การเรียนรู้แบบเสริมแรงเชิงลึกสำหรับการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ

DEEP REINFORCEMENT LEARNING FOR FOREX TRADING

นายธัชชัย สมุทรโสภากุล
TATCHAI SAMUTSOPAKUL

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2565
KMITL-2022-SC-M-017-99

DEEP REINFORCEMENT LEARNING FOR FOREX TRADING

TATCHAI SAMUTSOPAKUL

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
DATA SCIENCE AND ANALYTICS
KMITL-DIGITAL ANALYTICS AND INTELLIGENCE CENTER, SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2022
KMITL-2022-SC-M-017-99

หัวข้อการค้นคว้าอิสระ การเรียนรู้แบบเสริมแรงเชิงลึกสำหรับการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ

ชื่อนักศึกษา นายธัชชัย สมุทรโสภากุล

รหัสประจำตัว 63605096

ปริญญา วิทยาศาสตรมหาบัณฑิต (วิทยาการข้อมูลและการวิเคราะห์) ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง

พ.ศ. 2565

อาจารย์ที่ปรึกษาการค้นคว้าอิสระ รองศาสตราจารย์ ดร.ละออ บุญเกษม

บทคัดย่อ

ปัจจุบันคนรุ่นใหม่มีความสนใจในการลงทุนทรัพย์สินทางการเงินมากยิ่งขึ้น ไม่ว่าจะเป็นการซื้อขายหุ้น คู่สกุลเงินหรือแม้แต่คริปโทเคอร์เรนซี แต่ในความจริงน้อยคนนักที่จะประสบความสำเร็จในการลงทุนกับสินทรัพย์ประเภทนี้เนื่องจากขาดประสบการณ์ หรือการวิเคราะห์ที่ดี งานวิจัยนี้จึงจัดทำขึ้นโดยมีวัตถุประสงค์เพื่อสร้างตัวแทน (Agent) ที่ใช้กระบวนการเรียนรู้แบบเสริมแรงเชิงลึก (Deep Reinforcement Learning) ที่มีอัลกอริทึมในการเรียนรู้ซื้อขายคู่สกุลเงินในตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Forex) ที่แตกต่างกัน เพื่อให้ตัวแทนสามารถหาจุดเข้าซื้อขายและจุดออกได้อย่างเหมาะสม โดยจะเปรียบเทียบประสิทธิภาพการซื้อขายของตัวแทนแต่ละตัวผ่านผลตอบแทนสะสมที่ตัวแทนสามารถทำได้ ทั้งนี้ผู้วิจัยได้ทำการเก็บข้อมูลราคาคู่สกุลเงิน EUR/USD ในระดับวัน พร้อมทั้งการคำนวณหาค่าชี้วัดเชิงเทคนิค (Technical Indicators) เพื่อเป็นข้อมูลให้กับตัวแทนได้เรียนรู้และตัดสินใจเลือกการกระทำที่เหมาะสมที่สุดในแต่ละช่วงเวลา ผลการทดลองพบว่าตัวแทนโดยส่วนใหญ่สามารถทำกำไรจากการซื้อขายคู่สกุลเงินได้ นอกจากนี้พบว่าตัวแทนที่ใช้อัลกอริทึม Dueling Double Deep Q-Learning และใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญที่มีการให้น้ำหนักความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ระดับ 70% สามารถทำกำไรสะสมในขั้นตอนการทดสอบได้สูงที่สุด

คำสำคัญ : การเรียนรู้แบบเสริมแรงเชิงลึก, ตลาดแลกเปลี่ยนเงินตราต่างประเทศ, ตัวแทนซื้อขายแบบอัตโนมัติ

Independent Study Title	Deep Reinforcement Learning for Forex Trading
Student Name	Mr.Tatchai Samutsopakul
Student ID	63605096
Degree	Master of Science (Data Science and Analytics)
	KMITL-Digital Analytics and Intelligence Center
Year	2022
Independent Study Advisor	Assoc. Prof. Dr. Laor Boongasame

Abstract

In the present day, the new generations are more interested in investing in financial assets, whether stocks, currency pairs, or even cryptocurrencies. But in fact, very few people successfully invest in these assets due to their lack of experience or proper analysis. Therefore, this study aims to create agents that learn to trade currency pairs to get the highest returns in the foreign exchange market through Deep Reinforcement Learning with various algorithms. Accordingly, the agent can find the appropriate trading and exit points. The daily EUR/USD prices are collected and calculated as technical indicators to serve as training and testing data for agents. As a result, most agents can profit from currency pair trading. Also, the agent that uses the Dueling Double Deep Q-Learning algorithm, the Prioritized Experience Replay buffer, and weights the expected reward at 70% can get the best returns in the testing phase.

Keyword : Deep Reinforcement Learning, Foreign Exchange Market, Automated Trading Agent

กิตติกรรมประกาศ

โครงงานพิเศษฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับคำแนะนำด้านวิชาความรู้ คำปรึกษา มาจากคณาจารย์หลายท่าน ซึ่งข้าพเจ้าขอขอบพระคุณทุกท่านไว้ ณ ที่นี้

ข้าพเจ้าขอขอบพระคุณ รศ.ดร. ละออ บุญเกษม ที่กรุณารับเป็นอาจารย์ที่ปรึกษาโครงงานพิเศษที่ได้ให้ทั้งคำปรึกษา คำแนะนำด้านวิชาความรู้ กำลังใจ ตลอดจนการปรับปรุงแก้ไขข้อบกพร่องและให้ความช่วยเหลือในด้านต่าง ๆ เมื่อข้าพเจ้าพบปัญหาในระหว่างการจัดทำโครงงานพิเศษฉบับนี้ ซึ่งช่วยให้การจัดทำโครงงานพิเศษฉบับนี้มีความสมบูรณ์มากยิ่งขึ้น

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณบิดามารดาและครอบครัวของข้าพเจ้าที่ได้ให้ชีวิตและโอกาสทางการศึกษา คอยเป็นกำลังใจและให้ความห่วงใยเสมอมา ตลอดจนเพื่อน อาจารย์และผู้ที่อยู่เบื้องหลังทุกท่านที่กรุณาประสิทธิ์ประสาทวิชาความรู้อันเป็นประโยชน์แก่ข้าพเจ้า ให้ข้าพเจ้าสามารถทำโครงงานพิเศษฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

นายธัชชัย สมุทรโสภากุล

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	ก
บทคัดย่อภาษาอังกฤษ	ข
กิตติกรรมประกาศ	ค
สารบัญ	ง
สารบัญตาราง	ซ
สารบัญแผนภาพ	ฌ
สารบัญสมการ	ฒ
สารบัญอัลกอริทึม	ณ
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญชองปัญหา	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ขอบเขตของงานวิจัย	2
1.4 ประโยชน์ที่คาดว่าจะได้รับ	2
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง	3
2.1 ตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Foreign Exchange Market: Forex)	3
2.1.1 คู่สกุลเงิน (Currency Pairs)	3
2.1.2 ราคาคู่สกุลเงิน (Currency Pair Quotes)	3
2.1.3 PIP และ Spread	4
2.1.4 ตำแหน่ง (Positions) และการกระทำในการซื้อขาย (Actions)	4
2.2 ข้อมูลการเงิน (Financial Data)	4
2.3 การวิเคราะห์เชิงเทคนิค (Technical Analysis)	5
2.3.1 Trend Following	5
2.3.2 Momentum	7
2.3.3 Volatility	9
2.3.4 Volume	9
2.4 การปรับขนาดคุณสมบัติ (Feature Scaling)	10
2.5 ระบบโครงข่ายประสาทเทียม (Artificial Neural Network)	13
2.5.1 กระบวนการเรียนรู้ (Learning Process)	15
2.5.2 การทำให้เป็นมาตรฐาน (Regularization)	15

หน้า

2.5.3 ฟังก์ชันกระตุ้น (Activation Function) 16
2.5.4 การเรียนรู้แบบกลุ่ม (Batch Learning) 17
2.6 การเรียนรู้แบบเสริมแรง (Reinforcement Learning) 17
2.6.1 Markov Decision Process (MDP) 18
2.6.2 Discounted Cumulative Expected Reward 19
2.6.3 Episodic และ Continuous Tasks 20
2.6.4 Policy Function และ Value Function 20
2.6.5 วิธีการ Monte Carlo 24
2.6.6 วิธีการ Monte Carlo Prediction 25
2.6.7 วิธีการ Temporal Difference 29
2.6.8 วิธีการ Q-Learning 32
2.6.9 นโยบายละโมบด้วยค่า Epsilon (Epsilon Greedy) 35
2.7 การเรียนรู้แบบเสริมแรงเชิงลึก (Deep Reinforcement Learning) 36
2.7.1 วิธีการเรียนรู้แบบเสริมแรงเชิงลึก 37
2.7.1.1 วิธีการแบบ Value 37
2.7.1.2 วิธีการแบบ Policy 37
2.7.1.3 วิธีการแบบ Actor-Critic 37
2.7.2 Deep Q-Learning (DQN) 38
2.7.3 Double Deep Q-Learning (DDQN) 40
2.7.4 Dueling Double Deep Q-Learning (Dueling DDQN) 41
2.7.5 Dueling Double Deep Q-Learning with Prioritized Experience Replay (Dueling DDQN + PER) 42
2.7.6 ถังเก็บข้อมูลประสบการณ์ 42
2.7.6.1 ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) 42
2.7.6.2 ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay) 43
2.8 งานวิจัยที่เกี่ยวข้อง 45
บทที่ 3 วิธีการดำเนินงานวิจัย 48
3.1 ขั้นตอนในการวิจัย 48
3.2 ชุดข้อมูลและวิธีเก็บรวบรวมข้อมูลที่ใช้ในการทดลอง 49
3.3 ภาพรวมของระบบ 49

หน้า

3.3.1 สภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Forex Environment) 50
3.3.2 Agent ที่ใช้ในการแก้ปัญหา 51
3.4 การออกแบบการทดลอง 53
3.5 รายละเอียดการพัฒนาสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ 55
3.5.1 กำหนดการกระทำที่เป็นไปได้ทั้งหมดในสภาพแวดล้อม 55
3.5.2 กำหนดตำแหน่งที่เป็นไปได้ทั้งหมดในสภาพแวดล้อม 55
3.5.3 ปรับปรุงข้อมูลสถานะ (State) 56
3.5.4 ปรับปรุงกลไกการดำเนินการซื้อขายเงินตราต่างประเทศ 57
3.5.5 ปรับปรุงฟังก์ชันการให้รางวัล (Reward Function) 59
3.5.6 ปรับปรุงวิธีการคำนวณกำไรขาดทุน 65
3.5.7 ปรับปรุงช่วงของรางวัลเพื่อเพิ่มประสิทธิภาพการเรียนรู้ของ Agent ให้มีความเสถียรมากยิ่งขึ้น 66
3.5.8 สร้างสภาพแวดล้อมสำหรับการฝึกสอนและสภาพแวดล้อมสำหรับการทดสอบ 67
3.6 รายละเอียดในการพัฒนา Agent ที่ใช้สำหรับการแก้ปัญหา 67
3.6.1 การพัฒนานโยบายสำหรับการเลือกการกระทำ 67
3.6.2 การพัฒนาถังเก็บข้อมูลประสบการณ์ 68
3.6.3 การพัฒนา Agent ประเภท Double Deep Q-Learning (DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) 69
3.6.4 การพัฒนา Agent ประเภท Dueling Double Deep Q-Learning (Dueling DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) 73
3.6.5 การพัฒนา Agent ประเภท Double Deep Q-Learning (DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay) 75
3.6.6 การพัฒนา Agent ประเภท Dueling Double Deep Q-Learning (Dueling DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay) 78
บทที่ 4 ผลการวิจัยและการอภิปรายผล 79
4.1 การทดลองที่ 1 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับปริมาณข้อมูลในการสุ่มมาเรียนรู้ (Batch Size) 79
4.1.1 วัตถุประสงค์ 79

