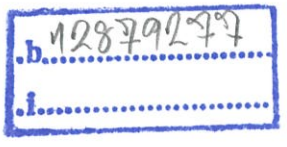
เอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง
ADAPTIVE GAME-PLAYING AGENT USING REINFORCEMENT LEARNING

จิรเดช อมรพิมลกุล
อภิพล ภู่งาม
จตุพงษ์ พุ่มจินดา

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

เอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง
ADAPTIVE GAME-PLAYING AGENT USING REINFORCEMENT LEARNING

จิรเดช อมรพิมลกุล
อภิพล ภู่งาม
จตุพงษ์ พุ่มจินดา



ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

ADAPTIVE GAME-PLAYING AGENT USING REINFORCEMENT LEARNING

JIRADET AMORNPIMONKUL
APIPON POONGAM
JATUPHONG PUMJINDA

A SPECIAL PROBLEM SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF
SCIENCE IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE KING
MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2015

หัวข้อปัญหาพิเศษ	เอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง Adaptive Game-Playing Agent using Reinforcement Learning
ชื่อนักศึกษา	นายจิรเดช อมรพิมลกุล รหัสนักศึกษา 55050241 นายอภิพล ภู่งาม รหัสนักศึกษา 55050528 นายจตุพงษ์ พุ่มจินดา รหัสนักศึกษา 55050230
ปริญญา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	ดร.สายชล ใจเย็น

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2558

คณะกรรมการสอบ	ลายมือชื่อ
ผศ.ดร.ศรัณย์ อินทโกสุม ประธานกรรมการ	
ผศ.ดร.อนันตพร หรรษคุณาฒัย กรรมการ	
ดร.สายชล ใจเย็น กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อปัญหาพิเศษ	เอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง			
ชื่อนักศึกษา	นายจิรเดช	อมรพิมลกุล	รหัสนักศึกษา	55050241
	นายอภิพล	ภู่งาม	รหัสนักศึกษา	55050528
	นายจตุพงษ์	พุ่มจินดา	รหัสนักศึกษา	55050230
ปริญญา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)			
ภาควิชา	วิทยาการคอมพิวเตอร์			
ปีการศึกษา	2558			
อาจารย์ที่ปรึกษา	ดร.สายชล ใจเย็น			

บทคัดย่อ

ในปัจจุบันเทคโนโลยีได้มีการเติบโตและก้าวหน้าอย่างมากในปัจจุบัน มีการเติบโตและก้าวหน้าทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ โดยในด้านซอฟต์แวร์ที่ในปัจจุบันมีการพัฒนาเพื่อให้ไปควบคุมการทำงานของฮาร์ดแวร์ เช่น หุ่นยนต์หรือเอเจนท์ ซึ่งในปัจจุบันการพัฒนาให้หุ่นยนต์ทำงานตามคำสั่งของผู้ใช้เป็นสิ่งที่ทำได้ทั่วไป แต่การจะพัฒนาให้หุ่นยนต์สามารถทำงานได้จากการเรียนรู้ การคิดและลงมือทำ เป็นสิ่งที่สามารถทำได้ยาก โดยในปัญหาพิเศษนี้จะประยุกต์ใช้การเรียนรู้แบบเสริมแรงเพื่อพัฒนาให้เอเจนท์มีความสามารถในการเรียนรู้และสามารถเล่นเกมที่สร้างได้

คำสำคัญ : การเรียนรู้ของเครื่อง การเรียนรู้แบบเสริมแรง การเรียนรู้แบบคิว

Title	Adaptive Game-Playing Agent using Reinforcement Learning			
Students	Mr. Jiradet	Amornpimonkul	Student ID	55050241
	Mr. Apipon	Poongam	Student ID	55050528
	Mr. Jatuphong	Pumjinda	Student ID	55050230
Degree	Bachelor of Science (Computer Science)			
Department	Computer Science			
Faculty	Science			
University	King Mongkut's Institute of Technology Ladkrabang(KMITL)			
Academic year	2015			
Advisor	Dr. Saichon Jaiyen			

Abstract

Nowadays technology has been growing considerably, not in just hardware but in software as well. Many developers try to develop software that can control the operations of the hardware, such as robot or AI (Artificial Intelligence. To make a computer do some task, we simply program it, but to make a computer learning how to do a task is still a hard things to do. In this project we apply the reinforcement learning to give a computer an ability to learn by itself and let it learn to play a game.

Keywords : Machine learning, Reinforcement learning, Q-learning

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเล่มนี้สามารถสำเร็จได้ด้วยดีจากการช่วยเหลือและสนับสนุนจากบุคคลหลายท่านคณะผู้จัดทำขอขอบพระคุณบุคคลดังต่อไปนี้

ดร.สายชล ใจเย็น อาจารย์ที่ปรึกษาปัญหาพิเศษที่กรุณาให้คำปรึกษาและแนวทางแก้ปัญหา รวมถึงตรวจสอบและแก้ไขการเขียนรายงานปัญหาพิเศษเล่มนี้อย่างละเอียด

อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้ประสิทธิ์ประสาทวิชาความรู้และให้คำปรึกษาทั้งในภาคทฤษฎีและภาคปฏิบัติเป็นอย่างดีตลอดระยะเวลา 4 ปี

เพื่อนและพี่ร่วมภาควิชาทุกคนในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้คำปรึกษาตลอดมา คุณพ่อและคุณแม่ที่คอยให้กำลังใจ ความช่วยเหลือในด้านต่าง ๆ นอกจากนี้อาจมีบุคคลท่านอื่นที่ไม่ได้กล่าวไว้ ณ ที่นี้ จึงใคร่ขอขอบพระคุณทุกท่านที่ให้ความกรุณามีส่วนร่วมในการให้ความช่วยเหลือในการทำปัญหาพิเศษเล่มนี้

จิรเดช อมรพิมลกุล
อภิพล ภู่งาม
จตุพงษ์ พุ่มจินดา

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหาพิเศษ

ในปัจจุบันเทคโนโลยีได้มีการเติบโตและก้าวหน้าอย่างมากในปัจจุบัน มีการเติบโตและก้าวหน้าทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ โดยในด้านซอฟต์แวร์ที่ในปัจจุบันมีการพัฒนาเพื่อให้ไปควบคุมการทำงานของฮาร์ดแวร์ เช่น หุ่นยนต์หรือเอเจนท์ ซึ่งในปัจจุบันการพัฒนาให้หุ่นยนต์ทำงานตามคำสั่งของผู้ใช้เป็นสิ่งที่ทำได้ทั่วไป แต่การจะพัฒนาให้หุ่นยนต์สามารถทำงานได้จากการเรียนรู้ การคิดและลงมือทำ เป็นสิ่งที่สามารถทำได้ยาก โดยในการจะพัฒนาให้หุ่นยนต์หรือเอเจนท์ ให้สามารถเรียนรู้ได้จะต้องใช้ความรู้จากวิชาปัญญาประดิษฐ์ ซึ่งการเรียนรู้ของเครื่อง (machine learning) เป็นหนึ่งในสาขาของวิชาปัญญาประดิษฐ์ โดยจะเกี่ยวข้องกับการเรียนรู้ข้อมูลและทำนายข้อมูล ซึ่งในการเรียนรู้ของเครื่องจะมีแขนงของการเรียนรู้แบบเสริมแรง (reinforcement learning)

การเรียนรู้แบบเสริมแรง (reinforcement learning) เป็นการเรียนรู้ที่สามารถนำมาพัฒนาเพื่อให้หุ่นยนต์หรือเอเจนท์ สามารถเรียนรู้และทำนายข้อมูลได้ ซึ่งจะใช้อัลกอริทึมของคิว (Q learning) ในการคำนวณหาค่าคิว (Q value) โดยในการเรียนรู้แบบเสริมแรงจะมีรูปแบบที่มีผู้สอนจะบอกหุ่นยนต์หรือเอเจนท์ ว่าสิ่งที่กระทำอยู่ในสิ่งแวดล้อมนั้นถูกหรือผิด ถ้าถูกจะมีการให้รางวัลเป็นบวก แต่ถ้าสิ่งที่กระทำนั้นไม่เหมาะสมกับสิ่งแวดล้อมนั้นหรือสถานการณ์นั้นจะมีการให้รางวัลเป็นลบ ซึ่งหุ่นยนต์หรือเอเจนท์ จะนำรางวัลนี้ไปคำนวณเป็นค่าคิว โดยค่าคิวจะเป็นค่าที่บันทึกไว้ทุกๆการกระทำในสถานการณ์หรือสภาพแวดล้อมต่างๆทั้งหมด เมื่อหุ่นยนต์หรือเอเจนท์ พบเจอสถานการณ์ที่ไม่เคยพบเจอก็จะสุ่มการกระทำและบันทึกค่าไว้ และเมื่อหุ่นยนต์หรือเอเจนท์ พบเจอสถานการณ์ที่เคยพบเจอก็จะไปดูค่าคิว แล้วเลือกที่จะกระทำตามค่าคิวที่มีการบันทึกไว้ โดยในการเลือกจะเลือกที่มีค่าคิวสูงที่สุดมากระทำ

1.2 วัตถุประสงค์ของปัญหาพิเศษ

เพื่อพัฒนาให้เอเจนท์ภายในเกมมีความสามารถในการเรียนรู้ และสามารถเล่นเกมเองได้โดยใช้การเรียนรู้แบบเสริมแรง และใช้อัลกอริทึมของคิวในการพัฒนา

1.3 ข้อจำกัดและขอบเขตของปัญหาพิเศษ

1) เอเจนท์ภายในเกมจะใช้การเรียนรู้แบบเสริมแรง และอัลกอริทึมของคิวในการพัฒนา
2) พัฒนาเอเจนท์และเกมโดยใช้ภาษาจาวาสคริปต์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้เรื่องการเรียนรู้แบบเสริมแรง
2) เอเจนท์ภายในเกมมีความสามารถในการเรียนรู้ และเล่นเกมเองได้
3) สามารถนำความรู้ในการพัฒนาไปประยุกต์และต่อยอดได้

1.5 ขั้นตอนการดำเนินการ

1) ศึกษาการเรียนรู้แบบเสริมแรง และอัลกอริทึมของคิว
2) ศึกษาวิธีการพัฒนาเกม
3) ออกแบบ พัฒนาเอเจนท์และเกม
4) พัฒนาเอเจนท์ในเกมให้สามารถเรียนรู้และเล่มเกมได้
5) ศึกษาและประเมินผลการกระทำของเอเจนท์
6) จัดทำเล่มเอกสารปัญหาพิเศษ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในการทำปัญหาพิเศษเรื่องเอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง ผู้จัดทำได้ศึกษาเอกสารและทฤษฎีต่างๆที่เกี่ยวข้องในหัวข้อดังต่อไปนี้

2.1 การเรียนรู้ของเครื่อง (Machine learning)

2.2 งานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้ของเครื่อง (Machine learning)

