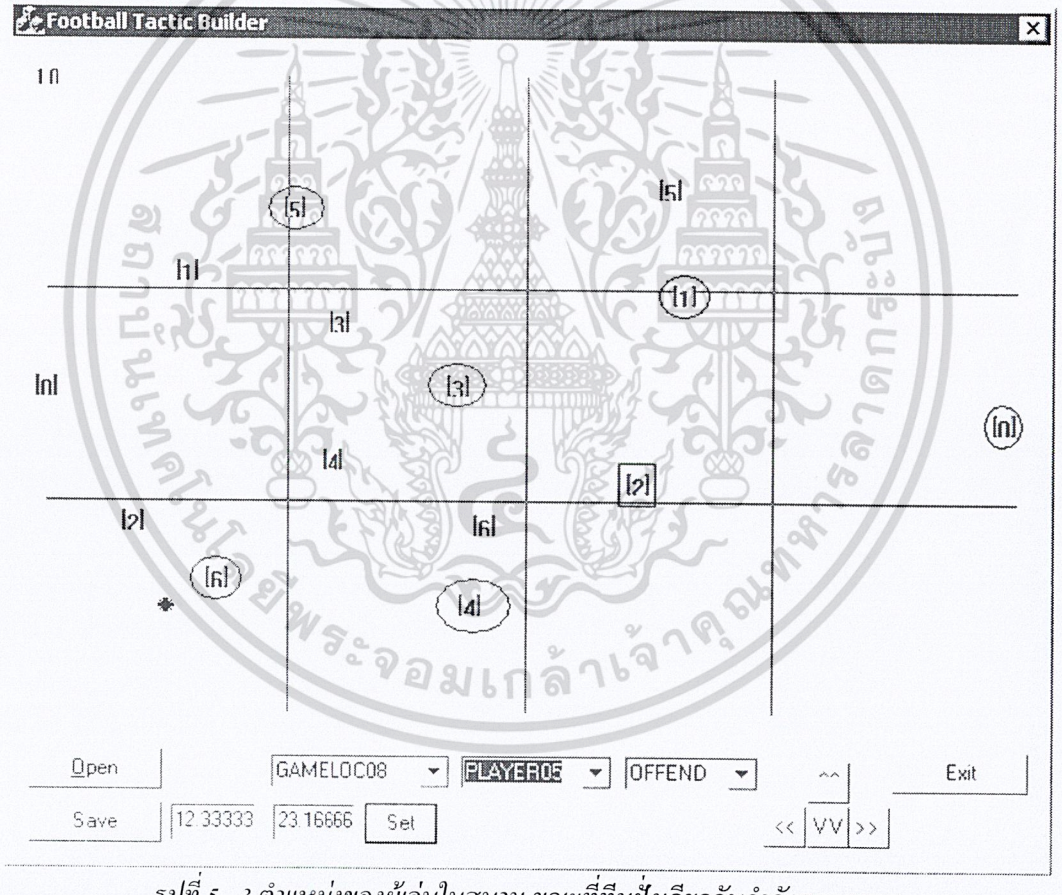
ในตำแหน่งกองกลางจะมีผู้เล่นที่จะ 1 คน นั่นคือหมายเลข 3 ที่จะอยู่กลางสนามเพื่อรับผิดชอบ

พื้นที่ในแดนกลางไม่ให้ว่างจนเกินไปอีกทั้งยังต้องพยายามรักษาระยะไม่ให้อยู่ห่างจากผู้เล่นหมายเลข 5 ของฝั่งตรงข้ามมากเกินไป ส่วนผู้เล่นกองกลางอีกคน คือ หมายเลข 4 ก็จะรักษาตำแหน่งไม่ให้อยู่ห่างหมายเลข 3 และ 6 ของฝั่งตรงข้ามมากเกินไป

- **ตำแหน่งกองหน้า**

ส่วนตำแหน่งกองหน้าก็จะพิจารณาหากลูกบอลอยู่สนามฝั่งตรงข้ามก็ให้กองหน้าพยายามเข้าไปใกล้แย่งบอลกับผู้เล่นที่มีบอลอยู่ แต่หากมีบอลอยู่ในตำแหน่งที่อยู่ในสนามของฝั่งผู้เล่นเองก็จะทิ้งกองหน้าไว้ที่ตำแหน่งของสนามฝั่งตรงกันข้ามเพียง 1 คน แล้วดึงกองหน้าคนที่อยู่ในสนามฝั่งตัวเองลงมาอยู่ในตำแหน่งกองกลาง ดังนั้นจากรูปที่ 5 - 2 ก็จะทิ้งผู้เล่นหมายเลข 5 ไว้ฝั่งตรงข้ามเพียงคนเดียวหากบอลอยู่ในสนามฝั่งตัวเอง