หน้า

4.1.2 การทดลองที่ 1.1 สุ่มข้อมูลทีละ 64 samples ในการเรียนรู้ 79
4.1.3 การทดลองที่ 1.2 สุ่มข้อมูลทีละ 128 samples ในการเรียนรู้ 81
4.1.4 การทดลองที่ 1.3 สุ่มข้อมูลทีละ 256 samples ในการเรียนรู้ 83
4.1.5 สรุปผลการทดลอง 85
4.2 การทดลองที่ 2 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับอัลกอริทึมในการเรียนรู้ (Optimizer) และอัตราการเรียนรู้ (Learning Rate) 86
4.2.1 วัตถุประสงค์ 87
4.2.2 การทดลองที่ 2.1 ใช้อัลกอริทึม RMSprop ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.001 87
4.2.3 การทดลองที่ 2.2 ใช้อัลกอริทึม RMSprop ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.0005 89
4.2.4 การทดลองที่ 2.3 ใช้อัลกอริทึม AdamW ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.001รู้ 91
4.2.5 การทดลองที่ 2.4 ใช้อัลกอริทึม AdamW ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.0005 94
4.2.6 สรุปผลการทดลอง 96
4.3 การทดลองที่ 3 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับการให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคต (Gamma) 97
4.3.1 วัตถุประสงค์ 97
4.3.2 การทดลองที่ 3.1 กำหนด Gamma เท่ากับ 0.1 98
4.3.3 การทดลองที่ 3.2 กำหนด Gamma เท่ากับ 0.3 99
4.3.4 การทดลองที่ 3.3 กำหนด Gamma เท่ากับ 0.5 102
4.3.5 การทดลองที่ 3.4 กำหนด Gamma เท่ากับ 0.7 103
4.3.6 การทดลองที่ 3.5 กำหนด Gamma เท่ากับ 0.9 106
4.3.7 สรุปผลการทดลอง 107
บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ 110
5.1 สรุปผลการดำเนินงาน 110
5.2 ขอบเขตและข้อจำกัด 111
5.3 ปัญหาและอุปสรรค 111
5.4 ข้อเสนอแนะ 112
เอกสารอ้างอิง 113
ประวัติผู้เขียน 115

สารบัญตาราง

ตารางที่		หน้า
3.1	ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD	49
3.2	ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ที่ถูกเพิ่มค่าชี้วัดเชิงเทคนิค	49
3.3	การพัฒนา Agent สำหรับการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ	52
3.4	Library ที่ใช้ในการพัฒนา	52
3.5	ตัวชี้วัดสำหรับการเปรียบเทียบประสิทธิภาพระหว่าง Agent	53
3.6	ค่าพารามิเตอร์สำหรับการทดลอง	53
3.7	การกระทำที่เป็นไปได้ทั้งหมด	55
3.8	ตำแหน่งที่เป็นไปได้ทั้งหมด	55
3.9	ข้อมูล 3 วันย้อนหลังในกรณีที่วันปัจจุบันเป็นวันที่ 2015-01-03	56
3.10	ข้อมูล 3 วันย้อนหลังนับตั้งแต่วันปัจจุบันที่ถูกปรับช่วงข้อมูล	56

สารบัญแผนภาพ

แผนภาพที่		หน้า
2.1	เส้นราคา SMA ตามจำนวนวันที่ใช้คำนวณ	6
2.2	เส้นราคา EMA ตามจำนวนวันที่ใช้คำนวณ	6
2.3	จุดไข่ปลาที่แสดงถึงแนวโน้มของราคา	7
2.4	เส้น RSI ที่บ่งบอกถึงสภาวะของตลาดว่าปกติหรืออยู่ในภาวะขายมากเกินไปหรือซื้อมากเกินไป	7
2.5	เส้น EMA12 และ EMA26 และเส้น EMA9 ที่เป็นเส้น Signal	8
2.6	เส้น ADX ทีบ่งบอกถึงความแข็งแกร่งของแนวโน้ม	8
2.7	เส้น Bollinger Bands ที่แสดงถึงความผันผวนของตลาด	9
2.8	เส้น OBV ที่แสดงถึงปริมาณการซื้อขายสะสม	10
2.9	แกนประสาทนำออก (Axon)	13
2.10	กระบวนการของระบบโครงข่ายประสาทเทียม	14
2.11	โครงข่ายประสาทเทียมประเภท Fully-connected ที่มี Input 1 ชั้น Output 1 ชั้นและ Hidden 1 ชั้น	14
2.12	ฟังก์ชัน Sigmoid	16
2.13	ฟังก์ชัน tanh	16
2.14	ฟังก์ชัน ReLU	17
2.15	แนวคิดเบื้องหลังของ Reinforcement Learning	18
2.16	สภาพแวดล้อมของเกม Grid World	20
2.17	นโยบายที่เหมาะสมที่สุดในแต่ละสถานะของเกม Grid World	20
2.18	กระบวนการประเมินและปรับปรุงนโยบายของวิธี Monte Carlo Control	25
2.19	ตารางค่า V ของแต่ละสถานะของเกม Grid World	26
2.20	ลำดับการปฏิสัมพันธ์ของตัวแทนทั้ง 2 Episode	26
2.21	ตารางค่า V ที่ถูกอัพเดทหลังปฏิสัมพันธ์ 2 Episode	28
2.22	ตารางค่า V ที่เหมาะสมที่สุดด้วยวิธีการ Monte Carlo Prediction	28
2.23	ตารางค่า V เริ่มต้นของวิธีการ Temporal Difference	30
2.24	วิธีการคำนวณหาค่ารางวัลที่คาดหวังของสถานะของ timestep แรก	30
2.25	ตารางค่า V ที่เหมาะสมที่สุดด้วยวิธีการ Temporal Difference	32
2.26	ตารางค่า Q เริ่มต้นของวิธีการ Q-Learning	33
2.27	วิธีการคำนวณหาค่ารางวัลที่คาดหวังของสถานะและการกระทำของ timestep แรก	33

แผนภาพที่ **หน้า**

2.28 ตารางค่า Q ที่เหมาะสมที่สุดด้วยวิธีการ Q-Learning 35
2.29 อัตราการลดของค่า Epsilon ของวิธีการ Linear และ Exponential 36
2.30 สถาปัตยกรรม Actor-Critic โดย Actor ทำหน้าที่เป็นนโยบายและแปลงค่าจากสถานะที่รับเข้ามาไปเป็นการกระทำ ในส่วนของ Critic ทำหน้าที่เป็นฟังก์ชันค่า โดยทั้งสองเครือข่ายสามารถอัพเดทผ่านค่า TD-error ได้เหมือนกัน ดังนั้น Actor จะใช้ Critic ในขั้นตอนการเรียนรู้ 38
2.31 โครงสร้างโครงข่ายประสาทเทียมแบบ Deep Q-Network 39
2.32 โครงสร้างโครงข่ายประสาทเทียมแบบ Dueling Deep Q-Network 41
3.1 ภาพรวมการตอบโต้ระหว่าง Agent และสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ 50
3.2 แสดงถึงตัวอย่างข้อมูลสถานะที่ Agent จะได้รับใน 1 Timestep ในกรณีที่ขนาดกรอบข้อมูลย้อนหลังเป็น 3 วัน และมีค่าชี้วัดเชิงเทคนิค 9 ตัว 57
3.3 ลำดับการซื้อขายและการปรับปรุงตำแหน่งการซื้อขาย 58
3.4 เงื่อนไขการซื้อขายและตำแหน่งการซื้อขาย 59
3.5 เกม CoastRunner ที่ Agent พยายามทำลายเรือเพื่อเก็บคะแนนในเกมให้ได้มากที่สุด 60
3.6 ตัวอย่างฟังก์ชันการให้รางวัลที่ส่งผลให้พฤติกรรมของ Agent ไม่ตรงตามความต้องการ 61
3.7 ตัวอย่างฟังก์ชันการให้รางวัลที่ส่งผลให้พฤติกรรมของ Agent ตรงตามความต้องการ 62
3.8 ลำดับการให้รางวัลการกระทำของฟังก์ชันการให้รางวัล 63
3.9 เงื่อนไขการให้รางวัลการกระทำ 64
3.10 ลำดับการคำนวณกำไรขาดทุน 65
3.11 เงื่อนไขการคำนวณกำไรขาดทุน 66
3.12 ฟังก์ชันสำหรับสร้างสภาพแวดล้อมที่มีการปรับปรุงช่วงของรางวัล 66
3.13 ชุดข้อมูลสำหรับสภาพแวดล้อมฝึกสอนและชุดข้อมูลสำหรับสภาพแวดล้อมทดสอบ 67
3.14 วิธีการเลือกการกระทำแบบละโมบ 67
3.15 วิธีการเลือกการกระทำแบบละโมบด้วยค่า Epsilon 68
3.16 ความแตกต่างระหว่าง Experience Replay และ Prioritized Experience Replay 69
3.17 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Double Deep Q-Learning 70
3.18 โครงสร้างโครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent ประเภท Double Deep Q-Learning 70
3.19 โครงข่ายประสาทเทียมสำหรับการเรียนรู้ประเภท Deep Q-Network 71
3.20 แสดงถึงวิธีการปรับปรุงค่า Epsilon ให้ลดลงตามจำนวนรอบการเรียนรู้ 71
3.21 ฟังก์ชันสำหรับการตอบโต้กับสภาพแวดล้อมในการเก็บข้อมูลประสบการณ์ 72

แผนภาพที่ **หน้า**

3.22 วิธีการเรียนรู้โครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent 72
3.23 แสดงถึงวิธีการคัดลอกน้ำหนักพารามิเตอร์จากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ไปใส่ในโครงข่ายประสาทเทียมสำหรับการประเมิน 73
3.24 แสดงถึงวิธีการทดสอบของ Agent กับสภาพแวดล้อมสำหรับการทดสอบ 73
3.25 โครงสร้างโครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent ประเภท Dueling Double Deep Q-Learning 74
3.26 โครงข่ายประสาทเทียมสำหรับการเรียนรู้ประเภท Dueling Deep Q-Network 74
3.27 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Dueling Double Deep Q-Learning 75
3.28 การสร้าง Dueling Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนโครงข่ายประสาทเทียมเป็นประเภท Dueling 75
3.29 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Double Deep Q-Learning ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ 76
3.30 การสร้าง Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนถังเก็บข้อมูลเป็นประเภทถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ 76
3.31 วิธีการอัพเดทค่าความสำคัญของประสบการณ์และการคำนวณค่าสูญเสียสำหรับการเรียนรู้ 77
3.32 วิธีการปรับปรุงค่า Alpha และ Beta ตามรอบการเรียนรู้ 77
3.33 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Dueling Double Deep Q-Learning ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ 78
3.34 การสร้าง Dueling Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนถังเก็บข้อมูลเป็นประเภทถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ 78
4.1 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples 80
4.2 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples 81
4.3 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples 81
4.4 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples 82
4.5 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples 83