การเรียนรู้ของเครื่องเป็นศาสตร์หนึ่งของสาขาวิทยาการคอมพิวเตอร์ซึ่งมีวิวัฒนาการมาจากการจดจำรูปแบบ (pattern recognition) ของปัญญาประดิษฐ์ (artificial intelligence) การเรียนรู้ของเครื่องเป็นการศึกษาการเรียนรู้และโครงสร้างของอัลกอริทึม ที่สามารถเรียนรู้ และทำการคาดเดาผลจากข้อมูลได้ อัลกอริทึม ในการเรียนรู้ของเครื่องนั้นจะทำงานโดยการสร้างโมเดลจากอินพุตตัวอย่างและคาดเดาการตัดสินใจที่จะทำ ต่างจากอัลกอริทึมของโปรแกรมทั่วไปที่โปรแกรมจะทำงานตามคำสั่งที่ให้ไว้

การเรียนรู้ของเครื่องนั้นมีความเกี่ยวข้องกันกับการคำนวณเชิงสถิติ ซึ่งจะเน้นในเรื่องของทำการทำนายผลลัพธ์การเรียนรู้ของเครื่องจักร โดยจะนำมาใช้ในการคำนวณในโปรแกรมที่ไม่สามารถใช้อัลกอริทึมแบบตรงๆได้ ตัวอย่างเช่น การทำการคัดกรองสแปม (spam filtering) ในอีเมล (Email), การจดจำใบหน้า, การทำเสิร์ทเอนจิน (Search engine) และอื่นๆ

การเรียนรู้ของเครื่องโดยทั่วไปแล้วจะแบ่งรูปแบบของปัญหาได้เป็น 3 รูปแบบ คือ

2.1.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning)

การเรียนรู้แบบมีผู้สอนคือ กลุ่มของข้อมูลที่ใส่เข้าไป (input data) ที่รู้ผลลัพธ์ของข้อมูลแล้ว การเรียนรู้แบบมีผู้สอน ข้อมูลจะถูกเตรียมไว้เพื่อใช้ในการทำนายข้อมูล โดยใช้ Regression และ Classification โดยตัวอย่างของการเรียนรู้แบบมีผู้สอนเช่น ต้องการจะสอนให้คอมพิวเตอร์แยกรูปภาพแมวกับรูปภาพอื่นๆ โดยข้อมูลที่จะสอนคอมพิวเตอร์คือการนำรูปภาพแมวให้คอมพิวเตอร์ดู และจะบอกคอมพิวเตอร์ว่าภาพที่เห็นคือภาพแมว และบอกด้วยลักษณะของแมวให้ด้วยเช่น มี 4 ขา เป็นต้น ด้วยขั้นตอนพวกนี้จะทำให้คอมพิวเตอร์สามารถแยกภาพแมวจากภาพอื่นๆได้

Regression

Regression คือการจะทำนายข้อมูลที่มีผลลัพธ์เป็นข้อมูลแบบต่อเนื่อง (Continuous) เช่น การมีข้อมูลของหุ้นรายเดือนย้อนหลัง 10 ปี แล้วนำเอาข้อมูลนั้นมาทำนายว่าในเดือนหน้าราคาหุ้นจะอยู่ที่ประมาณเท่าไหร่หรือการมีข้อมูลสภาพอากาศจะนำมาทำนายอุณหภูมิในวันพรุ่งนี้ว่าจะสูงหรือต่ำกี่องศา เป็นต้น

Classification

Classification คือการที่จะทำนายข้อมูลที่มีผลลัพธ์เป็นข้อมูลแบบไม่ต่อเนื่อง (Discrete) หรือการจัดกลุ่ม เช่น มีอีเมลล์แล้วจะจัดกลุ่มว่าอีเมลล์ที่เข้ามานั้นเป็นอีเมลล์สแปมหรืออีเมลล์ไม่สแปม จากข้อมูลอีเมลล์ที่เคยได้รับ หรือจากตัวอย่างการแยกภาพแมวจากภาพอื่นๆ เป็นต้น

2.1.2 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning)

การเรียนรู้แบบไม่มีผู้สอน คือ กลุ่มของข้อมูลที่ใส่เข้าไปนั้นจะไม่รู้ผลลัพธ์ที่จะออกมา ในการที่จะเข้าใกล้คำตอบหรือเข้าใจปัญหาเพิ่มมากขึ้นทำได้โดยการจัดโครงสร้างของข้อมูล โมเดลจะถูกเตรียมไว้เพื่อใช้ในการจัดโครงสร้างของข้อมูลเพื่อลดความซ้ำซ้อนและจัดข้อมูลให้อยู่ในลักษณะเดียวกัน ได้แก่ Clustering

Clustering

Clustering คือการแบ่งกลุ่มข้อมูลเช่น การต้องการแบ่งกลุ่มภาพของสัตว์สามชนิดที่ลักษณะหน้าตาไม่เหมือนกัน ซึ่งจะไม่มีการสอนคอมพิวเตอร์ถึงลักษณะของข้อมูลและผลลัพธ์ โดยคอมพิวเตอร์จะดูลักษณะของภาพสัตว์แล้วสามารถแบ่งเป็นสามกลุ่มได้ แต่เมื่อมีข้อมูลมาให้คอมพิวเตอร์ทำนาย คอมพิวเตอร์ก็แยกได้ว่าข้อมูลรูปที่ได้มาอยู่กลุ่มใด แต่บอกไม่ได้ว่าข้อมูลที่ได้คืออะไร

ระหว่างการเรียนรู้แบบมีผู้สอนและการเรียนรู้แบบไม่มีผู้สอน จะมีการเรียนรู้อีกรูปแบบหนึ่งเรียกว่า semi-supervised learning โดยผู้ฝึกสอนจะให้ชุดข้อมูลฝึกสอนที่ไม่สมบูรณ์ คือเอาต์พุตเป้าหมายบางตัวจะขาดหายไป รูปแบบอื่นๆของการเรียนรู้ของเครื่องเช่น Learning to Learn คือการให้คอมพิวเตอร์เรียนรู้การปรับค่า bias จากประสบการณ์ก่อนหน้า

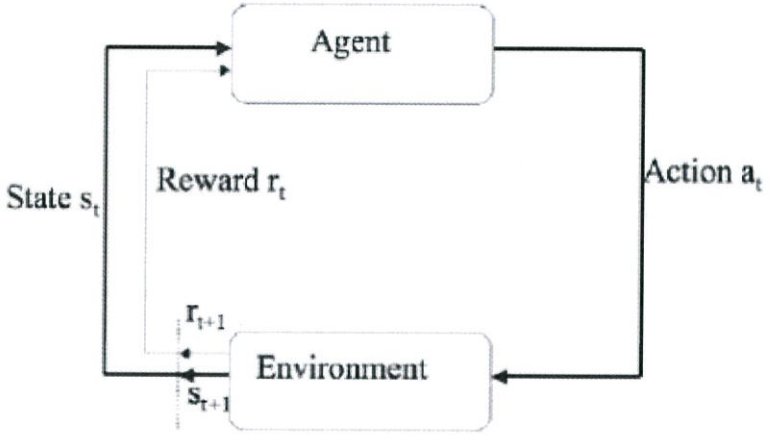
2.1.3 การเรียนรู้แบบเสริมแรง (Reinforcement Learning)

การเรียนรู้แบบเสริมแรงคือ การเรียนรู้ว่าจะต้องทำอะไร ให้เหมาะสมกับสถานการณ์ โดยผลของการกระทำจะให้รางวัล (Reward) ในรูปแบบตัวเลข ผู้เรียนจะไม่ถูกบอกว่าจะต้องกระทำอะไรถึงจะได้รางวัลที่มากที่สุด แต่ผู้เรียนจะต้องค้นพบการกระทำที่จะได้รับรางวัลที่มากที่สุดจากการกระทำของผู้เรียนเอง

การเรียนรู้แบบเสริมแรง ไม่ได้ถูกกำหนดการพัฒนาให้พัฒนาจากอัลกอริทึมการเรียนรู้ แต่ถูกพัฒนาจากปัญหาที่ได้เรียนว่า อัลกอริทึมใดที่เหมาะสมที่สุดในการแก้ปัญหา ที่เราตัดสินใจนำมาเป็นอัลกอริทึมการเรียนรู้แบบเสริมแรง ส่วนใหญ่รูปแบบของปัญหาการเรียนรู้แบบเสริมแรงในแง่ของการควบคุมที่เหมาะสมเป็นกระบวนการตัดสินใจในรูปแบบของมาร์คอฟ (Markov Decision Process : MDP) ซึ่งแนวคิดพื้นฐานนั้นมีส่วนประกอบคือ เอเจนท์ (Agent) , ชุดการกระทำ (Action) และสภาพแวดล้อม (Environment) ซึ่งในการเรียนรู้แบบเสริมแรงนั้น กระบวนการส่วนใหญ่จะเกี่ยวข้องกับความสัมพันธ์ระหว่าง เอเจนท์ และสภาพแวดล้อมเพื่อให้บรรลุเป้าหมาย

ซึ่งเอเจนท์ต้องสามารถรู้ขอบเขตและการกระทำที่เป็นไปได้ทั้งหมดที่จะส่งผลต่อสภาพแวดล้อมปัจจุบัน (State) หลังจากนั้น การกระทำที่ถูกเลือก จะมีผลทำให้สภาพแวดล้อม เปลี่ยนแปลงไป และเอเจนท์จะได้รับรางวัล (Reward) ซึ่งขึ้นอยู่ว่าการกระทำดังกล่าวมีผลให้สภาพแวดล้อมเปลี่ยนแปลงไปในทิศทางใด (ดีขึ้นหรือแย่ลง ถ้าดีขึ้นรางวัลจะได้มาก แต่ถ้าแย่ลงรางวัลจะได้น้อย) จากรางวัลที่ได้รับ เอเจนท์จะพยายามค้นหา นโยบาย (Policy) ในการเลือก การกระทำ ในสภาพแวดล้อมใดๆเพื่อให้รางวัลในระยะยาวนั้นมีค่ามากที่สุด

การเรียนรู้แบบเสริมแรงนี้แตกต่างจากการเรียนรู้แบบมีผู้สอน (Supervised learning) ตรงที่ตัวเอเจนท์เองจะไม่ได้เรียนรู้จากชุดข้อมูลตัวอย่าง (training set) แต่จะทำการโต้ตอบกับสภาพแวดล้อมที่เอเจนท์กำลังทำงานอยู่โดยตรง ดังนั้นข้อมูลเดียวที่ผู้เรียนสามารถใช้ในการเรียนรู้ได้ก็คือรางวัลที่ได้รับเมื่อมีการเลือกการกระทำใดการกระทำหนึ่ง ซึ่งเราสามารถมองการเรียนรู้แบบนี้ได้เป็น การเรียนรู้แบบลองผิดลองถูก (trial-and-error) โดยส่วนใหญ่แล้วจะมีการนำการเรียนรู้แบบนี้ไปใช้ในงานที่เกี่ยวข้องกับการควบคุมหรือเกมเช่น การควบคุมหุ่นยนต์ หมากรุก เป็นต้น



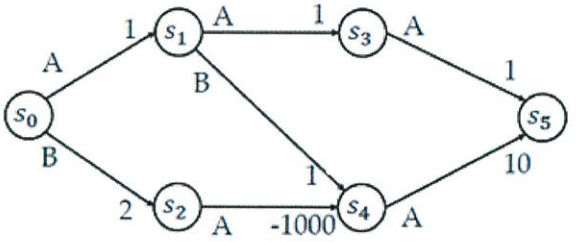
รูปที่ 2.1 ปฏิสัมพันธ์ระหว่างเอเจนท์กับสภาพแวดล้อม