5.2.2 การเคลื่อนที่ขณะผู้เล่นทีมเดียวกันครองบอล

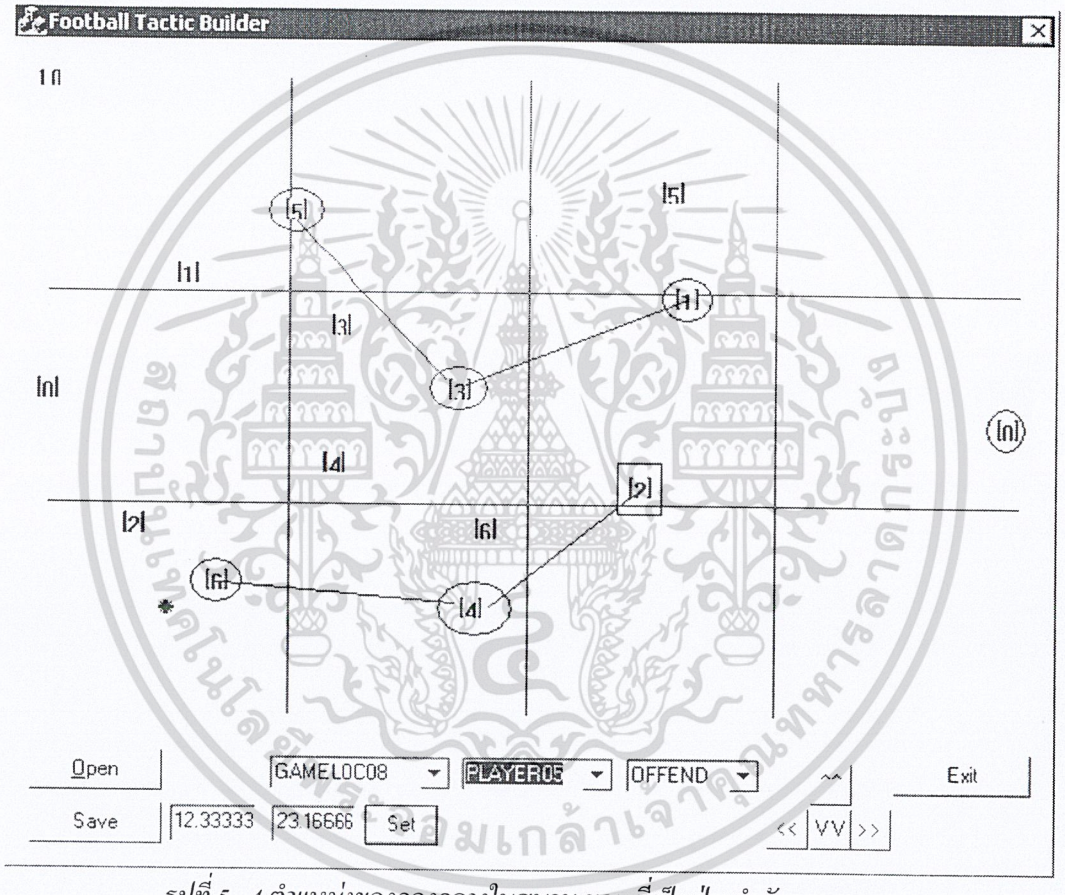


รูปที่ 5 - 3 ตำแหน่งของผู้เล่นในสนาม ขณะที่ทีมฝั่งเดียวกันกำลังครองบอล

จากรูปสมมติว่าผู้เล่นหมายเลข 2 ซึ่งถูกล้อมด้วยสี่เหลี่ยม กำลังครองบอลอยู่ ผู้เล่นที่วงกลมไว้คือ ผู้เล่นที่กำลังพิจารณาและไม่ได้ครองบอล

- ตำแหน่งกองหลัง

การเคลื่อนที่ของกองหลังจะคอยดูว่ามีผู้เล่นกองหน้าทีมตรงข้ามอยู่ในตำแหน่งของสนามฝั่งเดียวกับผู้เล่นฝั่งครองบอลหรือไม่ หากกองหน้าคนใดอยู่ฝั่งเดียวกันก็จะให้กองหลังคนนั้นคอยประกบติดเอาไว้เพื่อป้องกันการรุกโต้กลับของฝั่งตรงข้าม หากมีกองหน้าอยู่เพียงคนเดียวก็ใช้ประกบเพียงคนเดียว กองหลังอีกคนก็สามารถคอยประสานงานอยู่บริเวณกลางสนามได้ แต่ต้องอยู่ในตำแหน่งที่ต่ำกว่ากองหน้าอีกคนของฝั่งตรงข้าม สังเกตจากผู้เล่นในหมายเลข 1 จะคอยประผู้เล่นหมายเลข 5 ของฝั่งตรงข้ามเอาไว้ แต่ผู้เล่นหมายเลข 2 ไม่จำเป็นต้องประกบกองหน้าหมายเลข 6 ของฝั่งตรงข้าม เนื่องไม่ได้อยู่ในตำแหน่งสนามฝั่งเดียวกัน อยู่กันคนละครึ่งของสนาม