แผนภาพที่ **หน้า**

4.6 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples 83
4.7 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples 84
4.8 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples 85
4.9 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples 85
4.10 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึมและขนาดของข้อมูลที่สุ่มมาเรียนรู้ที่ส่งผลต่อรางวัลสะสม 86
4.11 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001 88
4.12 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001 88
4.13 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001 89
4.14 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005 90
4.15 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005 91
4.16 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005 91
4.17 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001 92
4.18 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001 93
4.19 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001 93
4.20 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005 94
4.21 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005 95

แผนภาพที่ **หน้า**

4.22 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005 95
4.23 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึม อัลกอริทึมในการเรียนรู้และอัตราการเรียนรู้ที่ส่งผลต่อรางวัลสะสม 97
4.24 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10% 98
4.25 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10% 99
4.26 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10% 99
4.27 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30% 100
4.28 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30% 101
4.29 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30% 101
4.30 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50% 102
4.31 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50% 103
4.32 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50% 103
4.33 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70% 104
4.34 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70% 105
4.35 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70% 105
4.36 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90% 106
4.37 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90% 107

แผนภาพที่ **หน้า**

4.38 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90% 107

4.39 กราฟแสดงการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศของ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ถังเก็บข้อมูลที่มีการจัดลำดับความสำคัญและมีการกำหนดค่า Gamma ที่ 0.7 กับข้อมูลทดสอบ 108

4.40 กราฟแสดงการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศของ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ถังเก็บข้อมูลที่มีการจัดลำดับความสำคัญและมีการกำหนดค่า Gamma ที่ 0.7 กับข้อมูลฝึกสอน 109

4.41 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึมและความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ส่งผลต่อรางวัลสะสม 109

สารบัญสมการ

สมการที่		หน้า
2.1	Min-Max Normalization	10
2.2	Min-Max Normalization with Customs Range	11
2.3	Standardization	11
2.4	Mean	12
2.5	Welford’s Mean	12
2.6	Variance	12
2.7	Welford’s Variance	12
2.8	Discounted Cumulative Expected Reward	19
2.9	Value Function	21
2.10	Value Function (Recursive Form)	21
2.11	State-Action Value Function	22
2.12	Optimal Value Function	23
2.13	Optimal State-Action Value Function	23
2.14	Optimal Value Function (Recursive Form)	24
2.15	Monte Carlo Value Function	26
2.16	Temporal Difference Value Function	29
2.17	Q-Learning State-Action Value Function	32
2.18	Epsilon Greedy Policy	35
2.19	Policy-based Objective Function	37
2.20	Deep Q-Learning TD-Target	39
2.21	Deep Q-Learning Loss Function	40
2.22	Double Deep Q-Learning TD-Target	40
2.23	Dueling Deep Q-Learning State-Action Value Function	41
2.24	Prioritized Experience Replay’s Probability Function	43
2.25	Prioritized Experience Replay’s Priority Function	43
2.26	Prioritized Experience Replay’s Weighted Function	44
2.27	Deep Q-Learning with Prioritized Experience Replay Loss Function	44

สารบัญอัลกอริทึม

อัลกอริทึมที่		หน้า
3.1	Greedy Action Selection	67
3.2	Epsilon-Greedy Action Selection	68

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Foreign Exchange Market: Forex) คือตลาดที่ทำการซื้อขายอัตราแลกเปลี่ยนเงินตรา โดยราคานั้นจะแปรผันตามความต้องการซื้อความต้องการขายของแต่ละสกุลเงินและปัจจัยอื่น ๆ เช่น ดอกเบี้ย อัตราเงินเฟ้อ ราคาน้ำมัน ราคาทองคำ สภาพเศรษฐกิจเป็นต้น [1] โดยตลาดแลกเปลี่ยนเงินตราต่างประเทศมีปริมาณการซื้อขายแลกเปลี่ยนเงินตราระหว่างสกุลเงินสูงถึง 6.6 ล้านล้านดอลลาร์ต่อวันในปีค.ศ. 2019 ซึ่งเป็นตลาดที่มีปริมาณการซื้อขายสูงที่สุดเมื่อเทียบกับตลาดอื่น ๆ เช่น ตลาดหุ้นลอนดอนมีปริมาณการซื้อขายเฉลี่ยต่อวันอยู่ที่ 22.4 พันล้านดอลลาร์ต่อวัน หรือตลาดหุ้นญี่ปุ่นมีปริมาณการซื้อขายเฉลี่ยต่อวันอยู่ที่ 18.9 พันล้านดอลลาร์ต่อวันเป็นต้น [2] ทำให้ในปัจจุบันมีนักลงทุนจำนวนมากหันมาลงทุนเก็งกำไรในตลาดแลกเปลี่ยนเงินตราต่างประเทศมากขึ้น ซึ่งสินทรัพย์ที่ทำการซื้อขายในตลาดแลกเปลี่ยนเงินตราต่างประเทศนั้นจะทำการซื้อขายผ่านสิ่งที่เรียกว่าคู่สกุลเงิน

คู่สกุลเงิน (Currency Pair) คือราคาของสองสกุลเงินที่แตกต่างกัน โดยมีมูลค่าของสกุลเงินหนึ่งเทียบกับอีกสกุลเงินหนึ่ง สกุลเงินรายการแรกจะเรียกว่าสกุลเงินหลัก (Base Currency) และสกุลเงินที่สองจะเรียกว่าสกุลเงินรอง (Quote Currency) ซึ่งก็คือราคาที่ต้องใช้ในสกุลเงินรองในการซื้อ 1 หน่วยของสกุลเงินหลัก [3] ด้วยเหตุนี้ราคาของคู่สกุลเงินจะผันผวนขึ้นอยู่กับปัจจัยต่าง ๆ ของทั้ง 2 สกุลเงินที่นำมาเปรียบเทียบกัน ทำให้นักลงทุนหันมาสนใจลงทุนในตลาดแลกเปลี่ยนเงินตราต่างประเทศมากขึ้น

นักลงทุนในตลาดแลกเปลี่ยนเงินตราต่างประเทศมีอยู่ 6 กลุ่มใหญ่คือ 1. ธนาคารกลาง (Central Bank) 2. ธนาคารพาณิชย์ (Commercial Bank) 3. บริษัทข้ามชาติ (Multinational Corporations) 4. ผู้จัดการการลงทุนและกองทุน (Investment Managers and Hedge Funds) 5. โบรกเกอร์ (Brokers) และ 6. นักลงทุนรายย่อย (Individual Investors) [2] ทำให้ปริมาณการซื้อขายในตลาดมีความผันผวนสูง นักลงทุนหน้าใหม่ส่วนใหญ่จึงไม่ประสบความสำเร็จในการลงทุนในตลาดแลกเปลี่ยนเงินตราต่างประเทศ เนื่องจากนักลงทุนหน้าใหม่ยังขาดความรู้ความเข้าใจถึงพฤติกรรมของราคา และกลไกการควบคุมราคาของคู่สกุลเงิน

ในปัจจุบันนักลงทุนมีการนำกลยุทธ์ Algorithmic Trading มาปรับใช้ในการซื้อขายอัตราแลกเปลี่ยนเงินตรา [4] มากยิ่งขึ้น จากความก้าวหน้าของอุปกรณ์คอมพิวเตอร์และอัลกอริทึมในการแก้ปัญหาที่มีความซับซ้อนสูงเช่น การนำกระบวนการเรียนรู้ของเครื่องเพื่อหาจุดซื้อขายคู่สกุลเงินที่เหมาะสม การพยากรณ์ราคาคู่สกุลเงินไปข้างหน้าเพื่อดูทิศทางและแนวโน้มของราคา หรือการจัดกลุ่มคู่สกุลเงินที่มักมีพฤติกรรมของราคาที่คล้ายคลึงกัน ด้วยเหตุนี้เองเพื่อให้นักลงทุนสามารถ

ตัดสินใจในการลงทุนได้ดียิ่งขึ้น จึงมักจะนำเอาวิธีการดังกล่าวมาเป็นอีกหนึ่งตัวช่วยในการตัดสินใจในการหาจุดเข้าซื้อขายที่เหมาะสม

สำหรับการค้นคว้านี้ จะทำการดึงข้อมูลราคาคู่สกุลเงินจากเว็บไซต์ที่ให้บริการทางอินเตอร์เน็ต และทำการจัดเตรียมข้อมูลให้อยู่ในรูปแบบที่เหมาะสมที่สุดไม่ว่าจะเป็นการคำนวณตัวชี้วัดเชิงเทคนิคหรือการปรับช่วงของข้อมูลเพื่อนำไปสร้างสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ รวมทั้งการออกแบบสถานะของสภาพแวดล้อม กล่าวคือข้อมูลที่ตัวแทนจะใช้ในการตัดสินใจในแต่ละครั้งเพื่อเลือกการกระทำที่เหมาะสมที่สุดและการออกแบบฟังก์ชันการให้รางวัลซึ่งเป็นปัจจัยหนึ่งที่ส่งผลต่อพฤติกรรมของตัวแทน ดังนั้นการค้นคว้านี้จะทำการฝึกสอนตัวแทนในการหาจุดเปิดออร์เดอร์และจุดปิดออร์เดอร์ได้ทั้งขาขึ้นและขาลงของราคาคู่สกุลเงินโดยใช้อัลกอริทึมในรูปแบบที่แตกต่างกันกับสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ โดยมุ่งหวังว่าการค้นคว้านี้ตัวแทนจะสามารถวิเคราะห์หาจุดเปิดออร์เดอร์และจุดที่จะทำการปิดออร์เดอร์ทั้งขาขึ้นและขาลงของราคาคู่สกุลเงินได้อย่างเหมาะสมในแต่ละช่วงเวลาและมีประสิทธิภาพ

1.2 วัตถุประสงค์ของงานวิจัย

1) ศึกษาเฟรมเวิร์กทางคณิตศาสตร์ต่าง ๆ ที่สามารถใช้ในการสร้างแบบจำลองการตัดสินใจในสถานการณ์ต่าง ๆ

2) ศึกษาโมเดลต่าง ๆ ที่สามารถใช้ในการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ

3) ศึกษาคุณลักษณะของพฤติกรรมการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ

4) สร้างโมเดลเพื่อใช้ในการหาจุดซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ และช่วยให้นักลงทุนสามารถตัดสินใจจุดเข้าซื้อขายได้ดียิ่งขึ้น

1.3 ขอบเขตของงานวิจัย

1) สามารถแนะนำจุดเข้าซื้อขายเงินตราต่างประเทศได้อย่างเหมาะสม

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้เรียนรู้พฤติกรรมการซื้อขายอัตราแลกเปลี่ยนเงินตรา

2) ได้เรียนรู้และเข้าใจกลไกตลาดแลกเปลี่ยนเงินตราต่างประเทศ

3) ได้เรียนรู้และเข้าใจวิธีการเรียนรู้เชิงลึก (Deep Learning Method) และการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning Method) มากขึ้น

4) ฝึกทักษะการเขียนภาษาไพธอน (Python)