รูป 2.1 แสดงแผนภาพการโต้ตอบระหว่างเอเจนท์กับสภาพแวดล้อมตามลำดับขั้นเวลา t=0,1,2,... ในแต่ละขั้นเวลา (time step) ผู้เรียนจะได้รับสภาพแวดล้อมปัจจุบันมาในรูปของ state, $s_t \in S$ โดยที่ S คือเซตของสถานะที่เป็นไปได้ทั้งหมด จากนั้นเอเจนท์ต้องทำการเลือกการกระทำ $a_t \in A(s_t)$ หมายถึงเซตของการกระทำที่มีอยู่ของสถานะ s_t และจะได้รับรางวัลเป็นเลขจำนวนหนึ่งกลับมา $r_t \in R$ และไปยังสถานะใหม่ $(s_t + 1)$

2.1.3.1 Value Functions

ผู้เรียนจะพยายามกำหนดว่าการกระทำอะไรที่จะเพิ่มรางวัลที่คาดหวัง (expected reward) ไว้ในอนาคต โดยรางวัลที่คาดหวังไว้จะเรียกว่า value ซึ่งจะมีวิธีที่การคำนวณหา value คือ Action-value function (Q (s,a)) ซึ่งจะมีสมการคือ

$$Q(s,a) = R(s,a,s') + \max Q(s',a') \qquad (2.1)$$



รูปที่ 2.2 ตัวอย่างเส้นทาง

จากรูปที่ 2.2 วงกลมแสดงถึง state และเส้นระหว่าง state จะมีรางวัลของแต่ละเส้นทาง ดังตัวอย่างเช่น จาก ถ้า state S_0 เลือกเส้นทาง A จะมี reward = 1 และจะไปที่ state S_1 ถ้า state S_0 เลือกเส้นทาง B จะมี reward = 2 และจะไปที่ state S_2

โดย state เริ่มต้นคือ S_0 และ state สุดท้ายคือ S_5 ซึ่งเส้นทางจาก S_0 ไป S_5 มีดังนี้

$S_0 \longrightarrow S_1 \longrightarrow S_3 \longrightarrow S_5$
$S_0 \longrightarrow S_1 \longrightarrow S_4 \longrightarrow S_5$
$S_0 \longrightarrow S_2 \longrightarrow S_4 \longrightarrow S_5$

ตัวอย่างการคำนวณคือ

จากสมการ $Q(s,a) = R(s,a,s') + \max Q(s',a')$ จะได้

$Q(S_0,A) = R(S_0, A_1, S_1) + \max (Q(S_1,A), Q(S_1,B))$

สิ่งที่ต้องหาคือ $Q(S_1,A)$, $Q(S_1,B)$ ซึ่งจะได้

$Q(S_1,A) = R(S_1, A_1, S_3) + \max (Q(S_3,A))$

สิ่งที่ต้องหาคือ $Q(S_3,A)$

$Q(S_3,A) = R(S_3, A_1, S_5)$

$Q(S_3,A) = 1$

เมื่อได้ $Q(S_3,A) - 1$ นำไปแทนค่าใน $Q(S_1,A) = R(S_1, A_1, S_3) + \max(Q(S_3,A))$ จะได้

$Q(S_1,A) = R(S_1, A_1, S_3) + \max(Q(S_3,A))$

$= 1 + 1$

$Q(S_1,A) = 2$

$Q(S_1,B) = R(S_1, B_1, S_4) + \max(Q(S_4,A))$

สิ่งที่ต้องหาคือ $Q(S_4,A)$

$Q(S_4,A) = R(S_4, A_{10}, S_5)$

$Q(S_4,A) = 10$

เมื่อได้ $Q(S_4,A) = 10$ นำไปแทนค่าใน $Q(S_1,B) = R(S_1, B_1, S_4) + \max(Q(S_4,A))$ จะได้

$Q(S_1,B) = R(S_1, B_1, S_4) + \max(Q(S_4,A))$

$= 1 + 10$

$Q(S_1,B) = 11$

เมื่อได้ $Q(S_1,A) = 2$ และ $Q(S_1,B) = 11$

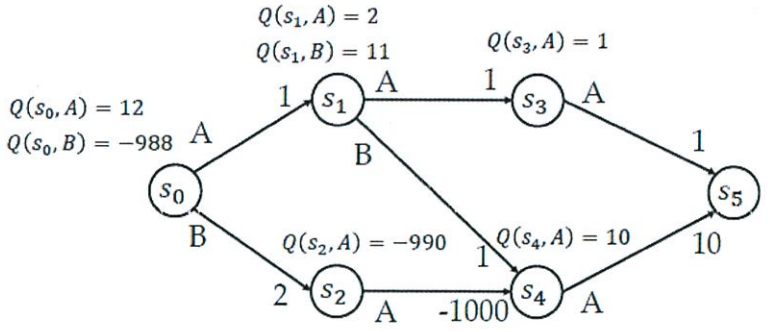
นำไปแทนค่าใน $Q(S_0,A) = R(S_0, A_1, S_1) + \max(Q(S_1,A), Q(S_1,B))$ จะได้

$Q(S_0,A) = R(S_0, A_1, S_1) + \max(Q(S_1,A), Q(S_1,B))$

$= 1 + \max(2,11)$

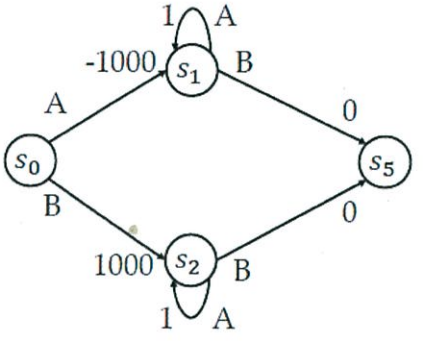
$= 1 + 11$

$Q(S_0,A) = 12$



รูปที่ 2.3 ตัวอย่างเส้นทางและ Q value

2.1.3.2 Discount factor (γ)



รูปที่ 2.4 ปัญหาวนลูป

จากรูปที่ 2.4 วงกลมแสดงถึง state และเส้นระหว่าง state จะมีรางวัลของแต่ละเส้นทางโดย state S_1 เส้นทาง A จะเป็นการวนเข้า state S_1 เป็นลักษณะของลูปไม่รู้จบ

$Q(S_1,A) = 1 + Q(S_1,A)$

$= 1 + 1 + Q(S_1,A)$

$= 1 + 1 + 1 + Q(S_1,A)$

$= \ldots$

จากรูปจะแสดงให้เห็นถึงปัญหาการวนลูปไม่รู้จบ ซึ่งจากปัญหานี้มีการใช้ discount factor (γ) ซึ่งจะมีค่า $0 \leq \gamma \leq 1$ เมื่อนำไปใส่ใน Q value จะได้สมการ

$$Q(s,a) = R(s,a,s') + \gamma \max Q(s',a') \qquad (2.2)$$

ซึ่งเมื่อกำหนดให้ $\gamma = 0.5$ จะได้

$Q(S_1,A) = 1 + 0.5(Q(S_1,A))$

$= 1 + 0.5 + 0.5(Q(S_1,A))$

$= 1 + 0.5 + 0.25 + 0.5(Q(S_1,A))$

$= 1 + 0.5 + 0.25 + 0.125 + 0.5(Q(S_1,A))$

$= \ldots$

ทำให้ค่า $Q(S_1,A)$ ที่เป็นลูปมีการบวกค่าที่น้อยลงเรื่อยๆ เมื่อผ่านไปหลายๆรอบ จะทำให้ค่าที่บวกใกล้เคียงกับ 0 ทำให้สามารถออกจากลูปได้ซึ่ง γ จะนำมาใช้ในปัญหาที่ state มีการติดอยู่ในลูปดังตัวอย่าง

2.1.3.3 Q-Learning

เป็นวิธีการสำหรับใช้ในการทำให้เครื่องคอมพิวเตอร์สามารถเรียนรู้การแก้ไขปัญหา ซึ่ง Q-learning เป็นหนึ่งในเทคนิคของ Reinforcement learning โดยใช้หลักการในการลองผิดลองถูกเพื่อที่จะหาการกระทำที่ทำให้ได้รับสิ่งตอบแทนที่พึงพอใจซึ่งเป็นผลจากการกระทำนั้นๆโดย Q-learning ถือได้ว่าเป็นวิธีการที่นำมาหาการกระทำที่ให้ผลลัพธ์ได้ดีที่สุด ณ สถานะหนึ่งๆ ซึ่งจะใช้หลักการของการตัดสินใจเลือกการกระทำจากฟังก์ชันคำนวณค่าของการกระทำโดยฟังก์ชันนี้จะทำการค้นหาว่าการกระทำใดมีค่าสูงที่สุดจากการกระทำที่เป็นไปได้ทั้งหมด ณ สถานะนั้นๆโดยค่าของการกระทำดังกล่าวจะถูกเรียกว่าค่าคิว (Q) จากนั้นก็จะเลือกการกระทำดังกล่าวมาใช้ ณ สถานะนั้นๆ เนื่องจากค่าคิวในตอนแรกนั้นยังไม่สามารถบ่งบอกได้ว่าการกระทำใดที่ควรจะถูกเลือก ต้องทำให้ฟังก์ชันดังกล่าวเรียนรู้ว่าควรจะให้เปลี่ยนแปลงค่าคิวของการกระทำต่างๆอย่างไร โดยเรียนรู้จากการลองผิดลองถูกก่อนเพื่อที่จะสังเกตว่าผลลัพธ์จากการกระทำนั้นๆส่งผลอะไรย้อนกลับมา ถ้าการกระทำนั้นส่งผลในทางบวกของปัญหานั้นๆก็จะได้รับสิ่งตอบแทนที่ดี ในทางกลับกันถ้าส่งผลไปในทางลบก็จะได้รับสิ่งตอบแทนที่ไม่ดีย้อนกลับมา จากนั้นฟังก์ชันก็จะทำการปรับค่าคิวของการกระทำที่เป็นต้นเหตุของผลลัพธ์ให้สอดคล้องกับสิ่งตอบแทนเพื่อให้รอบต่อมาที่กลับมา ณ สถานะนี้อีกครั้งก็สามารถที่จะเรียนรู้ว่าควรจะเลือกการกระทำใดจากค่าคิวที่มีการปรับค่ามาจากสิ่งที่เคยทำผ่านมา ทำให้เมื่อมีการลองผิดลองถูกไปนานๆก็ยิ่งสามารถที่จะหาการกระทำที่ดีที่สุด ณ สถานะนั้นๆของปัญหา

จะเห็นได้ว่าข้อดีของ Q-learning นั้นสามารถที่จะเลือกการกระทำที่ดีที่สุด ณ สถานะต่างๆได้จากค่าคิวของแต่ละการกระทำที่เป็นไปได้ทั้งหมดเพียงอย่างเดียว ไม่จำเป็นต้องอาศัยปัจจัยอื่นๆเข้ามาคำนึงทำให้การคำนวณเป็นไปได้อย่างรวดเร็ว ซึ่งวิธีการนี้ได้รับรองมาถึงปัจจุบันว่าเป็นวิธีการที่สามารถค้นหาวิธีการที่ดีที่สุดสำหรับปัญหา

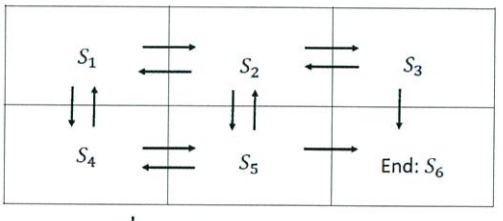
อัลกอริทึมของ Q-learning คือ

1. กำหนดค่า Q (s,a) โดยสุ่มเป็นค่าขนาดเล็ก
2. สำรวจสถานการณ์ (state, s)
3. เลือกการกระทำ (action,a) แล้วกระทำ
4. สำรวจสถานการณ์ต่อไป (s') และรางวัล (r)
5. $Q(s,a) \leftarrow (1 - \alpha)\, Q(s,a) + \alpha(r + \gamma \max Q(s',a'))$
6. กลับไปข้อ 2.