รูปที่ 5 - 4 ตำแหน่งของกองกลางในสนาม ขณะที่เป็นฝ่ายกำลังครองบอล

- ตำแหน่งกองกลาง

ส่วนของกองกลางจะต้องพยายามจะอยู่ในตำแหน่งที่สามารถที่จะส่งบอลถึงกันได้ ในลักษณะที่ต้องพยายามสร้างช่องให้เกิดขึ้นอย่างน้อย 2 ช่องเพราะว่ากองกลางถือสำคัญมากในการทำเกมรุก ขณะที่เป็นฝ่ายที่กำลังครองบอลอยู่จึงต้องทำให้เกิดช่องว่างอย่างน้อย 2 ทางกับผู้เล่นฝั่งเดียวที่อยู่ใกล้กับมันที่สุด ซึ่งจะทำให้สามารถจะส่งบอลถึงตัวผู้เล่นได้ คือ จากรูปให้สังเกตว่า ผู้เล่นกองกลาง 2 คนคือ หมายเลข 3 และ หมายเลข 4 โดยหมายเลข 4 จะพยายามที่จะเคลื่อนที่ให้หมายเลข 6 กับ 2 สามารถส่งบอลมาให้ได้

หากครองบอลอยู่ และ หมายเลข 3 จะพยายามที่จะเคลื่อนที่ให้หมายเลข 1 กับ 5 สามารถส่งบอลมาให้ได้ หรืออาจจะเป็นกรณีสมมติว่า ยกตัวอย่าง ผู้เล่นหมายเลข 2 ไม่มีช่องว่างสามารถที่จะส่งบอลมาให้หมายเลข 4 ได้ แต่สมมติหมายเลข 3 และ 6 สามารถมีช่องว่างที่จะส่งบอลให้หมายเลข 4 ได้ก็แสดงว่าหมายเลข 4 อยู่ในตำแหน่งที่ดีพอสมควร

- **ตำแหน่งกองหน้า**

ในส่วนตำแหน่งของกองหน้าแล้วหากครองบอลเกินครึ่งสนามของฝั่งตัวเองแล้ว ก็จะมีวิธีการเคลื่อนที่ โดยที่กองหน้าคนไหนอยู่ผู้เล่นฝั่งเดียวกันที่ครองบอลอยู่มากกว่าจะอยู่ตำแหน่งที่ต่ำกว่ากองหน้าอีกคน ซึ่งจะอยู่ตำแหน่งที่สูงกว่าหรือพูดง่าย ๆ คือ อยู่ใกล้ประตูฝั่งตรงข้ามมากกว่านั่นเอง จากรูปหากผู้เล่นหมายเลข 3 กำลังครอบครองบอลอยู่ ทำให้กองหน้าที่ใกล้ หมายเลข 3 ที่สุด คือ หมายเลข 5 ดังนั้นหมายเลข 5 ก็จะอยู่ในตำแหน่งที่ต่ำกว่ากองหน้าอีกคนคือ 6 ซึ่งจะอยู่ใกล้ประตูฝั่งตรงข้ามมากกว่า

5.3 รูปแบบการส่งลูกฟุตบอลของนักบอล

การส่งบอลถือว่าเป็นเรื่องที่สำคัญ เพราะว่าการส่งบอลแต่ละครั้งมักจะทำให้เกิดช่องว่าง เพื่อที่จะสามารถส่งบอลไปประตูอยู่ฝั่งตรงข้ามได้เร็วยิ่งขึ้น เพราะฉะนั้นการส่งบอลจึงมีหลายปัจจัยที่จะนำมาใช้พิจารณาเช่น ตำแหน่งที่จะส่งบอลไป ผู้เล่นทีมเดียวกันที่อยู่ใกล้ ทิศทางการเคลื่อนที่ของผู้ที่จะรับบอลเป็นต้น