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Foreign Exchange Market: Forex)

ตลาด Forex คือตลาดแลกเปลี่ยนเงินตราต่างประเทศ ตลาด Forex ถือเป็นตลาดที่ใหญ่และเติบโตเร็วที่สุดในโลก ด้วยมูลค่าการซื้อขายเฉลี่ยต่อวันเกือบ 6.6 ล้านล้านดอลลาร์ต่อวันในปี 2019 [2] โดยตลาด Forex เปิดทำการ 24 ชั่วโมง 5 วันต่อสัปดาห์ เนื่องจากมีตลาดทั้งหมด 4 ตลาดหลัก คือ 1. London 2. New York 3. Sydney และ 4. Tokyo ทุก ๆ ตลาดจะเปิดทำการ 8 ชั่วโมงต่อวัน และมีเวลาเริ่มทำการแตกต่างกันออกไปขึ้นอยู่กับความต่างของเขตเวลา นั่นหมายความว่าแต่ละตลาดอาจจะมีเวลาทำการที่ทับซ้อนกัน ทำให้ช่วงเวลาดังกล่าวมีปริมาณการซื้อขายที่สูงกว่าปกติ

2.1.1 คู่สกุลเงิน (Currency Pairs)

เนื่องจากตลาด Forex คือตลาดแลกเปลี่ยนเงินตราต่างประเทศ กล่าวคือเป็นการแลกเปลี่ยนสกุลเงินหนึ่งกับอีกสกุลเงินหนึ่ง ดังนั้นทำให้การซื้อขายสกุลเงินต้องทำเป็นคู่จึงเรียกว่า คู่สกุลเงิน (Currency Pairs) โดยคู่สกุลเงินจะถูกเขียนต่อกันตามหลักของรหัสสกุลเงิน ISO (ISO 4217) โดยตัวขึ้นต้นจะเรียกว่าเป็นสกุลเงินหลักคือ (Base Currency) ตามด้วยสกุลเงินรองหรือสกุลเงินอ้างอิง (Quote Currency) และแบ่งสกุลเงินด้วย / เช่น คู่สกุลเงิน EUR/USD เป็นต้น โดยคู่สกุลเงินในตลาด Forex สามารถแบ่งได้เป็น 3 ประเภทคือ 1. Major 2. Crosses และ 3. Exotics โดยคู่สกุลเงินที่สำคัญที่สุดจะรวมอยู่ในประเภท Major โดยจะมีอยู่ 7 คู่หลักคือ 1. EUR/USD 2. GBP/USD 3. USD/JPY 4. USD/CHF 5. USD/CAD 6. AUD/USD และ 7. NZD/USD ในประเภท Crosses คือคู่สกุลเงินที่ไม่มีสกุลเงินดอลลาร์เป็นคู่ด้วย และประเภท Exotics คือคู่สกุลเงินที่มีประเทศที่กำลังพัฒนารวมอยู่ด้วย ดังนั้นจะมีสภาพคล่องที่ไม่สูงมากนัก

2.1.2 ราคาคู่สกุลเงิน (Currency Pair Quotes)

ราคาของคู่สกุลเงินจะแทนจำนวน 1 หน่วยของสกุลเงินหลักมีค่าเท่าไรเมื่อแปลงไปเป็นสกุลเงินรอง เช่นถ้าคู่สกุลเงิน EUR/USD มีราคาอยู่ที่ 1.15300 หมายความว่า 1 เหรียญยูโรมีมูลค่าเท่ากับ 1.15300 เหรียญดอลลาร์สหรัฐ ดังนั้นหากราคา EUR/USD ขึ้นไปที่ 1.15350 หมายความว่า EUR มีมูลค่ามากขึ้นเมื่อเทียบกับ USD เนื่องจากเราสามารถซื้อ USD ได้จำนวน 1.15350 เหรียญดอลลาร์เพียงใช้เงินจำนวน 1 เหรียญยูโร ในทางตรงกันข้ามหากราคา EUR/USD ลดลงไปที่ 1.15250 หมายความว่า EUR มีมูลค่าน้อยลงเมื่อเทียบกับ USD เนื่องจากเราสามารถซื้อ USD ได้จำนวน 1.15250 เหรียญดอลลาร์ในจำนวน 1 เหรียญยูโรเท่าเดิม

2.1.3 PIP และ Spread

PIP ย่อมาจาก Price Interest Point ใช้แทนจำนวนหลักของการเปลี่ยนแปลงราคาคู่สกุลเงินนั้น ๆ โดยนิยมใช้กันที่หลักที่ 2 หรือ 4 เช่นคู่สกุลเงิน EUR/USD มีราคาอยู่ที่ 1.12800 หมายความว่าราคาของคู่สกุลเงินนี้สามารถเปลี่ยนแปลงได้ถึงทศนิยมตำแหน่งที่ 5 นั้นหมายความว่า PIP ของคู่สกุลเงินนี้จะอยู่ที่ทศนิยมหลักที่ 4 เช่นราคาคู่สกุลเงิน EUR/USD ปัจจุบันอยู่ที่ 1.12800 หมายความว่าราคาสามารถลงไปที่ 1.12799 หรือขึ้นไปที่ 1.12801 ได้เป็นต้น แต่ก็จะมีบางสกุลเงินที่ PIP เปลี่ยนแปลงที่ทศนิยมหลักที่ 2 เช่นราคาคู่สกุลเงิน USD/JPY มีราคาปัจจุบันอยู่ที่ 114.000 หมายความว่าราคาสามารถลงไปที่ 113.999 หรือขึ้นไปที่ 114.001 ได้เช่นกัน ในส่วนของค่า Spread คือส่วนต่างของราคาขายและราคาซื้อ ดังนั้นเมื่อเราทำการซื้อคู่สกุลเงินเรามักจะติดลบเสมอเนื่องจากราคาซื้อจะแพงกว่าราคาขายเสมอ ซึ่งส่วนต่างในตรงนี้จะเป็นส่วนที่โบรกเกอร์ได้กำไรไป

2.1.4 ตำแหน่ง (Positions) และการกระทำในการซื้อขาย (Actions)

การซื้อขายอัตราแลกเปลี่ยนเงินตรานั้นสามารถทำการซื้อขายได้ 2 ลักษณะคือ 1. การเปิดคำสั่งเข้าซื้อ (Buy) หรือทำการเปิดออร์เดอร์ตำแหน่ง Long และ 2. การเปิดคำสั่งเข้าขาย (Sell) หรือทำการเปิดออร์เดอร์ตำแหน่ง Short

การเปิดคำสั่งเข้าซื้อ (Buy) หรือทำการเปิดออร์เดอร์ตำแหน่ง Long นั้นเป็นคำสั่งเปิดออร์เดอร์ที่เราคาดการณ์ว่าแนวโน้มของอัตราแลกเปลี่ยนเงินตราของคู่สกุลเงินนั้นจะสูงขึ้น [5] ดังนั้นคำสั่งซื้อประเภทนี้จะเป็นการทำกำไรที่การเข้าซื้อที่ราคาถูกแล้วขายที่ราคาแพง ตัวอย่างเช่น เราทำการเปิดออร์เดอร์ Buy (Long) ของอัตราแลกเปลี่ยนเงินตราคู่สกุลเงิน EUR/USD ที่ราคา 1.12800 และปิดออร์เดอร์ Buy (Long) ที่ราคา 1.13300 ดังนั้นเราสามารถทำกำไรจากออร์เดอร์นี้ได้จำนวน 1.13300-1.12800 = 0.00500 หรือคิดเป็นจำนวน 50 PIP

การเปิดคำสั่งเข้าขาย (Sell) หรือทำการเปิดออร์เดอร์ตำแหน่ง Short นั้นเป็นคำสั่งเปิดออร์เดอร์ที่เราคาดการณ์ว่าแนวโน้มของอัตราแลกเปลี่ยนเงินตราของคู่สกุลเงินนั้นจะลดลง [5] ดังนั้นคำสั่งซื้อประเภทนี้จะเป็นการทำกำไรที่การขายราคาแพงแล้วซื้อกลับตอนราคาถูก ตัวอย่างเช่น เราทำการเปิดออร์เดอร์ Sell (Short) ของอัตราแลกเปลี่ยนเงินตราคู่สกุลเงิน EUR/USD ที่ราคา 1.12800 และปิดออร์เดอร์ Sell (Short) ที่ราคา 1.12500 ดังนั้นเราสามารถทำกำไรจากออร์เดอร์นี้ได้จำนวน 1.12800-1.12500 = 0.00300 หรือคิดเป็นจำนวน 30 PIP

2.2 ข้อมูลการเงิน (Financial Data)

ข้อมูลการเงินเป็นข้อมูลที่มีแกนของเวลาเข้ามาเกี่ยวข้องไม่ว่าจะเป็นข้อมูลระดับชั่วโมง ระดับวันหรือแม้แต่ระดับเดือน แต่ข้อมูลการเงินมีลักษณะที่แตกต่างจากข้อมูลที่เป็นอนุกรมเวลาประเภทอื่น ๆ เนื่องจากมีปัจจัยต่าง ๆ มากมายที่มีอิทธิพลต่อราคาสุดท้ายก่อนปิดตลาด

โดยปกติข้อมูลอนุกรมเวลาจะประกอบไปด้วย 4 องค์ประกอบหลักคือ 1. เทรนด์ (Trend) 2. ฤดูกาล (Seasonal) 3. วัฏจักรการเคลื่อนไหว (Cyclic movements) และ 4. ความผันผวนที่ผิดปกติ (Irregular fluctuations) รูปแบบเทรนด์คือการมองข้อมูลว่ามีแนวโน้มที่จะเพิ่มขึ้นหรือลดลงในระยะยาวหรือไม่ ฤดูกาลแสดงถึงความผันผวนที่มักจะเกิดขึ้นในช่วงเวลาหนึ่งซ้ำ ๆ บ่อย ๆ ครั้ง รูปแบบของฤดูกาลอาจจะสังเกตเห็นได้ชัดเจนเมื่อข้อมูลมีการเปลี่ยนแปลงภายในระยะเวลาที่กำหนดเช่น ช่วง 3 เดือนสุดท้ายของสิ้นปี วัฏจักรการเคลื่อนไหวแสดงถึงลักษณะการเคลื่อนไหวที่มักจะเกิดขึ้นซ้ำ ๆ หรือรูปแบบของราคาซ้ำ ๆ เดิม และความผันผวนที่ผิดปกติจะเป็นองค์ประกอบแบบสุ่มที่เกี่ยวข้องกับเวลา หรือก็คือเป็นเหตุการณ์ที่ไม่สามารถคาดเดาได้มีการเกิดขึ้นแบบสุ่ม

แต่ข้อมูลการเงินมีคุณสมบัติที่ทำให้เกิดความผันผวนที่สูงตามช่วงเวลาที่เปลี่ยนแปลง ทั้งนี้ปัจจัยภายนอกก็มีผลต่อความผันผวนที่ผิดปกติของราคา โดยสามารถสรุปคุณสมบัติของข้อมูลการเงินได้ดังนี้ 1. การไม่เป็นเชิงเส้น (Non-linear) กล่าวคือข้อมูลการเงินไม่สามารถจำลองความสัมพันธ์ระหว่างตัวแปรอิสระให้อยู่ในรูปแบบของเส้นตรงได้ 2. มีลักษณะไม่นิ่ง (Non-stationary) กล่าวคือข้อมูลการเงินเปลี่ยนแปลงอย่างต่อเนื่องเนื่องจากราคาที่เป็นไปได้มีสูงและปริมาณการซื้อขายที่สูงส่งผลให้ราคามีการเปลี่ยนแปลงอย่างต่อเนื่อง และ 3. มีการรบกวน (Noisy) กล่าวคือข้อมูลการเงินมีรูปแบบข้อมูลหรือเหตุการณ์แบบสุ่มรวมอยู่ทำให้ยากต่อการทำนายราคาในอนาคต