α คือ learning rate โดยมีค่า $0 \leq \alpha \leq 1$

ในทุกๆ state ของการเรียนรู้แบบเสริมแรง อัลกอริทึมจะเลือกการกระทำมาดำเนินการใน state ปัจจุบัน และคำนวณเป็นค่าในทุกๆการกระทำ

ตัวอย่าง Q learning



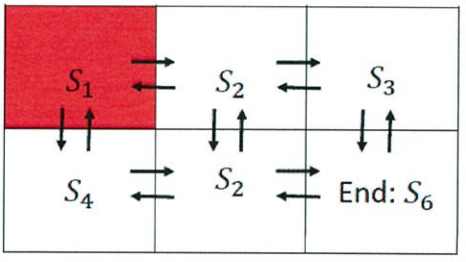
รูปที่ 2.5 ตัวอย่าง Q learning

กำหนดค่า $\gamma = 0.5$, $\alpha = 0.1$, r = 100 ถ้าเดินไปที่ S_6 และได้ 0 ถ้าเป็นอื่นๆ

ตารางที่ 2.1 ค่า Q เริ่มต้น

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	0
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เริ่มต้นเกม



รูปที่ 2.6 รูปแบบการเดินที่ 1 ของเกม

ตำแหน่งปัจจุบัน : S_1 สีแดง
การกระทำที่ทำได้ : $\rightarrow$, $\downarrow$
การกระทำที่เลือก : $\rightarrow$

อัพเดทค่า Q (S_1, $\rightarrow$) = (1- α) Q (S_1, $\rightarrow$) + α(r + γ · max (Q (S_2, $\leftarrow$) , Q (S_2, $\downarrow$) , Q(S_2, $\rightarrow$)))

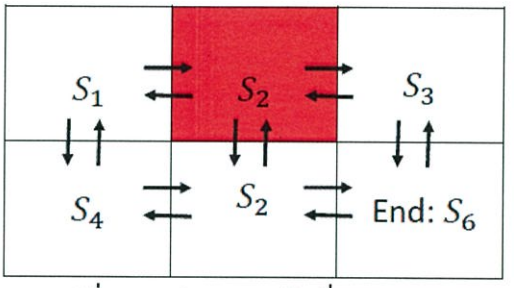
= 0 + 0.1 (0 + 0.5 · max(0,0,0))
= 0

อัพเดทค่า Q (S_1, $\rightarrow$) ในตาราง Q จะได้

ตารางที่ 2.2 การอัพเดทค่า Q (S_1, $\rightarrow$)

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	0
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เดินต่อไป



รูปที่ 2.7 รูปแบบการเดินที่ 2 ของเกม

ตำแหน่งปัจจุบัน : S_2 สีแดง

การกระทำที่ทำได้ : $\rightarrow$, $\downarrow$, $\leftarrow$

การกระทำที่เลือก : $\rightarrow$

อัพเดทค่า Q (S_2, $\rightarrow$) = (1- α) Q (S_2, $\rightarrow$) + α(r + γ $\cdot$ max (Q (S_3, $\leftarrow$) , Q (S_3, $\downarrow$)))

= 0 + 0.1 (0 + 0.5 $\cdot$ max(0,0))

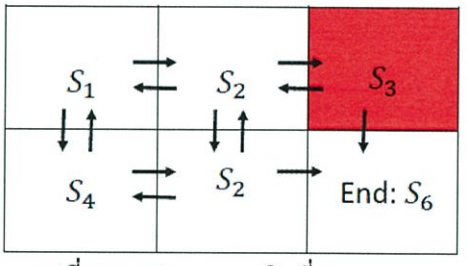
= 0

อัพเดทค่า Q (S_2, $\rightarrow$) ในตาราง Q จะได้

ตารางที่ 2.3 การอัพเดทค่า Q (S_2, $\rightarrow$)

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	0
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เดินต่อไป



รูปที่ 2.8 รูปแบบการเดินที่ 3 ของเกม

ตำแหน่งปัจจุบัน : S_3 สีแดง
การกระทำที่ทำได้ : $\leftarrow$, $\downarrow$
การกระทำที่เลือก : $\downarrow$

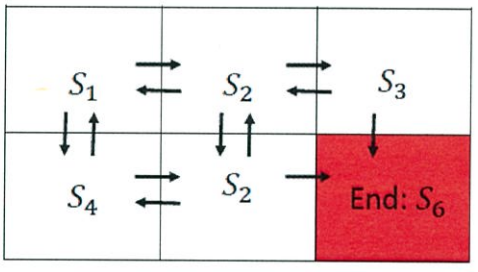
อัพเดทค่า Q (S_3, $\downarrow$) = (1- α) Q (S_3, $\downarrow$) + α(r + γ $\cdot$ max (0))
= 0 + 0.1 (100 + 0.5 $\cdot$ max(0))
= 10

อัพเดทค่า Q (S_3, $\downarrow$) ในตาราง Q จะได้

ตารางที่ 2.4 การอัพเดทค่า Q (S_3, ↓)

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	10
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เดินต่อไป

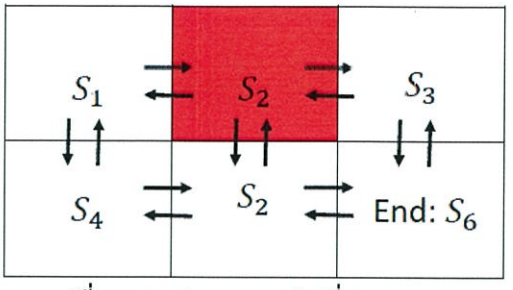


รูปที่ 2.9 รูปแบบการเดินที่ 4 ของเกม

ตำแหน่งปัจจุบัน : S_6 สีแดง

อยู่ใน final state

เริ่มต้นเกมใหม่



รูปที่ 2.10 รูปแบบการเดินที่ 5 ของเกม

ตำแหน่งปัจจุบัน : S_2 สีแดง
การกระทำที่ทำได้ : $\rightarrow$, $\downarrow$, $\leftarrow$
การกระทำที่เลือก : $\rightarrow$

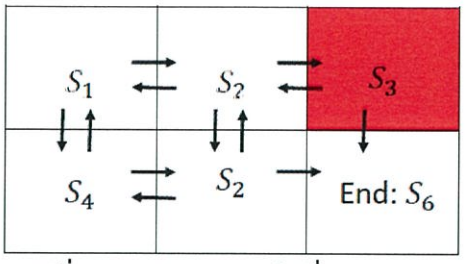
อัพเดทค่า Q (S_2, $\rightarrow$) = (1- α) Q (S_2, $\rightarrow$) + α(r + γ $\cdot$ max (Q (S_3, $\leftarrow$) , Q (S_3, $\downarrow$)))
= 0 + 0.1 (0 + 0.5 $\cdot$ max(0,10))
= 0.5

อัพเดทค่า Q (S_2, $\rightarrow$) ในตาราง Q จะได้

ตารางที่ 2.5 การอัพเดทค่า Q (S_2, $\rightarrow$)

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0.5
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	10
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เดินต่อไป



รูปที่ 2.11 รูปแบบการเดินที่ 6 ของเกม

ตำแหน่งปัจจุบัน : S_3 สีแดง
การกระทำที่ทำได้ : $\leftarrow$, $\downarrow$
การกระทำที่เลือก : $\downarrow$

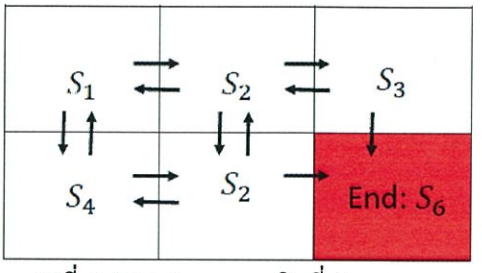
อัพเดทค่า Q (S_3, $\downarrow$) = (1- α) Q (S_3, $\downarrow$) + α(r + γ · max (0))
= (0.9)(10) + 0.1 (100 + 0.5 · max(0))
= 19

อัพเดทค่า Q (S_3, $\downarrow$) ในตาราง Q จะได้

ตารางที่ 2.6 การอัพเดทค่า Q (S_3, $\downarrow$)

$Q(s_1, \rightarrow)$	0
$Q(s_1, \downarrow)$	0
$Q(s_2, \leftarrow)$	0
$Q(s_2, \rightarrow)$	0.5
$Q(s_2, \downarrow)$	0
$Q(s_3, \leftarrow)$	0
$Q(s_3, \downarrow)$	19
$Q(s_4, \rightarrow)$	0
$Q(s_4, \uparrow)$	0
$Q(s_5, \leftarrow)$	0
$Q(s_5, \rightarrow)$	0
$Q(s_5, \uparrow)$	0

เดินต่อไป



รูปที่ 2.12 รูปแบบการเดินที่ 7 ของเกม

ตำแหน่งปัจจุบัน : S_6 สีแดง

อยู่ใน final state

2.2 งานวิจัยที่เกี่ยวข้อง

ภัทรภร วัฒนาชีพ และศิริรัตน์ บุญกว้าง [2] ได้ทำปริญญานิพนธ์เรื่อง การเรียนรู้ของเครื่องจากการทดลองนำการเรียนรู้สามอัลกอริทึมคือ อัลกอริทึม Monte Carlo, อัลกอริทึม Qlearning และ อัลกอริทึม Watkin's Q() นำมาทดลองในเกมเขาวงกตเพื่อให้ค้นหาทางออกและค้นหาระยะทางที่สั้นที่สุด ซึ่งจากบทสรุปจากการทดลองการกำหนดค่า Qเริ่มต้นและค่า e-greedy ที่ต่างกันทำให้ส่งผลต่อการเรียนรู้ด้วย จากการทดลองกำหนดค่า Q เริ่มต้นเป็น 0 และ 30 ในวิธี Monte Carlo พบว่าพัฒนาการเรียนรู้ของการทดลองที่กำหนดค่า Q เริ่มต้นเป็น 30 สามารถเรียนรู้ได้ดีมากกว่าถึงสองเท่า และจากการทดลองกำหนดค่า e-greedy ต่างกันในวิธี Qlearning กับ Watkin's Q() พบว่าการทดลองใช้ค่า 99:1 ส่งผลให้การเรียนรู้ระยะยาวมีผลการเดินที่คงที่ และเดินไปในเส้นทางที่สั้นที่สุด ส่วนการเปรียบเทียบผลการเรียนรู้ทั้งสามวิธี ทำให้พบว่า Qlearning เหมาะสมกับการเรียนรู้ในแบบจำลองมากที่สุด