- **ตำแหน่งกองหลัง**

ในส่วนการส่งบอลของกองหลังจะพิจารณาก่อนว่าผู้เล่นในตำแหน่งกองกลางที่จะส่งไปนั้นมีช่องว่างที่จะส่งให้กองกลางตัวอื่นหรือว่าส่งไปยังกองหน้าได้หรือเปล่า หากมีก็จะส่งไปกองกลางตัวนั้นก่อนเพราะว่ามีทางที่ให้กองหน้าได้แล้ว แต่หากส่งไปยังผู้เล่นกองกลางนั้นไม่ได้ ก็จะพิจารณาก่อนว่าสามารถที่จะเลี้ยงบอลไปข้างหน้าเองได้หรือเปล่าถ้าหากไปได้ก็เลี้ยงไปก่อน หากเลี้ยงบอลไปไม่ได้ก็จะพิจารณาว่าส่งให้กองหลังด้วยกันได้หรือไม่หากไม่ได้ก็จะส่งกลับหลังให้ผู้รักษาประตู แต่หากส่งไปให้ใครไม่ได้เลยก็จะต้องทำการเลี้ยงหลบหลีกหรือพักบอลไว้ก่อน รอให้มีช่องว่างจึงค่อยส่ง

- **ตำแหน่งกองกลาง**

ส่วนผู้เล่นตำแหน่งกองกลางจะพิจารณากองหน้าว่าส่งไปได้หรือเปล่าและต้องดูก่อนว่ากองหน้านั้นมีช่องว่างที่จะยิงประตูได้หรือไม่ หากเป็นไปตามเงื่อนไขนี้ก็สามารถส่งบอลให้คนนั้นได้ แต่หากส่งไปไม่ได้ก็จะพิจารณากองกลางแล้วจากนั้นก็จะเป็นกองหลังตามลำดับ

- **ตำแหน่งกองหน้า**

ตำแหน่งกองหน้าเริ่มต้นก็จะพิจารณาก่อนว่ามีช่องว่างสำหรับที่จะยิงประตูได้หรือไม่ หากมีช่องทางทำประตูได้ก็จะทำการยิงประตูทันทีแต่หากยิงไม่ได้ก็จะพิจารณากองหน้าตัวอื่น โดยต้องดูก่อนว่ากองหน้าตัวนั้นมีช่องที่จะยิงประตูได้หรือเปล่า หากกองหน้าตัวนั้นมีช่องว่างยิงได้ก็จะส่งไปให้ ถ้าไม่ได้ก็จะพิจารณาที่ตำแหน่งกองกลางว่ามีช่องว่างส่งให้ตัวใดได้บ้างโดยกองกลางตัวที่จะรับนั้นต้องสามารถมีช่องที่จะส่งกลับมายังที่กองหน้าได้ด้วย หากไม่ได้ก็จะทำการพักบอลไว้ก่อนเพื่อรอโอกาสช่องว่างเกิดขึ้นแล้วค่อยส่งไปให้ผู้เล่นที่ได้พิจารณาแล้วว่าส่งไปแล้วได้เปรียบและสามารถส่งต่อไปให้ผู้เล่นคนอื่นได้

5.4 การนำแมชชีนเลิร์นนิ่งมาใช้วิเคราะห์รูปแบบการส่งบอล

เมื่อทำการแข่งขันจบครั้งแรกของการแข่งขัน ก็จะนำข้อมูลที่ได้จากการแข่งขันไปทำการเทรน ซึ่งข้อมูลที่จะใช้เป็นอินพุตเพื่อจะไปเทรนคือ ตำแหน่งของผู้เล่นขณะที่ส่งบอล, ทิศทางการเคลื่อนที่ของนักบอลขณะที่มีการส่งบอล ส่วนเอาต์พุตที่จะนำไปเทรนคือ ผลลัพธ์ของการส่งครั้งนั้นว่าส่งสำเร็จคือลูกบอลสามารถส่งไปให้ฝั่งเดียวกันหรือว่าส่งไปแล้วผิดพลาด ลูกบอลไปอยู่ในการครอบครองของฝั่งตรงกันข้ามโดยมีขั้นตอนการทำการเทรนดังนี้

- กำหนดค่าของอินพุตที่และเอาต์พุตที่เป็นไปได้ที่จะนำเข้าไปเทรน

- ตำแหน่งของผู้เล่นในสนามขณะที่มีการส่งบอล ซึ่งจะประกอบด้วยตำแหน่งในสนามมีอยู่ 2 ค่าคือค่าตำแหน่งแกน X และ แกน Y โดยแกน X คือขนาดความกว้างของสนาม แกน Y คือขนาดความสูงของสนาม ซึ่งจะต้องทำการแปลงตำแหน่งของนักบอลในการแกน X และ Y ให้อยู่ในช่วง [0, 1] คือนำตำแหน่งนักบอลที่อยู่ในสนามที่ตำแหน่งแกน X มาหารด้วยขนาดความกว้างของสนาม ส่วนตำแหน่งนักบอลที่อยู่ในสนามที่ตำแหน่งแกน Y มาหารด้วยขนาดความสูงของสนาม ซึ่งก็จะทำให้ค่าตำแหน่งของนักบอลที่อยู่ในสนามไม่ว่าจะเป็นแกน X หรือ Y ก็อยู่ในช่วง [0, 1]