2.3 การวิเคราะห์เชิงเทคนิค (Technical Analysis)

ในปัจจุบันนักลงทุนมีสองวิธีการที่ใช้ในการตัดสินใจในการลงทุนคือ การวิเคราะห์เชิงเทคนิค (Technical Analysis) และการวิเคราะห์ปัจจัยพื้นฐาน (Fundamental Analysis) โดยการวิเคราะห์ปัจจัยพื้นฐานเป็นการวิเคราะห์ในเรื่องของปัจจัยทางเศรษฐกิจและการเงินที่ส่งผลกระทบต่อตลาด เช่น สภาวะตลาดหรือข่าวสารทางเศรษฐกิจ แต่ในทางตรงกันข้ามการวิเคราะห์เชิงเทคนิคจะเป็นวิธีการที่ใช้ในการพยากรณ์ทิศทางของราคาในอนาคตผ่านข้อมูลที่ได้มาจากกิจกรรมการซื้อขายในตลาด ดังนั้นการวิเคราะห์เชิงเทคนิคจะสนใจในส่วนของการเคลื่อนไหวของราคา (Price Movement) และตัวชี้วัดเชิงเทคนิคอื่น ๆ (Technical Indicator) เข้ามาประกอบการตัดสินใจ สำหรับการศึกษานี้จะมุ่งเน้นที่การนำการวิเคราะห์เชิงเทคนิคมาปรับใช้กับตลาด Forex

ค่าชี้วัดเชิงเทคนิคเป็นค่าที่ถูกคำนวณมาจากราคาหรือปริมาณที่เกิดขึ้นในอดีตตามแต่สมการของแต่ละตัวชี้วัด เพื่อช่วยให้นักลงทุนสามารถประเมินการลงทุนและจับจังหวะการเทรดจากแนวโน้มของราคาและรูปแบบของราคาได้ โดยตัวชี้วัดเชิงเทคนิคถูกแบ่งออกได้เป็น 4 ประเภทหลักคือ 1. Trend 2. Momentum 3. Volatility และ 4. Volume

2.3.1 Trend Following

ตัวชี้วัดประเภทติดตามแนวโน้มจะเป็นตัวที่ช่วยระบุแนวโน้มของตลาด เพื่อช่วยในการหาจุดเข้าซื้อหรือขายในระยะเวลานั้น ๆ เช่น

1. Simple Moving Average (SMA) คือค่าเฉลี่ยของราคาในช่วงกรอบระยะเวลาหนึ่ง เช่น SMA9 หมายความว่าเป็นเส้นราคาที่ถูกหาค่าเฉลี่ยมาจาก 9 วันย้อนหลังดังแผนภาพที่ 2.1

แผนภาพที่ 2.1 เส้นราคา SMA ตามจำนวนวันที่ใช้คำนวณ

หมายเหตุ. จาก https://th.tradingview.com/

2. Exponential Moving Average (EMA) คือค่าเฉลี่ยของราคาในช่วงกรอบระยะเวลาหนึ่งเช่นเดียวกันกับ SMA แต่เพิ่มค่าถ่วงน้ำหนักความสำคัญกับราคาปัจจุบันมากกว่าราคาในอดีต เช่น EMA20 หมายความว่าเป็นเส้นราคาที่ถูกหาค่าเฉลี่ยมาจาก 20 วันย้อนหลังโดยมีการเพิ่มค่าถ่วงน้ำหนัก ดังแผนภาพที่ 2.2

แผนภาพที่ 2.2 เส้นราคา EMA ตามจำนวนวันที่ใช้คำนวณ

หมายเหตุ. จาก https://th.tradingview.com/

3. Parabolic SAR คือค่าที่ถูกคำนวณเพื่อหาแนวโน้มของราคาเช่น ในกรณีที่ค่าที่คำนวณมาหรือจุดไข่ปลาอยู่ใต้ราคาจะแสดงถึงแนวโน้มขาขึ้น แต่ถ้าหากค่าที่คำนวณมาหรือจุดไข่ปลาอยู่เหนือราคาจะแสดงถึงแนวโน้มขาลง ดังนั้นจะเป็นค่าที่บ่งบอกถึงแนวโน้มของราคาและสัญญาณการเปลี่ยนแนวโน้มของราคา ดังแผนภาพที่ 2.3

แผนภาพที่ 2.3 จุดไข่ปลาที่แสดงถึงแนวโน้มของราคา

หมายเหตุ. จาก https://th.tradingview.com/

2.3.2 Momentum

ตัวชี้วัดประเภทโมเมนตัมจะพยายามวัดความเร็วของทิศทางการเคลื่อนไหวของราคา เพื่อที่จะระบุความเร็วหรือความแข็งแกร่งของราคาที่เปลี่ยนแปลงจากผู้ซื้อและผู้ขายในตลาด โดยตัวชี้วัดที่นิยมที่ใช้ในปัจจุบันมีดังนี้

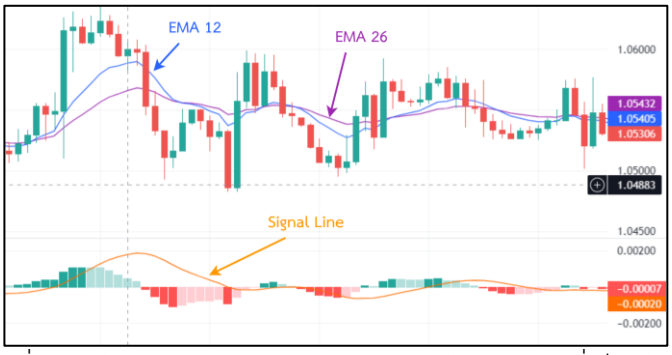
1. Relative Strength Index (RSI) เป็นตัวชี้วัดเพื่อบอกสภาวะของตลาดว่าคู่สกุลเงินนั้นถูกซื้อมากเกินไป (Overbought) หรือขายมากเกินไป (Oversold) โดยมีค่าตั้งแต่ 0 ถึง 100 โดยค่ามาตรฐานทั่วไปที่นิยมใช้กันคือ ค่า RSI อยู่ต่ำกว่า 30 จะถือว่าราคาอยู่ในภาวะขายมากเกินไป และหากมากกว่า 70 จะถือว่าราคาอยู่ในภาวะการซื้อมากเกินไป ดังแผนภาพที่ 2.4

แผนภาพที่ 2.4 เส้น RSI ที่บ่งบอกถึงสภาวะของตลาดว่าปกติหรืออยู่ในภาวะขายมากเกินไปหรือซื้อมากเกินไป

หมายเหตุ. จาก https://th.tradingview.com/

2. Moving Average Convergence Divergence (MACD) เป็นตัวชี้วัดที่สามารถให้ข้อมูลทั้งทิศทางแนวโน้มราคา (Trend) และแรงส่งของราคา (Momentum) เนื่องจากเป็นการวิเคราะห์ลักษณะที่เกิดขึ้นระหว่าง Moving Average ทั้ง 2 เส้น โดยการวิเคราะห์ที่มักใช้กันจะเป็น

เส้น EMA12 และ EMA26 และเส้นที่ทำหน้าที่เป็น Signal ก็คือ EMA9 มาช่วยระบุทิศทางเส้นแนวโน้มของ MACD ว่าเป็นขาขึ้นหรือขาลง ตัวอย่างการวิเคราะห์ในเชิงแนวโน้มเมื่อเส้น EMA12 อยู่เหนือ EMA26 หมายความว่ากราฟราคาเป็นแนวโน้มขาขึ้น ในทางตรงกันข้ามหาก EMA12 อยู่ใต้ EMA26 หมายความว่ากราฟราคาเป็นแนวโน้มขาลง และยังสามารถวิเคราะห์ในเชิงความเร็วของราคาได้เช่น ระยะห่างระหว่างเส้น EMA12 และ EMA26 ห่างกันมากขึ้นในขาขึ้นหมายความว่าราคาเพิ่มขึ้นด้วยอัตราเร็วที่เร็วขึ้นเป็นต้น ดังแผนภาพที่ 2.5



แผนภาพที่ 2.5 เส้น EMA12 และ EMA26 และเส้น EMA9 ที่เป็นเส้น Signal

หมายเหตุ. จาก https://th.tradingview.com/

3. Average Directional Index (ADX) เป็นตัวชี้วัดที่ช่วยวัดความแข็งแกร่งของสภาวะแนวโน้มของตลาด กล่าวคือเป็นตัวช่วยวิเคราะห์แนวโน้มว่ามีโอกาสจบลงหรือยังคงไปต่อเป็นต้น โดยมีค่าตั้งแต่ 0 ถึง 100 โดยค่ามาตรฐานจะแบ่งออกเป็น 4 ระดับคือ 1. เส้น ADX อยู่ระหว่าง 0-25 กล่าวคือแนวโน้มมีความอ่อนแอ 2. เส้น ADX มีค่าอยู่ระหว่าง 25-50 กล่าวคือแนวโน้มมีความแข็งแกร่ง 3. เส้น ADX มีค่าอยู่ระหว่าง 50-75 กล่าวคือแนวโน้มมีความแข็งแกร่งมากและ 4. เส้น ADX มีค่าอยู่ระหว่าง 75-100 กล่าวคือแนวโน้มมีความแข็งแกร่งเกินค่าเฉลี่ยทั่วไปอย่างมาก ซึ่งมักจะเกิดขึ้นได้น้อยและสภาวะแบบนี้จะอยู่ได้ไม่นานนัก ดังแผนภาพที่ 2.6

แผนภาพที่ 2.6 เส้น ADX ทีบ่งบอกถึงความแข็งแกร่งของแนวโน้ม

หมายเหตุ. จาก https://th.tradingview.com/

2.3.3 Volatility

ตัวชี้วัดความผันผวนเป็นตัววัดความไม่แน่นอนและความเสี่ยงของตลาด ซึ่งเป็นตัวชี้วัดที่สะท้อนให้เห็นถึงพฤติกรรมการเคลื่อนไหวของราคา หากค่าความผันผวนสูงหมายความว่าช่วงของราคาสามารถเปลี่ยนแปลงได้กว้างมากในระยะเวลาอันสั้น ในทางตรงกันข้ามหากค่าความผันผวนต่ำหมายความว่าราคามีความเสถียร อัตราการเปลี่ยนแปลงของราคาจะไม่รวดเร็ว โดยตัวชี้วัดที่นิยมที่ใช้ในปัจจุบันมีดังนี้