พิเชษฐ์ พูลสวัสดิ์ และภวิศ ลิมป์ศิระ [3] ได้ทำปริญญานิพนธ์เรื่อง การเรียนรู้ของเครื่อง ได้กล่าวว่า การเรียนรู้เสริมกำลังเป็นหลักการทางการเรียนรู้ของเครื่องซึ่งเลียนแบบพฤติกรรมการเรียนรู้ของสิ่งมีชีวิต ซึ่งการเรียนรู้เสริมกำลังนั้นสามารถนำมาประยุกต์ใช้ในชีวิตประจำวันได้มากมาย ไม่ใช่แค่นำมาประยุกต์การแก้ปัญหาเขาวงกตเท่านั้น แต่ยังสามารถนำไปประยุกต์ใช้ก็ตามแค่ต้องสามารถที่จะกำหนดกฎเกณฑ์และผลลัพธ์ที่ต้องการและให้เวลาเครื่องได้ทำการเรียนรู้ลองผิดลองถูกเพื่อให้เครื่องได้เรียนรู้รวมถึงสร้างสถานการณ์ใหม่ๆจนได้การกระทำที่ดีที่สุดเพื่อใช้ในการแก้ปัญหา และจากการทดลองในเกมเขาวงกตพบว่ารูปแบบการเรียนรู้ที่ใช้มีความสามารถในการเรียนรู้ และหาเส้นทางที่ใช้ระยะทางที่สั้นที่สุดได้ แต่สามารถลดเวลาที่ใช้ในการเรียนรู้ได้โดยใช้กฎที่สร้างขึ้นจากการเรียนรู้ที่

ผ่านมาเป็นความรู้มาเป็นตัวเลือกในสถานะที่ไม่เคยเจอทำให้สามารถลดเวลาในการเรียนรู้เพื่อหาทางแก้ไขปัญหาแรกได้แต่ในกรณีที่ต้องการทางแก้ปัญหาที่ดีที่สุดนั้นยังต้องใช้เวลาอยู่แต่ก็ยังใช้เวลาน้อยกว่าแบบไม่ใช้กฎดังกล่าว

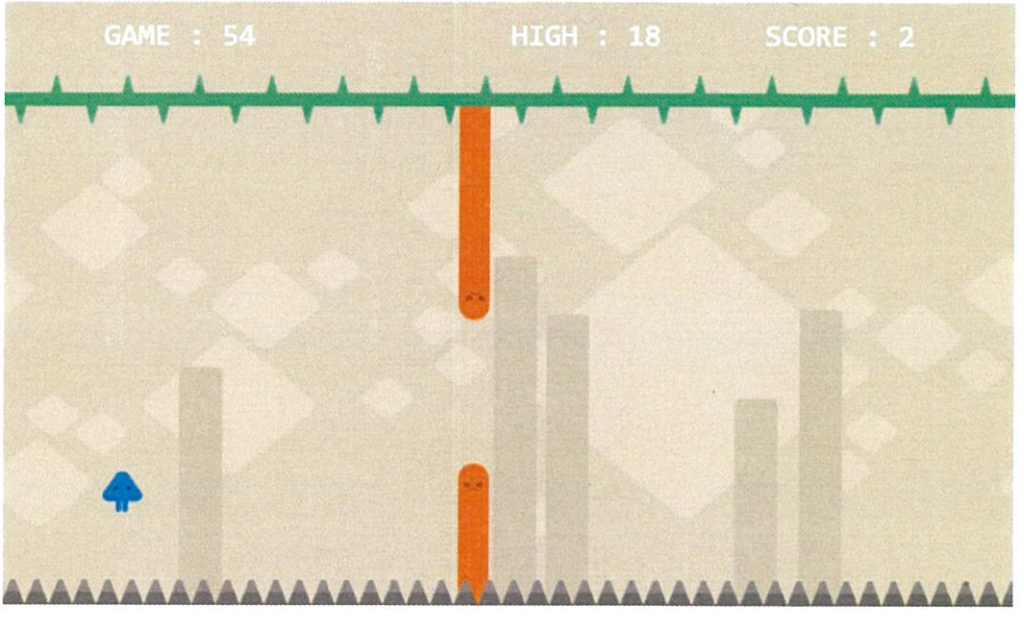
บทที่ 3

การออกแบบและพัฒนา

ในบทนี้จะกล่าวถึงเนื้อหาขั้นตอนการออกแบบและพัฒนาเอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง โดยหัวข้อที่จะกล่าวถึงคือ รูปแบบของเกม องค์ประกอบส่วนต่างๆที่จะใช้ในการเรียนรู้ของเอเจนท์ และอธิบายถึงฟังก์ชันที่ใช้

3.1 รูปแบบของเกม

เกมที่ใช้จะเป็นเกมแนว Endless-Jumper ที่มีรูปแบบให้กดกระโดดโดยจะต้องคอยกระโดดอยู่ตลอดเพื่อประคองไม่ให้ตัวละครตกพื้นตายแต่ถ้ากดกระโดดมากเกินไปตัวละครก็จะชนหนามด้านบนตายเช่นกันและยังมีอุปสรรคระหว่างทางที่จะต้องคอยหลบโดยลอดช่องว่างระหว่างอุปสรรคหน้าตาและองค์ประกอบภายในตัวเกมมีดังนี้



รูปที่ 3.1.1 หน้าตาภายในเกม

1. ตัวเอเจนท์

รูปที่ 3.1.2 ตัวเอเจนท์

2. อุปสรรคด้านบน

เมื่อเอเจนท์กดกระโดดมากจนเกินไปก็จะชนสิ่งกีดขวางด้านบนทำให้เอเจนท์ตาย และเริ่มต้นเกมใหม่

รูปที่ 3.1.3 อุปสรรคด้านบน

3. อุปสรรคด้านล่าง

เมื่อเอเจนท์ปล่อยตัวร่วงลงมาด้านล่างจะทำให้เอเจนท์ตาย และเริ่มต้นเกมใหม่

รูปที่ 3.1.4 อุปสรรคด้านล่าง

4. อุปสรรคระหว่างทาง

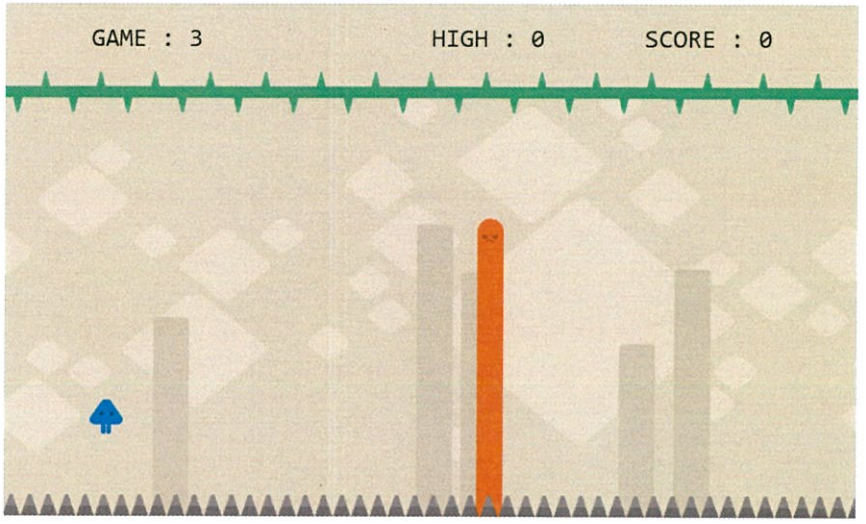
จะมีสิ่งกีดขวางขนาดความยาวต่างๆระหว่างทางให้เอเจนท์ต้องหลบ และเมื่อชนเอเจนท์จะตาย และเริ่มต้นเกมใหม่

รูปที่ 3.1.5 อุปสรรคระหว่างทาง

3.1.1 รูปแบบของอุปสรรคระหว่างทาง

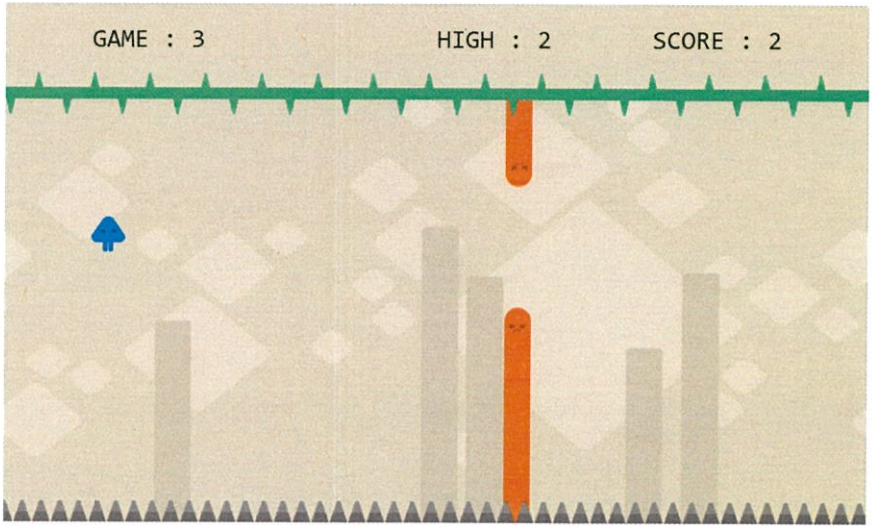
อุปสรรคระหว่างทางจะมีรูปแบบที่แตกต่างกันอยู่ 4 รูปแบบที่มีความสูงของช่องว่างแตกต่างกัน เอเจนท์จะต้องเรียนรู้ที่จะเล่นให้ผ่าน

รูปแบบที่ 1



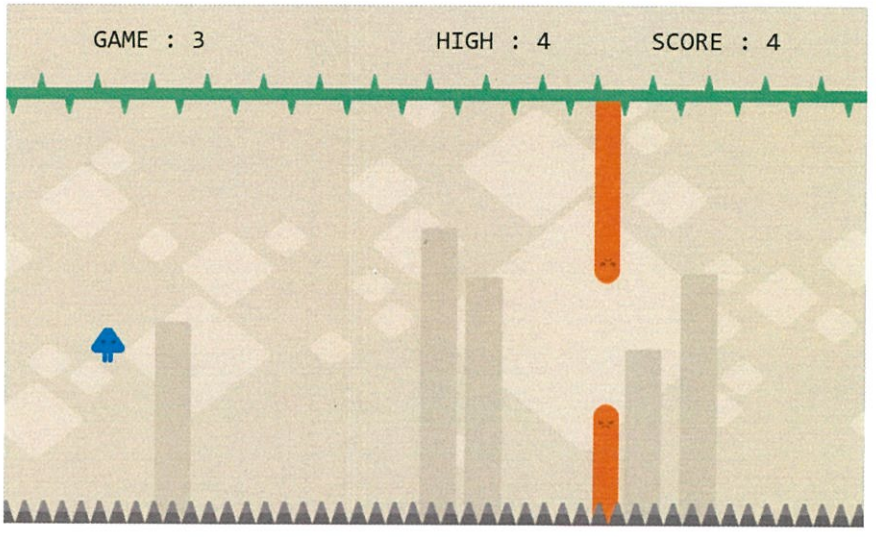
รูปที่ 3.1.6 อุปสรรคระหว่างทางรูปแบบที่ 1