- ทิศทางของนักบอล ขณะกำลังส่งบอลอยู่ โดยจะมีทิศทางอยู่ 8 ทิศทาง ซึ่งกำหนดเป็นค่าคงที่ตั้งแต่ 0 - 7 ซึ่งก็จะต้องมีค่าอยู่ในช่วง [0, 1] เช่นกันด้วยวิธีการนำค่าคงที่ของทิศทางของผู้เล่นแล้วหารด้วยค่าคงที่สูงสุดของทิศทางซึ่งก็คือ 7

- กำหนดค่าของเอาต์พุตไว้ 2 ค่าคือ 1 กับ 0 หากมีค่าเป็น 1 คือแสดงว่ารูปแบบของการส่งบอลครั้งนั้นสามารถส่งไปให้ผู้เล่นทีมเดียวกันได้สำเร็จ แต่หากเอาต์พุตมีค่าเป็น 0 ก็แสดงว่าการส่งบอลครั้งนั้นผิดพลาดไม่สามารถส่งบอลไปให้ผู้เล่นทีมเดียวกันได้

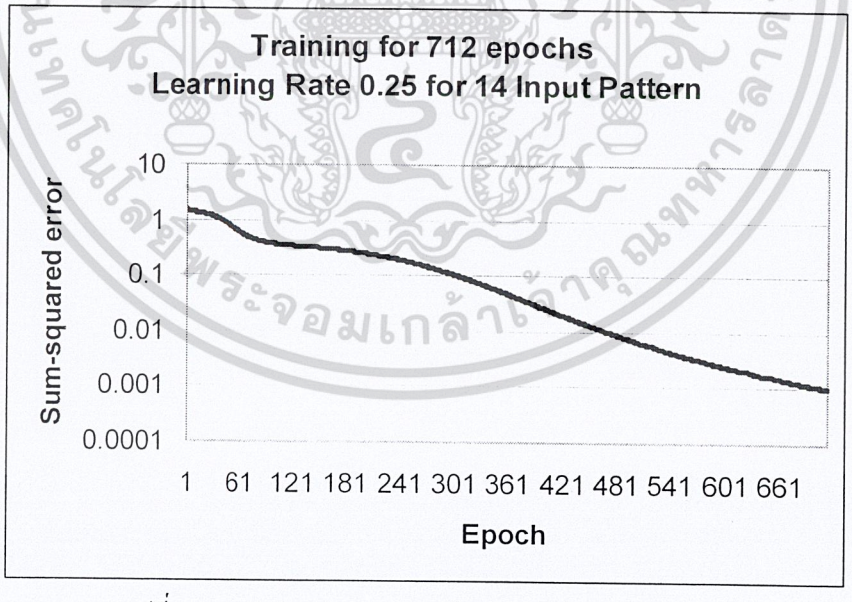
- กำหนดค่าของอัตราของการเรียนรู้ (Learning Rate) ให้มีค่าเท่ากับ 0.25
- กำหนดจำนวนอินพุตที่นำเข้าไปเทรน ซึ่งจะมีจำนวนอินพุตที่จะนำเข้าไปเทรนเท่ากับ จำนวนผู้เล่น * จำนวนทีม * ตำแหน่งแกน X และ Y + ทิศทางผู้ส่งบอล
  ซึ่งมีค่าเท่ากับ 7 * 2 *2 + 1 = 29 อินพุต
- กำหนดโครงสร้างของนิวรอลซึ่งจะประกอบด้วย อินพุตเลเยอร์มี 29 นิวรอลเท่ากับจำนวนอินพุต ฮิดเดนเลเยอร์มี 31 นิวรอลและเอาต์พุตเลเยอร์ 1 นิวรอลซึ่งเท่ากับ จำนวนเอาต์พุต
- ใช้วิธีการสุ่มเลือกแบบมาตรฐานทำการสุ่มเลือกสุ่มเลือกค่าของน้ำหนักและเทรดโฮล ซึ่งจะอยู่ในช่วงแคบๆที่อยู่ในช่วง $(-2.4 / F_i, 2.4 / F_i)$ โดย Fi คือ จำนวนอินพุตที่เข้าไปในอินพุตเลเยอร์ ซึ่งก็เท่ากับ 29
- นำชุดข้อมูลของอินพุต (Input Pattern) ทั้งหมดเข้าไปทำการเทรน ซึ่งก็จะมีจำนวนชุดของอินพุตเท่ากับจำนวนการส่งทั้งหมดของทั้ง 2 ทีมนั่นเอง
- ทำการกระตุ้นนิวรอลโดยจากอินพุตในชั้นอินพุตเลเยอร์ไปจนถึงเอาต์พุตที่คาดหวังในชั้นเอาต์พุตเลเยอร์
- ทำการปรับค่าถ่วงน้ำหนักโดย ทำการปรับจากค่าความผิดพลาดในชั้นเอาต์พุตเลเยอร์ แล้วทำการปรับมาจนถึงฮิดเดนเลเยอร์