1. Bollinger Bands (BB) เป็นตัวชี้วัดที่สามารถวัดความผันผวนของตลาดได้ ซึ่งจะประกอบไปด้วยเส้น 3 เส้นโดยเส้นกลางคือ SMA20 กล่าวคือเส้นค่าเฉลี่ย 20 วัน เส้นบนคือ SMA20 รวมกับ 2 เท่าของค่าเบี่ยงเบนมาตรฐานของราคา 20 วันย้อนหลัง และเส้นล่างคือ SMA20 ลบกับ 2 เท่าของค่าเบี่ยงเบนมาตรฐานของราคา 20 วันย้อนหลัง โดยการวิเคราะห์ส่วนใหญ่หากทั้งสามเส้นมีลักษณะบีบชิดเข้าหากันหมายความว่าตลาดค่อนข้างเงียบกล่าวคือ มีนักลงทุนทำการซื้อขายน้อย แต่ถ้าหากทั้งสามเส้นแยกออกจากกันแบบห่าง ๆ หมายความว่าตลาดมีความคึกคักกล่าวคือมีนักลงทุนกำลังซื้อขายกันเป็นจำนวนมาก ดังแผนภาพที่ 2.7

แผนภาพที่ 2.7 เส้น Bollinger Bands ที่แสดงถึงความผันผวนของตลาด

หมายเหตุ. จาก https://th.tradingview.com/

2.3.4 Volume

ตัวชี้วัดปริมาณเป็นตัววัดปริมาณความต้องการซื้อและความต้องการขาย เพื่อช่วยในการคาดการณ์แนวโน้มของราคา โดยตัวชี้วัดที่นิยมที่ใช้ในปัจจุบันมีดังนี้

1. On Balance Volume (OBV) เป็นตัวชี้วัดที่วัดปริมาณการซื้อขายสะสม เพื่อวัดแรงซื้อแรงขายเพื่อใช้ในการยืนยันทิศทางของแนวโน้ม และหาจุดกลับตัวของราคา โดยการวิเคราะห์ส่วนใหญ่หาก OBV เคลื่อนไหวไปในทิศทางเดียวกับราคา หมายความว่าปริมาณและราคาสอดคล้องกันเป็นการยืนยันทิศทางของแนวโน้มนั้น แต่ถ้าหาก OBV เปลี่ยนทิศทางและราคาก็เปลี่ยนทิศทางเช่นเดียวกัน เป็นการยืนยันสัญญาณการเปลี่ยนแนวโน้ม ดังแผนภาพที่ 2.8

แผนภาพที่ 2.8 เส้น OBV ที่แสดงถึงปริมาณการซื้อขายสะสม

หมายเหตุ. จาก https://th.tradingview.com/

2.4 การปรับขนาดคุณสมบัติ (Feature Scaling)

การปรับขนาดคุณสมบัติของข้อมูลนั้น เป็นวิธีการสำหรับการปรับช่วงของข้อมูลเพื่อให้ข้อมูลมีช่วงที่เหมาะสมซึ่งเรียกกระบวนการนี้ว่าการปรับข้อมูลให้เป็นมาตรฐาน (Data Normalization) เนื่องจากบางครั้งคุณสมบัติของข้อมูลนั้น มีช่วงของค่าที่แตกต่างกันทำให้บางครั้งอัลกอริทึมที่ใช้ในการแก้ปัญหาให้ความสำคัญกับคุณสมบัติที่มีช่วงที่กว้างกว่า คุณสมบัติที่มีช่วงที่แคบกว่า ดังนั้นการปรับช่วงของข้อมูลให้เป็นมาตรฐานเดียวกันนั้น จะช่วยแก้ปัญหาในส่วนนี้ได้และข้อดีอีกอย่างหนึ่งคือการปรับช่วงข้อมูลให้เป็นมาตรฐานนั้นช่วยให้กระบวนการเรียนรู้หรือการทำ Gradient Descent ของอัลกอริทึมสามารถเรียนรู้ได้รวดเร็วมากกว่าการไม่ปรับช่วงข้อมูลให้เป็นมาตรฐาน วิธีการหลัก ๆ ในการปรับช่วงข้อมูลให้เป็นมาตรฐานมีดังนี้

1. วิธีการ Rescaling หรือ Min-Max Normalization

วิธีการ Min-Max Normalization เป็นวิธีการปรับช่วงข้อมูลอย่างง่าย โดยที่วิธีการนี้จะปรับช่วงข้อมูลให้อยู่ในช่วง [0, 1] โดยสามารถเขียนเป็นสมการได้ดังสมการที่ 2.1

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.1}$$

โดย x' คือ ค่า x เดิมที่ถูกปรับช่วงใหม่

x คือ ค่า x เดิมที่จะปรับ

$\min(x)$ คือ ค่าที่น้อยที่สุดในคุณสมบัตินั้น ๆ

$\max(x)$ คือ ค่าที่มากที่สุดในคุณสมบัตินั้น ๆ

แต่ในบางครั้งเราอาจจะต้องการช่วงข้อมูลที่แตกต่างกันออกไป ซึ่งเราสามารถใช้วิธีการ Min-Max Normalization แบบกำหนดช่วงของข้อมูลที่ต้องการได้สามารถเขียนได้ดังสมการที่ 2.2

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)} \tag{2.2}$$

โดย x' คือ ค่า x เดิมที่ถูกปรับช่วงใหม่

x คือ ค่า x เดิมที่จะปรับ

a คือ ขอบล่างของช่วงข้อมูลใหม่ที่ต้องการ

b คือ ขอบบนของช่วงข้อมูลใหม่ที่ต้องการ

$\min(x)$ คือ ค่าที่น้อยที่สุดในคุณสมบัตินั้น ๆ

$\max(x)$ คือ ค่าที่มากที่สุดในคุณสมบัตินั้น ๆ

2. วิธีการ Standardization หรือ Z-score Normalization

วิธีการ Standardization เป็นวิธีการปรับช่วงข้อมูลให้มีค่าเฉลี่ย (Mean) เป็น 0 และค่าส่วนเบี่ยงเบนมาตรฐานเป็น 1 (Standard Deviation) โดยสามารถเขียนเป็นสมการได้ดังสมการที่ 2.3

$$x' = \frac{x - \bar{x}}{\sigma} \tag{2.3}$$

โดย x' คือ ค่า x เดิมที่ถูกปรับช่วงใหม่

x คือ ค่า x เดิมที่จะปรับ

$\bar{x}$ คือ ค่าเฉลี่ยของคุณสมบัตินั้น ๆ

σ คือ ค่าส่วนเบี่ยงเบนมาตรฐานของคุณสมบัตินั้น ๆ

3. วิธีการ Welford

วิธีการปรับข้อมูลให้เป็นมาตรฐานที่กล่าวมา 2 วิธีการข้างต้นนั้นจะเป็นการคำนวณสำหรับชุดข้อมูลที่ครบครันหรือสำหรับการเรียนรู้แบบออฟไลน์นั้นเอง แต่ในบางครั้งอัลกอริทึมของเราอาจจะเป็นการเรียนรู้ที่ออนไลน์ตลอดเวลา กล่าวคือข้อมูลจะค่อย ๆ ไหลมาตามช่วงเวลาทำให้วิธีการทั้ง 2 วิธีการข้างต้นนั้นไม่สามารถทำงานกับการเรียนรู้แบบนี้ได้จึงเกิดวิธีการ Welford ขึ้นสำหรับการปรับข้อมูลให้เป็นมาตรฐานสำหรับอัลกอริทึมแบบออนไลน์ ซึ่งวิธีการนี้จะเป็นวิธีการที่คำนวณไปข้างหน้าเพียงครั้งเดียว (Single Pass) กล่าวคือต้องการเห็นข้อมูลแต่ละตัว x_i เพียงครั้งเดียวเท่านั้น โดยเราสามารถใช้วิธีการ Welford ในการปรับช่วงของข้อมูลให้มีค่าเฉลี่ยเป็น 0 และมีส่วนเบี่ยงเบนมาตรฐานเป็น 1 ได้ โดยปกติค่าเฉลี่ยสามารถหาได้จากสมการที่ 2.4

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.4}$$

โดย $\bar{x}_n$ คือ ค่าเฉลี่ยสำหรับข้อมูล n samples

x_i คือ ค่า x แต่ละตัว

n คือ จำนวน samples ทั้งหมด

จากสมการที่ 2.4 จะเป็นค่าเฉลี่ยสำหรับข้อมูล n samples แต่เราต้องการคำนวณหาค่าเฉลี่ยเพียงใช้ค่าเฉลี่ยเดิมและค่า sample ใหม่เท่านั้น ซึ่งสามารถพิสูจน์สมการที่ 2.4 และเขียนให้อยู่ในรูปใหม่ได้ดังสมการที่ 2.5 สำหรับการหาค่าเฉลี่ย

$$\begin{aligned}\bar{x}_n &= \frac{(n-1)\bar{x}_{n-1} + x_n}{n} \\ &= \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n}\end{aligned} \tag{2.5}$$

โดย $\bar{x}_n$ คือ ค่าเฉลี่ยใหม่สำหรับข้อมูล n samples

$\bar{x}_{n-1}$ คือ ค่าเฉลี่ยเดิมสำหรับข้อมูล $n-1$ samples

x_n คือ ข้อมูลใหม่

n คือ จำนวนข้อมูลทั้งหมด

ส่วนค่าความแปรปรวนนั้นโดยปกติสามารถหาได้จากสมการที่ 2.6

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2 \tag{2.6}$$

โดย σ_n^2 คือ ค่าความแปรปรวนสำหรับข้อมูล n samples

n คือ จำนวนข้อมูลทั้งหมด

$\bar{x}_n$ คือ ค่าเฉลี่ยสำหรับข้อมูล n samples

x_i คือ ข้อมูลแต่ละตัว

จากสมการที่ 2.6 เราต้องการหาค่าความแปรปรวนใหม่โดยใช้เฉพาะข้อมูลใหม่เท่านั้นไม่ต้องการใช้ข้อมูลทั้งหมดเฉกเช่นเดียวกันกับวิธีการหาค่าเฉลี่ย ซึ่งสามารถพิสูจน์สมการที่ 2.6 และเขียนให้อยู่ในรูปใหม่ได้ดังสมการที่ 2.7 สำหรับการหาค่าความแปรปรวน

$$\begin{aligned}\sigma_n^2 &= \frac{(n-1)\sigma_{n-1}^2 + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n)}{n} \\ &= \sigma_{n-1}^2 + \frac{(x_n - \bar{x}_{n-1})(x_n - \bar{x}_n) - \sigma_{n-1}^2}{n}\end{aligned} \tag{2.7}$$

โดย σ_n^2 คือ ค่าความแปรปรวนสำหรับข้อมูล n samples

σ_{n-1}^2 คือ ค่าความแปรปรวนสำหรับข้อมูล $n-1$ samples

n คือ จำนวนข้อมูลทั้งหมด

$\bar{x}_n$ คือ ค่าเฉลี่ยสำหรับข้อมูล n samples

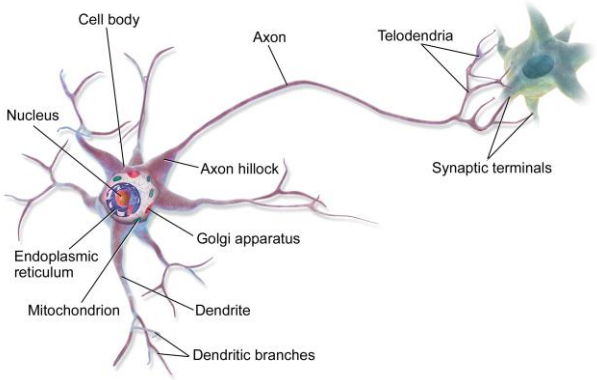
$\bar{x}_{n-1}$ คือ ค่าเฉลี่ยสำหรับข้อมูล $n-1$ samples

x_n คือ ข้อมูลใหม่

ดังนั้นเราจะสามารถหาค่าเฉลี่ยและค่าส่วนเบี่ยงเบนมาตรฐานจากค่าความแปรปรวนได้จากวิธีการเหล่านี้ ทำให้เรามีข้อมูลที่เพียงพอที่จะนำไปปรับช่วงของข้อมูลให้เป็นมาตรฐานตามวิธีการ Standardization ได้