รูปแบบที่ 2



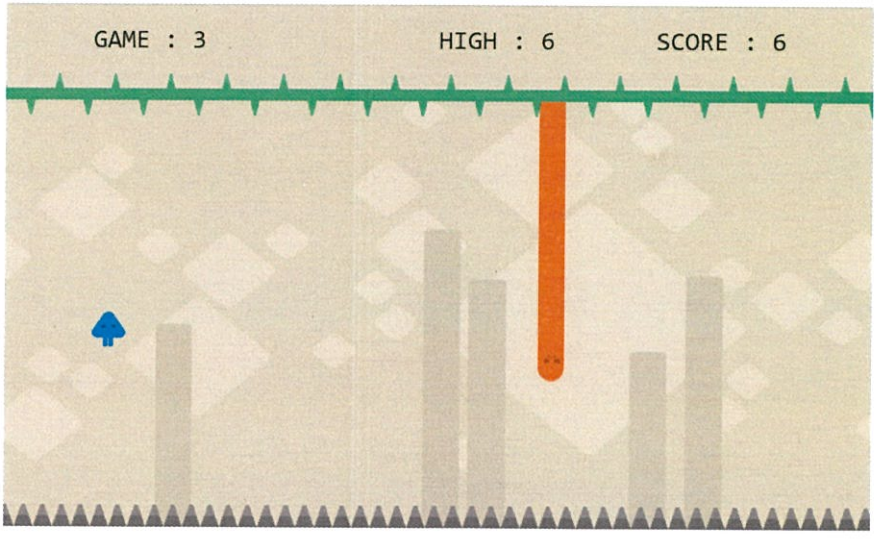
รูปที่ 3.1.7 อุปสรรคระหว่างทางรูปแบบที่ 2

รูปแบบที่ 3



รูปที่ 3.1.8 อุปสรรคระหว่างทางรูปแบบที่ 3

รูปแบบที่ 4



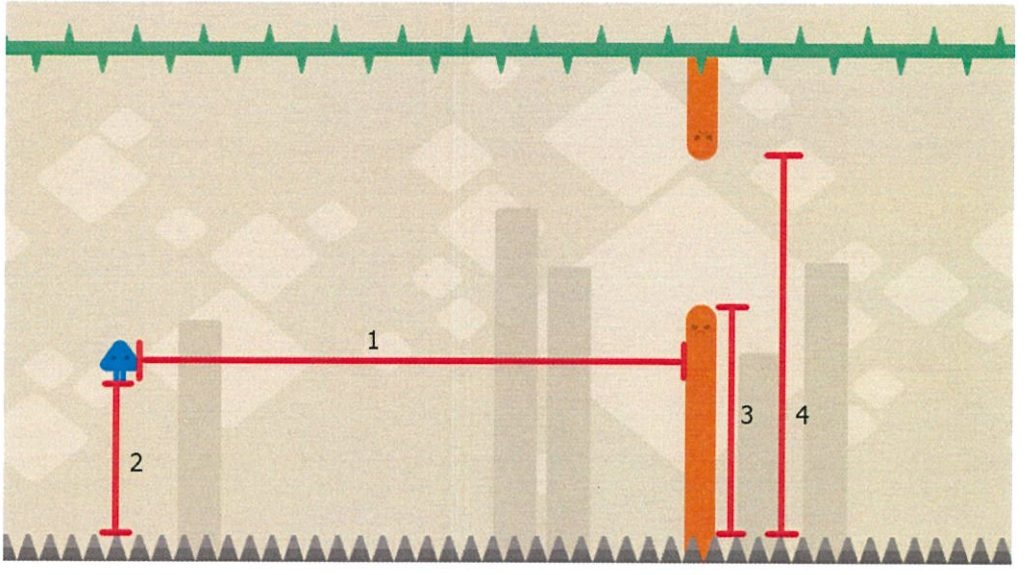
รูปที่ 3.1.9 อุปสรรคระหว่างทางรูปแบบที่ 4

ลำดับในของอุปสรรคระหว่างทางที่ออกมาจะเรียงตามรูปแบบคือ 1,2,3 และ 4 ตามลำดับหลังจากนั้นจะเริ่มที่รูปแบบที่ 1 ใหม่อีกครั้งและวนต่อไปเรื่อยๆ

3.2 การออกแบบองค์ประกอบส่วนต่างๆที่จะใช้ในการเรียนรู้ของเอเจนท์

ในการจะสอนให้เอเจนท์สามารถเรียนรู้ได้ ต้องมีการออกแบบสถานะ(State) การกระทำ (Action) และรางวัล (Reward) ให้กับเอเจนท์ ซึ่งจะกำหนดดังนี้

3.2.1 การกำหนดสถานะ (State)



รูปที่ 3.2.1 การกำหนดสถานะ (State)

เพื่อให้เอเจนท์สามารถเรียนรู้ที่จะเล่นเกมได้ผ่าน จะต้องมีการกำหนดสถานะให้กับเอเจนท์โดยการออกแบบสถานะจะแตกต่างกันไปในแต่ละเกมขึ้นอยู่กับสภาพแวดล้อมภายในเกมที่มีผลต่อการเล่นเกม โดยในเกมนี้เราได้กำหนดสถานะให้เอเจนท์โดยดูจากอุปสรรคระหว่างทางและตำแหน่งที่เอเจนท์อยู่เป็นหลัก ซึ่งแบ่งเป็น 4 ส่วนดังรูปที่ 3.2.1 คือ

1. ระยะทางในแนวแกน x ระหว่างตัวเอ-เจนท์กับอุปสรรคระหว่างทางที่อยู่ใกล้ที่สุด

2. ระยะทางในแนวแกน y ระหว่างเอเจนท์กับอุปสรรคด้านล่าง

3. ตำแหน่งปลายด้านล่างของช่องหว่างระหว่างอุปสรรค

4. ตำแหน่งปลายด้านบนของช่องว่างระหว่างอุปสรรค เมื่อเอเจนท์ทำการเคลื่อนที่หรืออุปสรรคที่เจอเปลี่ยนไป สถานะก็จะเปลี่ยนแปลงตามไปด้วย

3.2.2 การกระทำ (Action)

การกระทำภายในเกมจะมีทั้งหมด 2 แบบคือ กระโดดและอยู่เฉย โดยเมื่อเอเจนท์เลือกการกระโดดก็จะลอยตัวขึ้นสูงเล็กน้อย และเมื่อเลือกการกระทำอยู่เฉย เอเจนท์จะร่วงลงมาข้างล่าง

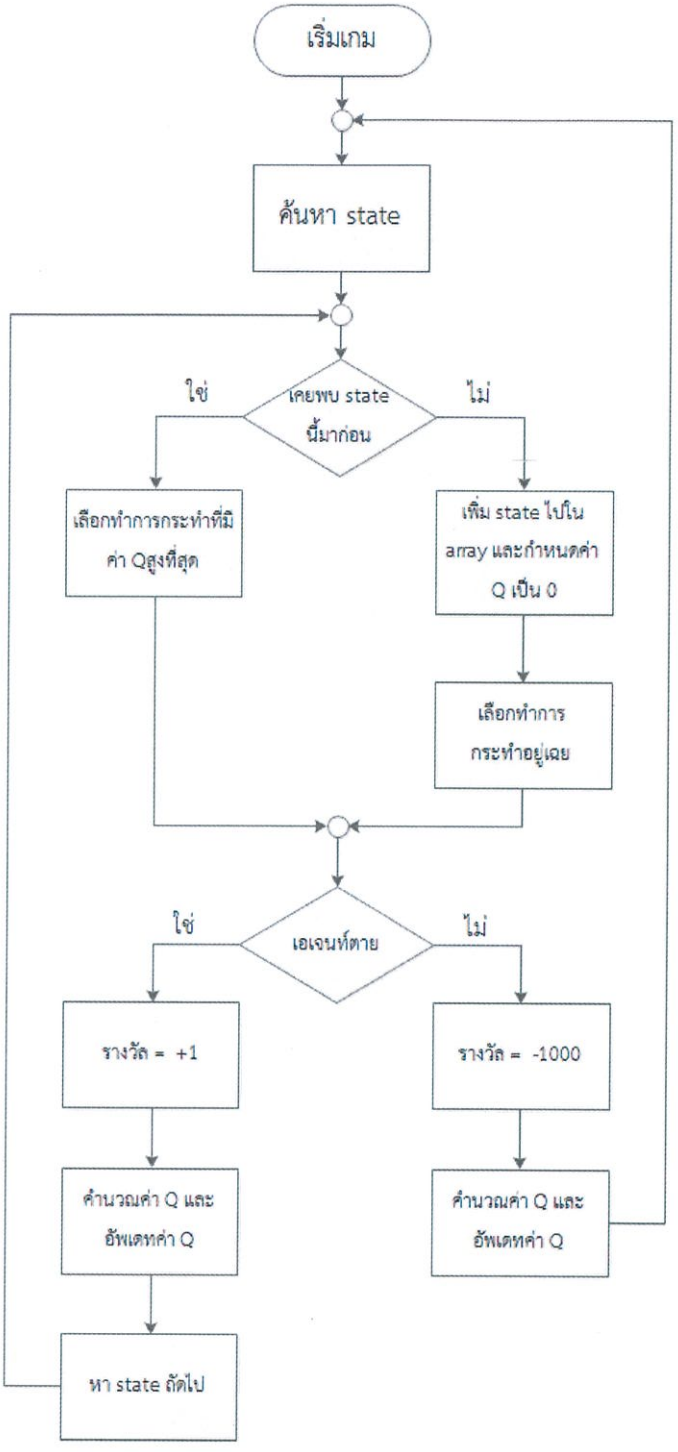
3.2.3 การกำหนดรางวัล (Reward)

รางวัลที่ให้กับเอเจนท์เมื่อเอเจนท์กระทำในสถานะต่างๆได้ถูกต้องจะมีการให้รางวัลและจะให้รางวัลเป็นลบเมื่อเอเจนท์กระทำผิดในสถานะต่างๆ ซึ่งในที่นี้ได้กำหนดให้เอเจนท์จะได้รางวัลดังนี้