- ทำซ้ำไปเรื่อย ๆ จนกว่าจะได้ค่าผลรวมของกำลังสองของค่าความผิดพลาดน้อยกว่า 0.001 ก็จะเป็นอันสิ้นสุดการเทรน

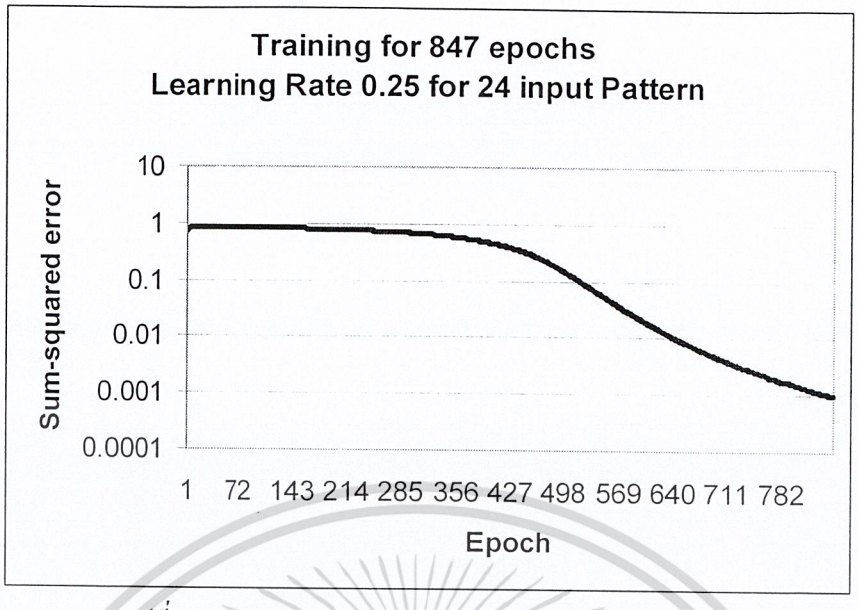
หลังจากที่ทำการเทรนเสร็จแล้วก็จะเก็บอินพุตที่นำเข้าไปเทรนเอาไว้เพื่อที่จะนำไปใช้ในการเทรนครั้งต่อไป ครานี้ก็จะมาถึงการทดสอบโดยจะเริ่มจากการนำค่าถ่วงน้ำหนักที่ผ่านการเทรนแล้วมาทดสอบนำมาใช้ในการแข่งขันในเกม ซึ่งเป็นส่วนช่วยในการตัดสินใจว่าจะส่งบอลเมื่อใดจึงจะได้ผลดี จากนั้นเมื่อทำการทดสอบเสร็จแล้วก็จะได้ผลการทดลองคือ สามารถเปรียบเทียบเปอร์เซ็นต์การส่งบอลระหว่างช่วงก่อนการเทรนกับหลังจากเทรนเสร็จแล้ว หลังจากนั้นก็จะทำการเทรนด้วยโดยนำข้อมูลอินพุตการเทรนรอบที่แล้วมาใช้เทรนด้วยเพื่อให้ได้ข้อมูลที่มีความแม่นยำมากยิ่งขึ้น แล้วก็จะเก็บอินพุตเหล่านี้ไปใช้ในการเทรนครั้งต่อ ๆ ไปอีก แต่ข้อมูลอินพุตเหล่านี้ก็จะมีการเก็บค่าไว้จำกัดคงที่ไม่ได้ขยายไปเรื่อย ๆ โดยจะใช้หลักการว่าข้อมูลอินพุตที่เพิ่งใช้ล่าสุดจะถูกเก็บค่าไว้ แต่หากอินพุตตัวไหนไม่ได้ใช้นาน ๆ ก็จะไม่เก็บค่าเหล่านั้นไว้แล้วก็จะทิ้งไป