2.5 ระบบโครงข่ายประสาทเทียม (Artificial Neural Network)

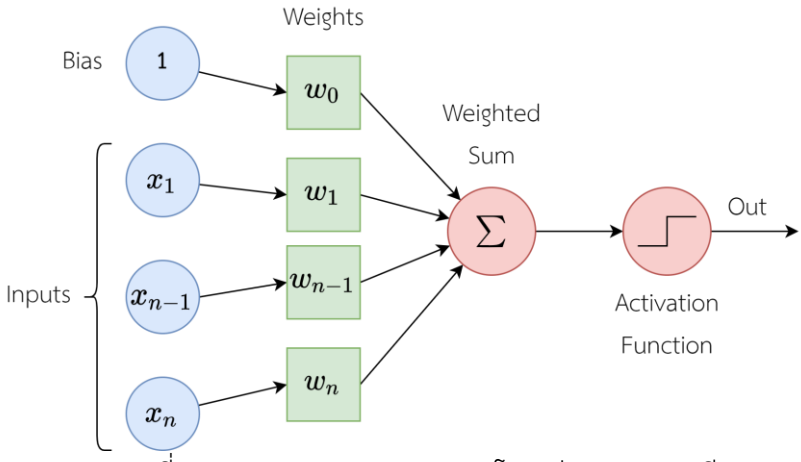
เซลล์ประสาทเทียม (Artificial neuron) ได้รับแรงบันดาลใจมาจากเซลล์สมองของมนุษย์ สมองของมนุษย์นั้นประกอบด้วยเซลล์ประสาทประมาณ 86,000 ล้านเซลล์เพื่อประมวลผลข้อมูลทางด้านประสาทสัมผัส ไม่ว่าจะเป็นการมองเห็น การสัมผัสและการได้ยิน เซลล์ประสาทหนึ่งเซลล์สามารถรับข้อมูลได้หลากหลาย ส่วนที่รับข้อมูลเรียกว่า Dendrite ซึ่งรับมาจากเซลล์ประสาทก่อนหน้า จากนั้นเซลล์ประสาทจะทำการประมวลผลจากข้อมูลที่ได้รับและถ้าข้อมูลนั้นถึงจุดศักยภาพก็จะส่งสัญญาณต่อไปให้เซลล์ประสาทอื่นที่เชื่อมต่อกันอยู่ผ่าน Axon ดังแผนภาพที่ 2.9



แผนภาพที่ 2.9 แกนประสาทนำออก (Axon)

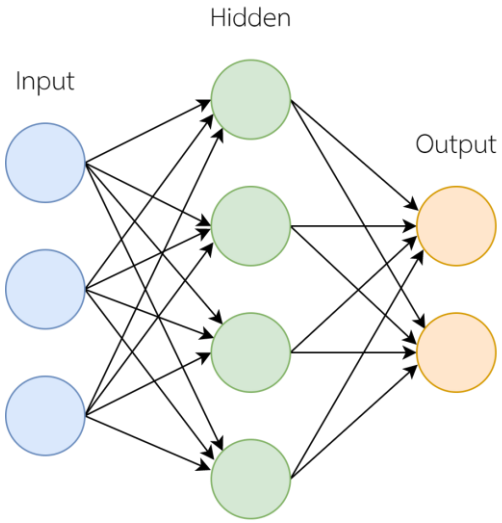
หมายเหตุ. จาก https://en.wikipedia.org/wiki/Axon

เซลล์ประสาทเทียมหรือเรียกได้อีกชื่อว่า Perceptron ถูกจำลองให้อยู่ในรูปแบบที่ง่าย โดยแต่ละเซลล์ประสาทเทียมจะมีช่องรับข้อมูล n ช่อง จากนั้นเซลล์ประสาทจะประมวลผลจากข้อมูลที่รับเข้ามาทั้งหมด โดยนำผลรวมมาถ่วงน้ำหนัก (Weights) และเพิ่มค่าอคติ (Bias) เข้าไปจากนั้นก็ทำฟังก์ชันกระตุ้นบางอย่าง (Activation Function) ดังแผนภาพที่ 2.10



แผนภาพที่ 2.10 กระบวนการของระบบโครงข่ายประสาทเทียม

โครงข่ายประสาทเทียมสามารถประมาณการฟังก์ชันที่ไม่เป็นเชิงเส้นได้ $f^*(x)$ จากการผสมเซลล์ประสาทหลาย ๆ เซลล์จนเป็นโครงข่าย โดยพารามิเตอร์ w จะเก็บน้ำหนักทั้งหมดของ w_i ของเซลล์ทั้งหมดเพื่อนำไปปรับ เพื่อให้โครงข่ายประสาทเทียมสามารถประมาณฟังก์ชันได้ดีที่สุด โครงข่ายประสาทเทียมแบบ Feed-forward จะเป็นโครงข่ายที่เซลล์ประสาทในชั้นต่าง ๆ จะเชื่อมต่อกันในลักษณะการส่งผลลัพธ์ต่อไปข้างหน้าหมายความว่ามันจะไม่มีการส่งผลลัพธ์กลับมาด้านหลัง แต่ละโครงข่ายของประสาทเทียมจะมีหนึ่งชั้นที่ทำหน้าที่รับข้อมูลเข้ามาจะเรียกว่าชั้น Input และมีหนึ่งชั้นที่เป็นผลลัพธ์เรียกว่าชั้น Output ที่จะเป็นผลลัพธ์จากการประมาณ โดยระหว่างชั้น Input และ Output สามารถมีชั้นที่เรียกว่า Hidden Layer ตั้งแต่ 1 ชั้นขึ้นไปได้ดังแผนภาพที่ 2.11



แผนภาพที่ 2.11 โครงข่ายประสาทเทียมประเภท Fully-connected ที่มี Input 1 ชั้น Output 1 ชั้นและ Hidden 1 ชั้น

2.5.1 กระบวนการเรียนรู้ (Learning Process)

เป้าหมายการเรียนรู้คือการหาค่า θ ที่ให้ผลลัพธ์ในการประมาณดีที่สุด ถ้าเป็นการเรียนรู้แบบมีผู้สอน (Supervised Learning) เราจะนำข้อมูล X เข้ามาประมวลผลเพื่อหาว่าเป้าหมาย Y ควรเป็นอะไร โดยกระบวนการเรียนรู้จะเป็นการวนซ้ำซึ่งประกอบไปด้วยขั้นตอนเหล่านี้

1. Forward Pass ข้อมูล X ที่รับเข้ามาจะถูกส่งผ่านเครือข่ายเพื่อทำการคำนวณและพยากรณ์ผลลัพธ์ $Y_{pred} = f(X, \theta)$

2. Loss ค่าพยากรณ์ Y_{pred} จะถูกนำมาเปรียบเทียบกับค่าจริง Y เพื่อคำนวณหาค่า Loss $L(\theta)$ โดยฟังก์ชันสูญเสีย (Loss Function) ที่จะนำมาใช้วัดก็ขึ้นอยู่กับเป้าหมายการเรียนรู้ เช่น Mean Squared Error (MSE) หรือ Logistic Loss Function

3. Back-propagation เป็นกระบวนการที่นำ Gradient ของ Loss $\nabla L(\theta)$ ทั้งหมดที่ถูกคำนวณส่งกลับเข้าไปยังเครือข่าย โดยกระบวนการนี้จะใช้หลัก Chain Rule ในการคำนวณ Derivative ของฟังก์ชัน โดยแต่ละเซลล์ก็จะนำ Loss ทั้งหมดหรือ Global Loss ที่ถูกคำนวณในขั้นตอนที่ 2. ที่มาจากการเปรียบเทียบค่าพยากรณ์ที่ได้มาจากกระบวนการ Forward Pass ในขั้นตอนที่ 1. มาเปรียบเทียบกับค่าจริงจนได้ค่า Global Loss ออกมาใช้คำนวณกับค่า Loss ของแต่ละเซลล์เอง

4. อัพเดทค่าน้ำหนักของทุกเซลล์ซึ่งจะถูกอัพเดทผ่าน Optimizer โดย Optimizer ที่นิยมใช้กันคือ Stochastic Gradient Descent (SGD) ซึ่งเป็นวิธีการวนซ้ำ (Iterative) เพื่อเพิ่มประสิทธิภาพของฟังก์ชันเป้าหมาย (Objective Function) โดยใช้หลักการหาทิศทางของ Gradient แบบสุ่ม ซึ่งวิธีการเพิ่มประสิทธิภาพของฟังก์ชันเป้าหมายนั้น ในที่นี่คือการลดค่าสูญเสียจากฟังก์ชันสูญเสียให้น้อยที่สุด หมายความว่ายิ่งค่าสูญเสียน้อยแบบจำลองก็สามารถทำงานได้ดีมากยิ่งขึ้น

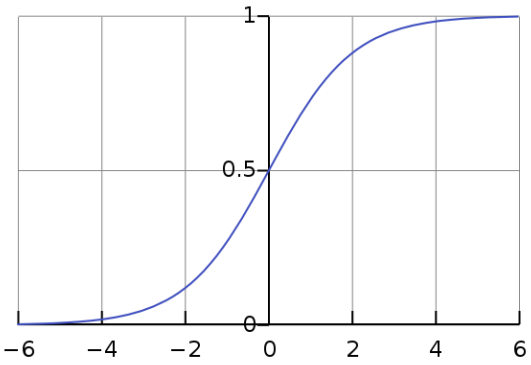
2.5.2 การทำให้เป็นมาตรฐาน (Regularization)

กระบวนการทำให้เป็นมาตรฐานคือวิธีการป้องกันไม่ให้แบบจำลองเกิดปัญหา Overfitting กล่าวคือเป็นการจดจำการเรียนรู้กับชุดข้อมูลที่ใช้ฝึกสอนมากเกินไป ทำให้เมื่อเจอกับชุดข้อมูลที่ไม่เคยเห็นมาก่อนจะไม่สามารถทำงานได้อย่างมีประสิทธิภาพ ซึ่งวิธีการแก้ไขมีอยู่หลากหลายวิธี เช่นการทำ Regularization ด้วยวิธีการ L1 และ L2 ซึ่งเป็นวิธีการที่นำส่วนของ Regularization $\Omega(\theta)$ เข้าไปคำนวณกับฟังก์ชันสูญเสียด้วย หรือวิธีการ Dropout เป็นการลดจำนวนเซลล์ในระหว่างการฝึกสอนเพื่อไม่ให้เกิดการจำมากเกินไปเป็นต้น

2.5.3 ฟังก์ชันกระตุ้น (Activation Function)