รางวัลการมีชีวิตอยู่เป็น +

รางวัลเมื่อเอเจนท์ตายเป็น -

3.3 ผังงานการทำงานของเอเจนท์



รูปที่ 3.3.1 ผังการทำงานของเอเจนท์

จากรูปที่ 3.3.1 แสดงลำดับการทำงานของเอเจนท์ เมื่อเริ่มเกมเอเจนท์จะหาสถานะ (State) โดยดูจากอุปสรรคต่างๆ และเลือกทำการกระทำที่ดีที่สุดในสถานะนั้นๆ หากเพิ่งเคยพบสถานะนี้ เอเจนท์จะเลือกทำการกระทำอยู่เฉย เมื่อเอเจนท์กระทำการกระทำใดๆก็จะได้รับรางวัลเพื่อนำไปคำนวณค่า Q และเข้าสู่สถานะถัดไป เอเจนท์จะอัพเดทค่า Q ไปเรื่อยๆจนได้การกระทำที่ดีที่สุดสำหรับทุกๆสถานะ

3.4 ฟังก์ชันที่ใช้

หัวข้อนี้จะแสดงและอธิบายฟังก์ชันต่างๆที่เกี่ยวข้องกับการเรียนรู้ของเอเจนท์โดยจะมีหัวข้อดังนี้ getQ, findState, getActionWithHighestQ และ think

3.4.1 getQ

ฟังก์ชันนี้เป็นฟังก์ชันที่ใช้สำหรับดูค่า Q ของ action ใน state ต่างๆ มีการรับพารามิเตอร์ 1 ตัวคือ state ฟังก์ชันจะเริ่มการทำงานโดยทำการเช็คก่อนว่า state ที่รับเข้ามานั้นมีอยู่แล้วใน array Q หรือไม่ ถ้าหากยังไม่มี state นั้นอยู่ใน array ก็จะทำการเพิ่ม state นั้นเข้าไปโดยกำหนดค่า Q เริ่มต้นของทุกๆ action สำหรับ state นั้นเป็น 0 และทำการคืนค่า Q กลับไป แต่ถ้าหากเช็คแล้วพบว่ามี state นั้นอยู่ใน array อยู่แล้วก็จะทำการคืนค่า Q กลับไปตามปกติ

```
getQ: function(state){
    if (!this.Q.hasOwnProperty(state)) {
        this.Q[state] = {};

        for (var actionIndex = 0; actionIndex < this.actions.length; actionIndex++) {
            var action = this.actions[actionIndex];

            this.Q[state][action] = 0;
        }
    }
    return this.Q[state];
},
```

รูปที่ 3.4.1 ฟังก์ชัน getQ

3.4.2 findState

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับหา state ต่างๆซึ่งจะพิจารณาจากตัวละครที่เอเจนท์ทำการเล่นกับอุปสรรคต่างๆที่เอเจนท์ต้องเล่นให้ผ่าน โดยการกำหนด state นั้นจะแบ่งออกเป็นสี่ส่วนได้แก่ ระยะทางในแนวแกน x ระหว่างตัวละครกับอุปสรรคที่อยู่หน้าสุด, ระยะทางในแนวแกน y ระหว่างตัวละครกับพื้นด้านล่าง, ระยะทางในแนวแกน y ระหว่างจุดปลายของอุปสรรคด้านล่างกับพื้น, ระยะทางในแนวแกน y ระหว่างจุดปลายของอุปสรรค

ด้านบนกับพื้น เมื่อคำนวณระยะทางทั้งสี่ส่วนแล้วฟังก์ชันจะทำการคืนค่ากลับไป โดยจะทำการใช้ “.” เป็นตัวแยกระหว่างแต่ละส่วนเพื่อไม่ทำให้ตัวเลขชนกันซึ่งอาจทำให้การคำนวณในภายหลังผิดพลาดได้ นอกจากนั้นยังได้ทำการคูณค่า resolution ซึ่งกำหนดไว้เท่ากับ 0.01 เพื่อทำการลดจำนวน state ทั้งหมดลงเพื่อช่วยลดระยะเวลาที่เอเจนท์จะทำการสำรวจจนพบ state ทั้งหมด

```
findState: function(){
    var closest = -1;
    var botPipeY = 0;
    var topPipeY = 0;
    for (var blockIndex = 0; blockIndex < jumpOb.length; blockIndex++) {
        var block = jumpOb[blockIndex];
        var botBlock = jumpOb2[0];

        var distanceY = botBlock.y;

        var distance = block.x - this.x;

        topPipeY = jumpOb[0].height + ceiling.y + ceiling.height



        if (distance >= 0 && (closest === -1 || distance < closest)) {
            closest = distance;
        }
        botPipeY = distanceY;

    }

    return Math.max(0, Math.floor(closest * resolution)) + "." + Math.floor(this.y * resolution) + "." + Math.floor(botPipeY *
        resolution) + "." + Math.floor(topPipeY * resolution);
},
```

รูปที่ 3.4.2 ฟังก์ชัน findState

3.4.3 getActionWithHighestQ

ฟังก์ชันนี้เป็นฟังก์ชันที่จะเลือกการกระทำโดยพิจารณาจากค่า Q ใน array มีการรับพารามิเตอร์ 1 ตัว คือ distance ซึ่งก็คือ state เมื่อรับมาแล้วก็จะนำไปดูค่า Q ใน array โดยใช้ฟังก์ชัน getQ และนำค่า Q ของทุกๆ action ใน state นั้นมาเปรียบเทียบกัน โดยฟังก์ชันจะคืนค่าเป็น action ที่มีค่า Q มากที่สุดกลับไป

```
getActionWithHighestQ: function(distance){

    var jumpReward = this.getQ(distance)[this.actions[0]];
    var doNothingReward = this.getQ(distance)[this.actions[1]];

    if (jumpReward > doNothingReward) {
        return this.actions[0];
    } else if (doNothingReward > jumpReward) {
        return this.actions[1];
    } else {
        return this.actions[1];
    }

},
```

รูปที่ 3.4.3 ฟังก์ชัน getActionWithHighestQ

3.4.4 think

ฟังก์ชันนี้เป็นฟังก์ชันสำหรับคำนวณค่า Q ซึ่งในการคำนวณจะใช้ค่า reward ที่กำหนดไว้ซึ่งกำหนดไว้ว่าทุกๆ state ที่เอเจนท์ยังมีชีวิตอยู่ก็จะได้รางวัลเป็น +1 แต่ถ้าหาก state ไหนที่เอเจนท์ตายก็จะได้รางวัลเป็น -1000 แทน และนำรางวัลนั้นไปคำนวณเป็นค่า Q และอัพเดทเข้าไปใน array ที่เก็บค่า Q ของ action ใน state นั้นๆ

```
think: function(){
    var reward = 0;

    var distance = this.findState(),
        maxQ = this.getQ(distance)[this.getActionWithHighestQ(distance)],
        previousQ = this.getQ(this.lastDistance)[this.lastAction];

    if(this.playerAlive){
        reward = 1;
        this.getQ(this.lastDistance)[this.lastAction] = ((1 - this.alpha) * previousQ) + (this.alpha * (reward + (this.gamma * maxQ))
            );
        this.lastAction = this.getActionEpsilonGreedy();


        this.lastDistance = distance;
    }

    if(!this.playerAlive){
        reward = -1000;

        this.getQ(this.lastDistance)[this.lastAction] = ((1 - this.alpha) * previousQ) + (this.alpha * (reward + (this.gamma * maxQ))
            );
        this.playerAlive = true;
        this.lastAction = this.getActionEpsilonGreedy();
        this.lastDistance = distance;

    }

    switch (this.lastAction) {
        case this.actions[0]:
            this.jump();
            break;
    }
},
```

รูปที่ 3.4.4 ฟังก์ชัน think

บทที่ 4

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองและผลการทดลองที่ได้ โดยจะทำการทดลองประสิทธิภาพการเรียนรู้ของเอเจนท์ ในการทดลองจะให้เอเจนท์เล่นเกมโดยมีการกำหนดลำดับอุปสรรคให้เหมือนกันทุกๆครั้งในแต่ละรอบการเล่น และทำการเปรียบเทียบประสิทธิภาพโดยทำการปรับค่าในอัลกอริทึม โดยค่าที่จะปรับในอัลกอริทึมจะมีค่า α (Learning rate) และ γ (Discount factor) จากนั้นทำการสังเกตจำนวนเกมที่เอเจนท์สามารถเล่นเกมได้ดีหรือเล่นจนได้คะแนนมากกว่า 100 คะแนนและหากเอเจนท์ทำการเล่นไปมากกว่า 200 เกม แต่ยังทำคะแนนได้ไม่ถึง 100 คะแนนจะดูคะแนนมากที่สุดที่เอเจนท์สามารถทำได้และบันทึกผล รางวัลที่ทำการให้เอเจนท์คือเมื่อเอเจนท์มีชีวิตอยู่จะได้รางวัล +1 และเมื่อเอเจนท์ตายจะได้รางวัล -1000

4.1 การทดลอง

จะทำการปรับค่า α และ γ ครั้งละ 0.2 และดูจำนวนเกมที่เอเจนท์ทำการเล่นจนสามารถเล่นได้ดี

4.1.1 α = 0.2

ตารางที่ 4.1 จำนวนเกมและคะแนนที่เอเจนท์เล่นได้ที่ α = 0.2

α	γ	จำนวนเกมที่เล่น	คะแนนสูงสุดที่ทำได้
0.2	0.2	มากกว่า 200 เกม	0 คะแนน
0.2	0.4	มากกว่า 200 เกม	0 คะแนน
0.2	0.6	มากกว่า 200 เกม	0 คะแนน
0.2	0.8	มากกว่า 200 เกม	12 คะแนน
0.2	1.0	101 เกม	มากกว่า 100 คะแนน

จากตารางที่ 4.1 ที่ค่า α = 0.2 เมื่อลองปรับค่า γ เท่ากับ 0.2, 0.4 และ 0.6 และทำการปล่อยให้เอเจนท์เล่นไปมากกว่า 200 เกม เอเจนท์ยังไม่สามารถเล่นเกมได้โดยไม่สามารถทำคะแนนได้เลย และเมื่อปรับค่า γ เท่ากับ 0.8 และปล่อยให้เอเจนท์เล่นไปมากกว่า 200 เกมเอเจนท์สามารถทำคะแนนสูงสุดได้ 12 คะแนน และ เมื่อปรับค่า γ เท่ากับ 1.0 เมื่อเอเจนท์เล่นเกมไป 101 เกมก็สามารถทำคะแนนได้มากกว่า 100 คะแนน

4.1.2 α = 0.4

ตารางที่ 4.2 จำนวนเกมและคะแนนที่เอเจนท์เล่นได้ที่ α = 0.4

α	γ	จำนวนเกมที่เล่น	คะแนนสูงสุดที่ทำได้
0.4	0.2	มากกว่า 200 เกม	0 คะแนน
0.4	0.4	มากกว่า 200 เกม	0 คะแนน
0.4	0.6	มากกว่า 200 เกม	6 คะแนน
0.4	0.8	มากกว่า 200 เกม	18 คะแนน
0.4	1.0	58 เกม	มากกว่า 100 คะแนน