5.5 วิธีการทดลองและผลการทดลอง

การทดลองเริ่มจากการแข่งขันเล่นเกมฟุตบอล โดยใช้กฎการส่งบอลแบบกฎเกณฑ์ที่ได้กำหนดไว้ โดยเมื่อแข่งขันครบตามเวลาการแข่งขันที่ได้กำหนดไว้แล้วเก็บค่าเปอร์เซ็นต์การส่งบอลที่ไม่ผิดพลาดเอาไว้เพื่อนำไปเปรียบเทียบ จากนั้นก็นำค่าอินพุตต่าง ๆ ที่จะใช้ในการเทรน นำไปทำการเทรนแล้วเก็บค่าถ่วงน้ำหนักเอาไว้เพื่อใช้ในการแข่งขันต่อไป นอกจากนั้นก็เก็บอินพุตเหล่านั้นที่นำไปเทรนไว้ด้วยเพื่อใช้ในการเทรนครั้งต่อ ๆ ไปอีกด้วย

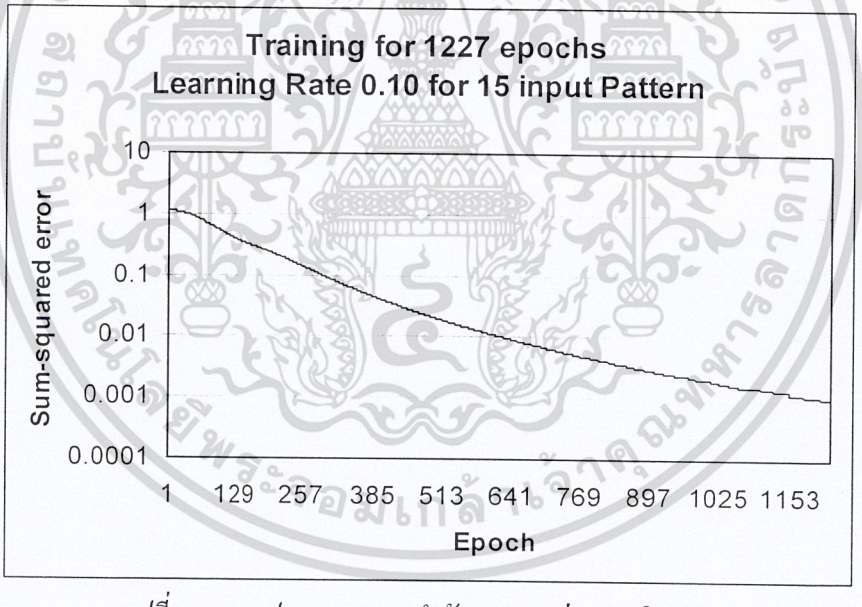


รูปที่ 5 - 5 กราฟแสดงผลรวมกำลังสองของค่าความผิดพลาด



รูปที่ 5 - 6 กราฟแสดงผลรวมกำลังสองของค่าความผิดพลาด

จากผลการทดลอง จำเป็นต้องใช้จำนวนรอบมากขึ้น เมื่อเพิ่มจำนวนรูปแบบของข้อมูล



รูปที่ 5 - 7 กราฟแสดงผลรวมกำลังสองของค่าความผิดพลาด

จากผลการทดลอง จะได้ว่าค่าที่เปอร์เซ็นต์การเรียนรู้ 0.25 เหมาะสมกว่าค่า 0.1

บทที่ 6
สรุปผลและวิจารณ์

โครงงานนี้เกี่ยวข้องกับเกมฟุตบอล ซึ่งได้พัฒนาโดยนำหลักการของแมชชีนเลิร์นนิ่งมาใช้ในการเรียนรู้ของเกม ซึ่งจะใช้หลักการของแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กที่จะทำการนิวรอลเชื่อมต่อกัน และนิวรอลนี่เองเปรียบเสมือนเป็นที่เก็บความรู้และความรู้นี้ก็สามารถเปลี่ยนแปลงได้โดยการฝึกหัด (Training) โดยจะนำมาใช้เรียนรู้เฉพาะการส่งบอลของระหว่างผู้เล่นเท่านั้น เพราะสามารถวัดผลได้ง่ายและเห็นผลได้อย่างชัดเจน โดยสามารถวัดผลได้จากเปอร์เซ็นต์การส่งบอลที่ไม่ผิดพลาด คือส่งบอลแล้วไม่เสียการครองบอลให้ฝั่งตรงข้าม เมื่อทำการเทรนแล้วแข่งขันใหม่ ถ้าหากเปอร์เซ็นต์การส่งบอลที่ไม่ผิดพลาดจะมีค่าเพิ่มขึ้น แสดงว่ามีการเรียนรู้ที่ถูกต้อง หากไม่เป็นเช่นนั้นแสดงว่าเรียนรู้ไม่สำเร็จ