จากตารางที่ 4.2 ที่ค่า α = 0.4 เมื่อปรับค่า γ เท่ากับ 0.2 และ 0.4 และปล่อยให้เอเจนท์เล่นเกมไปมากกว่า 200 เกมแล้ว เอเจนท์ยังไม่สามารถทำคะแนนได้ ที่ค่า γ เท่ากับ 0.6 เอเจนท์สามารถทำคะแนนได้ 6 คะแนนหลังจากเล่นไปมากกว่า 200 เกม ที่ค่า γ เท่ากับ 0.8 เอเจนท์สามารถทำคะแนนได้ 18 คะแนนหลังจากเล่นไปมากกว่า 200 เกม และที่ค่า γ เท่ากับ 1.0 เอเจนท์สามารถทำได้มากกว่า 100 คะแนนหลังจากเล่นไปได้ 58 เกม

4.1.3 α = 0.6

ตารางที่ 4.3 จำนวนเกมและคะแนนที่เอเจนท์เล่นได้ที่ α = 0.6

α	γ	จำนวนเกมที่เล่น	คะแนนสูงสุดที่ทำได้
0.6	0.2	มากกว่า 200 เกม	0 คะแนน
0.6	0.4	มากกว่า 200 เกม	6 คะแนน
0.6	0.6	มากกว่า 200 เกม	12 คะแนน
0.6	0.8	59 เกม	มากกว่า 100 คะแนน
0.6	1.0	113 เกม	มากกว่า 100 คะแนน

จากตารางที่ 4.3 ที่ค่า α = 0.6 เมื่อปรับค่า γ เท่ากับ 0.2 เอเจนท์ไม่สามารถทำคะแนนได้หลังจากเล่นไปมากกว่า 200 เกม ที่ค่า γ เท่ากับ 0.4 เอเจนท์สามารถทำคะแนนได้ 6 คะแนนหลังจากเล่นไปมากกว่า 200 เกม ที่ค่า γ เท่ากับ 0.6 เอเจนท์สามารถทำคะแนนได้ 12 คะแนนหลังจากเล่นไปมากกว่า 200 เกม ที่ค่า γ เท่ากับ 0.8 และ 1.0 เอเจนท์สามารถเล่นเกมได้มากกว่า 100 คะแนน โดย γ เท่ากับ 0.8 ใช้จำนวนการเล่น 59 เกม และ γ เท่ากับ 1.0 ใช้จำนวนการเล่น 113 เกม

4.1.4 α = 0.8

ตารางที่ 4.4 จำนวนเกมและคะแนนที่เอเจนท์เล่นได้ที่ α = 0.8

α	γ	จำนวนเกมที่เล่น	คะแนนสูงสุดที่ทำได้
0.8	0.2	มากกว่า 200 เกม	0 คะแนน
0.8	0.4	38 เกม	มากกว่า 100 คะแนน
0.8	0.6	38 เกม	มากกว่า 100 คะแนน
0.8	0.8	44 เกม	มากกว่า 100 คะแนน
0.8	1.0	มากกว่า 200 เกม	38 คะแนน

จากตารางที่ 4.4 ที่ค่า α = 0.8 ที่ค่า γ เท่ากับ 0.2 เอเจนท์ยังไม่สามารถทำคะแนนได้หลังจากทำการเล่นไปมากกว่า 200 เกม ที่ค่า γ เท่ากับ 0.4 และ 0.6 เอเจนท์สามารถเล่นได้มากกว่า 100 คะแนนหลังจากทำการเล่นไปได้ 38 เกม ซึ่งใช้จำนวนการเล่นเท่ากันระหว่าง γ เท่ากับ 0.4 และ γ เท่ากับ 0.6 ที่ γ เท่ากับ 0.8 เอเจนท์สามารถเล่นได้มากกว่า 100 คะแนน หลังจากทำการเล่นเกม 44 เกม และที่ γ เท่ากับ 1.0 เอเจนท์ไม่สามารถเล่นได้ถึง 100 คะแนน โดยทำคะแนนสูงสุดได้ 38 คะแนนหลังจากเล่นไปมากกว่า 200 เกม

4.1.5 α = 1.0

ตารางที่ 4.5 จำนวนเกมและคะแนนที่เอเจนท์เล่นได้ที่ α = 1.0

α	γ	จำนวนเกมที่เล่น	คะแนนสูงสุดที่ทำได้
1.0	0.2	มากกว่า 200 เกม	4 คะแนน
1.0	0.4	มากกว่า 200 เกม	4 คะแนน
1.0	0.6	มากกว่า 200 เกม	4 คะแนน
1.0	0.8	125 เกม	มากกว่า 100 คะแนน
1.0	1.0	มากกว่า 200 เกม	4 คะแนน

จากตารางที่ 4.5 ที่ค่า α = 1.0 ทำการปรับค่า γ เท่ากับ 0.2, 0.4 และ 0.6 เอเจนท์สามารถทำคะแนนได้เพียง 4 คะแนน หลังจากทำการเล่นเกมไปมากกว่า 200 เกม ที่ γ เท่ากับ 0.8 เอเจนท์สามารถเล่นได้มากกว่า 100 คะแนน หลังจากทำการเล่นเกมไป 125 เกม และที่ γ เท่ากับ 1.0 เอเจนท์ก็สามารถทำคะแนนได้เพียง 4 คะแนน หลังจากเล่นไปมากกว่า 200 เกม

4.2 ผลการทดลอง

จากการทดลองจะเห็นว่าการปรับค่า α และ γ มีผลต่อการเรียนรู้ของเอเจนท์ จะเห็นว่าการเพิ่มค่า α และ γ สามารถทำให้เอเจนท์สามารถเรียนรู้ได้ดีขึ้น โดยสามารถทำคะแนนได้ดีขึ้นและถ้าหากเอเจนท์สามารถเล่นจนได้มากกว่า 100 คะแนน การปรับค่าทั้งสองก็สามารถทำให้จำนวนรอบในการเล่นจนเอเจนท์สามารถเล่นได้มากกว่า 100 คะแนน ลดลงได้ ที่ค่า α เท่ากับ 0.2, 0.4 และ 0.6 การเพิ่มค่า γ สามารถทำให้เอเจนท์เรียนรู้ได้ดียิ่งขึ้น แต่ที่ α เท่ากับ 0.8 และ 1.0 การเพิ่มค่า γ มากจนเกินไปจะส่งผลให้เอเจนท์ไม่สามารถเล่นเกมได้

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากการทดลองและศึกษาการเรียนรู้แบบเสริมแรง เป็นการเรียนรู้ที่สามารถนำมาประยุกต์ใช้และให้ประโยชน์ในงานต่างๆได้มาก ซึ่งหลักการของการเรียนรู้แบบเสริมแรงคือการค้นหาสถานะ (state) การกระทำ (action) และการให้รางวัล (reward) การค้นหาสถานะ เป็นสิ่งที่ทำได้ยากต้องอาศัยการศึกษาและค้นหาวิธีการในการหาสถานะที่ดีที่สุดมาใช้ในงานนั้น และเมื่อสามารถค้นหาสถานะ กำหนดการกระทำ และให้รางวัล ได้แล้วงานที่เหลือคือการให้เวลากับเครื่องในการลองผิดลองถูก ค้นหาวิธีการที่ดีที่สุด ซึ่งเราสามารถศึกษาการกระทำของเครื่องได้เมื่อเครื่องทำได้ดีแล้ว เป็นการช่วยลดการทำงานของคนได้ เพราะไม่จำเป็นต้องมาค้นหาวิธีการที่ดีที่สุดเอง สามารถสอนให้กับเครื่องแล้วรอให้เครื่องเรียนรู้และศึกษาหาวิธีการการทำงานที่ดีที่สุดมาให้

จากการทดลองการปรับค่า α (Learning rate) และค่า γ (Discount factor) พบว่าการปรับค่าทั้งสองนั้นมีผลต่อการเรียนรู้ของเอเจนท์ ซึ่งการเพิ่มค่า α และ γ สามารถทำให้เอเจนท์เรียนรู้ได้ดีขึ้นได้ แต่ก็มีในบางกรณีเช่นที่ค่า α เท่ากับ 0.8 การเพิ่มค่า γ มากเกินไปก็จะทำให้เอเจนท์ไม่สามารถเรียนรู้ได้ดีเช่นกัน และที่ค่า α เท่ากับ 1.0 เอเจนท์สามารถเล่นได้เกิน 100 คะแนนด้วยค่า γ เพียงค่าเดียวคือ 0.8 และยังใช้จำนวนเกมที่เล่นมากถึง 125 เกม

5.2 ปัญหาและอุปสรรค

1. ทฤษฎีที่ทำการศึกษานั้นไม่มีกฎเกณฑ์ตายตัวว่าจะต้องทำการออกแบบหรือเขียนโปรแกรมอย่างไรเพราะมีแค่รูปแบบแนวคิด และแนวทางในการแก้ไขปัญหาเท่านั้น ทำให้ต้องศึกษาแนวคิดให้เข้าใจและสมาชิกในกลุ่มช่วยกันเรียนรู้และเรียบเรียงวิธีการเพื่อใช้ในการออกแบบและเขียนโปรแกรมตามความรู้ที่เรียนมา

2. ปัญหาในค้นหาสถานะ ในการสร้างสถานะ ถ้าไม่มีการกำหนดที่ดีจะทำให้เอเจนท์ไม่สามารถเรียนรู้ได้ หรือใช้เวลานานในการเรียนรู้ จึงจำเป็นต้องให้ความสำคัญกับการหาสถานะที่เหมาะสมเป็นสิ่งสำคัญที่สุด

3. ปัญหาการให้รางวัลกับเอเจนท์ การให้รางวัลต้องมีการกำหนดการให้ที่เหมาะสมและถูกต้อง ถ้าให้กำหนดการให้รางวัลกับเอเจนท์ไม่ดีจะทำให้เอเจนท์เรียนรู้ไม่ถูกต้องและกระทำไม่ถูกต้อง

5.3 ข้อเสนอแนะ

ปัญหาพิเศษในหัวข้อ “เอเจนท์เล่นเกมที่สามารถปรับตัวเองได้โดยใช้การเรียนรู้แบบเสริมแรง” เป็นปัญหาพิเศษที่เป็นการเริ่มต้นในการใช้การเรียนรู้แบบเสริมแรงในเกม ซึ่งเนื่องจากในช่วงเวลาที่จำกัด ทำให้ไม่ได้มีการทดสอบในเกมรูปแบบอื่น ดังนั้นจึงมีแนวทางสำหรับผู้ที่จะศึกษาและพัฒนาต่อดังนี้

1. การพัฒนาเกมให้มีความยากขึ้น เพื่อทดสอบว่าเอเจนท์สามารถเล่นได้ผ่านหรือไม่
2. การนำการเรียนรู้แบบเสริมแรงไปใช้ในเกมอื่นเพื่อค้นหารูปแบบการหาสถานะและการกระทำอื่นๆ

เอกสารอ้างอิง

[1] Richard S., Sutton. **"Reinforcement Learning."** [Online]. Available: http://webdocs.cs.ual.berta.ca/~sutton/book/ebook/the-book.html. 2012

[2] ภัทรภร วัฒนาชีพ, ศิริรัตน์ บุญกว้าง. **"การเรียนรู้ของเครื่อง"** ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2555

[3] พิเชษฐ์ พูลสวัสดิ์, ภวิศ ลิ้มปศิระ. **"การเรียนรู้ของเครื่อง"** ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2556