6.1 ปัญหาที่พบและแนวทางแก้ไข

จากที่ได้ทำการศึกษาถึงการนำหลักการของแมชชีนเลิร์นนิ่งไปใช้ในเกมฟุตบอลนั้น ถือว่าเป็นเรื่องที่ยากเพราะเกมฟุตบอลมีปัจจัยที่สำคัญที่ส่งผลกระทบต่อการส่งบอลเป็นอย่างมากคือรูปแบบเคลื่อนที่ของผู้เล่นทั้ง 2 ทีม เพราะการเคลื่อนที่มีรูปแบบที่ค่อนข้างหลากหลายมาก ไม่สามารถที่จะกำหนดกฎตายตัวที่จะมาอธิบายได้ง่ายนัก จึงทำให้การเคลื่อนที่ในรูปแบบที่ต่างกันนี้ก็จะส่งผลกระทบต่อการส่งบอล ทำให้คาดเดาทิศทางในการส่งได้ยากขึ้น ซึ่งก็จะส่งผลกระทบต่อเปอร์เซ็นต์การส่งบอลที่อาจจะไม่เพิ่มขึ้นตามที่ควรจะเป็น

6.2 แนวทางพัฒนาเพิ่มเติม

- นำหลักการนิวรอลเน็ตเวิร์กผสมกับหลักการของเจเนติกอัลกอริทึม (Genetic Algorithms) เพื่อปรับปรุงการเรียนรู้ให้เร็วและทำให้รูปแบบโครงสร้างของนิวรอลดีขึ้น หรืออาจจะแผนภาพต้นไม้สำหรับการตัดสินใจ มาช่วยในการตัดสินใจเลือกอินพุตที่ไม่ควรถูกนำมาเทรนด้วยนิวรอลเน็ตเวิร์กออกไป
- ควรนำอินพุตบางตัวมาใช้ในการเทรนด้วย เช่น ระยะเวลานับตั้งแต่เริ่มรับบอลจากผู้ส่งจนกระทั่งเสียการครองบอลให้ฝั่งตรงข้ามหรืออื่น ๆ
- อาจจะวัดการเรียนรู้ไม่ใช่แค่เพียงการส่งบอลเท่านั้น เช่น เปอร์เซ็นต์ของการครอบครองบอล จำนวนการยิงประตู เป็นต้น แต่ก็หาตัวแปรที่ทำการเทรนและสามารถนำไปใช้เรียนรู้ได้ยากขึ้นตามไปด้วย

6.3 สรุปผลการทำงาน

จากการศึกษาการทำงานของแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์กนั้นมีความสามารถที่ในการทำงานสูง แต่มีข้อเสียตรงที่ไม่สามารถหาโครงสร้างของเน็ตเวิร์กที่เหมาะสมได้ จึงต้องทดลองแล้วตรวจสอบผลลัพธ์ที่ได้ ว่าผลรวมกำลังสองของค่าความผิดพลาดเข้าใกล้ศูนย์ ใช้เวลามากในการเทรนนิวรอลเน็ตเวิร์ก จากการประยุกต์ในการเรียนรู้ในการเล่นเกมฟุตบอล นับว่าได้ผลออกมาเป็นที่น่าพอใจ โดยที่จะเห็นได้ว่าเปอร์เซ็นต์การส่งบอลสำเร็จจะสูง โดยที่จะมีเปอร์เซ็นต์เพิ่มขึ้น

บรรณานุกรม

[1] Michael Negnevisky: "Artificial Intelligent: A Guide to Intelligent Systems", 1st ed. Addison-Wesley, 2002

[2] J. Ross Quinlan: "Programs For Machine Learning", Morgan Kaufmann

[3] Scott Robert Ladd: "Genetic Algorithms in C++", M&T Books, 1996

[4] Deitel & Deitel: "C++ How to Program", 2nd ed. Prentice Hall

[5] Stuart Russell & Peter Norvig: "Artificial Intelligence: A Modern Approach", Prentice Hall

[6] Valluru Rao & Hayagriva Rao: "C++ Neural Network and Fuzzy Logic", 2nd ed.MIS Press, 1995

เว็บไซต์อ้างอิง

http://www.gamedev.net

http://www.thaigamedevx.com

http://www.sourceforge.net

http://lancet.mit.edu/ga