ฟังก์ชันกระตุ้นทำหน้าที่รับผลรวมการประมวลผลทั้งหมดจาก Input ที่ส่งเข้ามาแล้วพิจารณาตามแต่ละฟังก์ชันว่าควรทำอย่างไรก่อนส่งผลลัพธ์ต่อไป ฟังก์ชันกระตุ้นที่นิยมใช้กันมีดังนี้

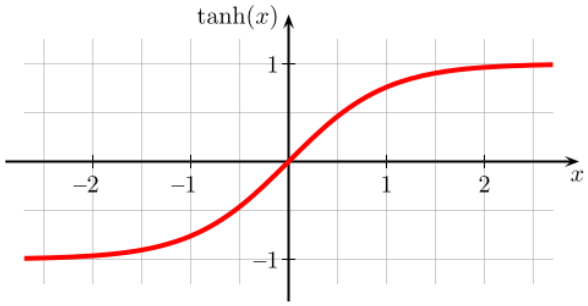
1. ฟังก์ชัน Sigmoid จะเป็นการแปลงข้อมูล X ให้อยู่ในช่วง 0 ถึง 1 ดังแผนภาพที่ 2.12 สามารถคำนวณได้จากสมการ $f(x) = \frac{1}{1+e^{-x}}$



แผนภาพที่ 2.12 ฟังก์ชัน Sigmoid

หมายเหตุ. จาก https://en.wikipedia.org/wiki/Sigmoid_function

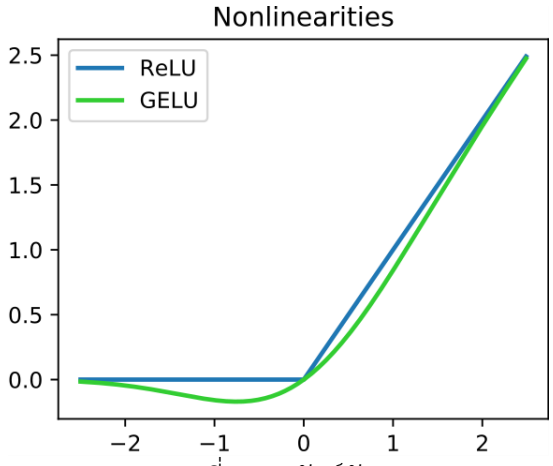
2. ฟังก์ชัน tanh (Tangent Hyperbolic) จะเป็นการแปลงข้อมูล X ให้อยู่ในช่วง -1 ถึง 1 ดังแผนภาพที่ 2.13 สามารถคำนวณได้จากสมการ $f(x) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



แผนภาพที่ 2.13 ฟังก์ชัน tanh

หมายเหตุ. จาก https://commons.wikimedia.org/wiki/File:Hyperbolic_Tangent.svg

3. ฟังก์ชัน ReLU (Rectified Linear Units) เป็นฟังก์ชันที่ได้รับการนิยม โดยฟังก์ชันนี้จะไม่ยอมให้ค่าติดลบดังนั้นค่าลบทั้งหมดจะถูกแปลงให้กลายเป็น 0 ซึ่งข้อดีคือช่วยให้การฝึกสอนลู่เข้าจุดเหมาะสมได้เร็วยิ่งขึ้น และใช้ทรัพยากรในการประมวลผลน้อย ดังแผนภาพที่ 2.14 สามารถคำนวณได้จากสมการ $f(x) = \max(0, x)$



แผนภาพที่ 2.14 ฟังก์ชัน ReLU

หมายเหตุ. จาก https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

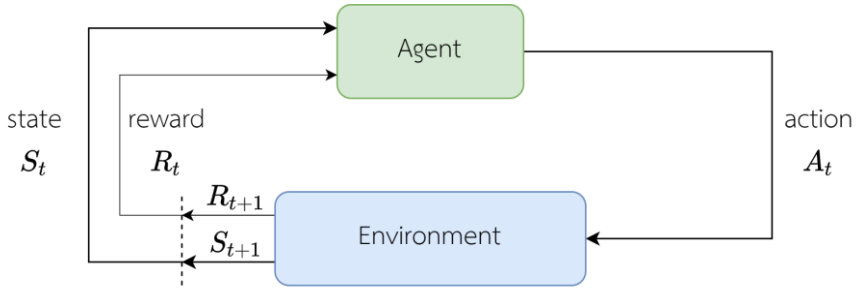
2.5.4 การเรียนรู้แบบกลุ่ม (Batch Learning)

เนื่องจากการเรียนรู้ของโครงข่ายประสาทเทียมเป็นการเรียนรู้ตามจำนวนรอบหรือ Epoch ซึ่งแต่ละ Epoch ก็จะประกอบไปด้วยการทำ Forward Pass 1 ครั้งและ Back-propagation 1 ครั้งจากข้อมูลทั้งหมดที่ได้รับมาดังนั้นเราสามารถคำนวณหาค่า Gradient ที่แท้จริงได้ จะเรียกว่าการเรียนรู้แบบเต็มรูปแบบ (Full Batch Learning) แต่การเรียนรู้แบบนี้มักจะมีปัญหาเมื่อชุดข้อมูลสำหรับการฝึกสอนมีขนาดใหญ่เกินไปทำให้ต้องใช้ทรัพยากรที่สูงและใช้เวลาในฝึกสอนที่นาน ดังนั้นจึงมีการเรียนรู้แบบกลุ่มขึ้นมา โดยการเรียนรู้แบบกลุ่มคือกระบวนการสุ่มชุดข้อมูลการเรียนรู้ (mini-batches) แทนการเรียนรู้ทั้งหมดในครั้งเดียว ทำให้สามารถหา Gradient จากการหาค่าเฉลี่ยแทนได้ ซึ่งช่วยให้ Gradient มีความแม่นยำมากยิ่งขึ้น อีกทั้งความแปรปรวนยังน้อยลงส่งผลให้ใช้เวลาในฝึกฝนน้อยลงอีกด้วย

2.6 การเรียนรู้แบบเสริมแรง (Reinforcement Learning)

วิธีการเรียนรู้แบบเสริมแรง (Reinforcement Learning) ตัวแทน (Agent) จะเรียนรู้พฤติกรรมเฉพาะผ่านการลองผิดลองถูก โดยตัวแทนจะตอบโต้กับสภาพแวดล้อม (Environment) เพื่อเก็บรวบรวมประสบการณ์ จะพบว่ากระบวนการของการเรียนรู้แบบเสริมแรงจะเรียนรู้เป็นลำดับขั้นตอน (Sequential) และวนทำซ้ำไปเรื่อย ๆ ด้วยเหตุนี้เราจึงกำหนดปัญหาของวิธีการเรียนรู้แบบเสริมแรงให้อยู่ในรูปของ Markov Decision Process ได้เนื่องจาก Markov Decision Process เป็นกรอบทางคณิตศาสตร์สำหรับการสร้างแบบจำลองการตัดสินใจในสถานการณ์ที่ผลลัพธ์เกิดขึ้นแบบสุ่มและถูกเลือกโดยผู้ตัดสินใจหรือตัวแทนนั้นเอง ดังนั้นกระบวนการเรียนรู้ของวิธีการเรียนรู้แบบเสริมแรงสามารถแสดงได้ดังแผนภาพที่ 2.15 โดนกระบวนการจะเป็นดังนี้ ในแต่ละเวลา $t = 1,2,3 \ldots$ ตัวแทนจะไปอยู่ในสถานะ (State) $s_t \in S$ และเลือกการกระทำ (Action) $a_t \in A(s)$ เพียงหนึ่งการกระทำที่สามารถทำได้ในสถานะนั้น โดยการกระทำจะส่งผลต่อการเปลี่ยนแปลงของสภาพแวดล้อม

และตัวแทนจะย้ายไปอยู่ ณ สถานะใหม่ s_{t+1} และจะได้รับรางวัล (Reward) R_{t+1} จากสภาพแวดล้อมที่เปรียบเสมือนข้อเสนอแนะที่บ่งบอกถึงความเหมาะสมที่เลือกการกระทำ a_t ในสถานะ s_t



แผนภาพที่ 2.15 แนวคิดเบื้องหลังของ Reinforcement Learning

2.6.1 Markov Decision Process (MDP)

Markov Decision Process (MDP) เป็นเฟรมเวิร์กทางคณิตศาสตร์ที่ใช้ในการสร้างโมเดลในการตัดสินใจ (Decision Making) ในสถานการณ์ที่มีความไม่แน่นอน (Stochastic) โดยข้อสันนิษฐานของ Markov คือสถานะในอดีตและอนาคตต้องเป็นอิสระจากกัน ความหมายคือสถานะและพฤติกรรมของสภาพแวดล้อมในเวลา t ไม่ได้รับอิทธิพลจากการตอบโต้ระหว่างตัวแทนกับสภาพแวดล้อมในอดีต $a_1, \dots, a_{t-1}$ ถ้างานที่ต้องทำสามารถเป็นไปตามข้อสันนิษฐานของ Markov ได้ ซึ่งตรงกับวิธีการเรียนรู้แบบเสริมแรงดังนั้นปัญหาที่วิธีการเรียนรู้แบบเสริมแรงนั้นจะเรียนรู้จึงสามารถกำหนดให้อยู่ในรูปแบบของ Markov Decision Process $\left(\mathrm{S}, \mathrm{A}, P^a_{s,s'}, R^a_{s,s'}, \gamma\right)$ ได้ ซึ่งประกอบไปด้วย

1. เซ็ตของสถานะ S
2. เซ็ตของการกระทำ A และ $\mathrm{A}(s)$ คือเซ็ตของการกระทำที่สามารถทำได้ในสถานะ s
3. ความน่าจะเป็นของการเปลี่ยนแปลง (Transition Probability) $P^a_{s,s'}$: $(S \times A \times S) \rightarrow [0,1]$ ก็คือความน่าจะของการเปลี่ยนแปลงสถานะจาก s เป็น s' เมื่อทำการกระทำ a ในสถานะ s ณ เวลา t
4. ความน่าจะเป็นของรางวัล (Reward Probability) $R^a_{s,s'}$: $(S \times A \times S) \rightarrow IR$ ก็คือรางวัลปัจจุบันที่ตัวแทนจะได้รับเมื่อเปลี่ยนแปลงสถานะจาก s เป็น s'
5. ปัจจัยส่วนลด (Discount Factor) $\gamma \in [0,1]$ ใช้สำหรับคำนวณรางวัลที่คาดหวังที่ถูกลดทอน (Discounted Expected Reward)

โดย Markov Decision Process (MDP) จะจำกัด (Finite) ก็ต่อเมื่อเซ็ตของสถานะ S และเซ็ตของการกระทำ A จำกัด

2.6.2 Discounted Cumulative Expected Reward

การฝึกสอนตัวแทนให้มีประสิทธิภาพนั้นขึ้นอยู่กับการกำหนดเป้าหมาย (Goal) ให้กับตัวแทน ซึ่งการกำหนดเป้าหมายที่ดีควรจะเป็นการเพิ่มรางวัลที่คาดหวังที่ถูกลดทอนสูงสุดในระยะยาวมากกว่ารางวัลที่คาดหวังที่ถูกลดทอนสูงสุดในปัจจุบัน จะเรียกว่ารางวัลที่คาดหวังสะสมที่ถูกลดทอน (Discounted Cumulative Expected Reward หรือ Return) คือผลรวมของรางวัลในอนาคตที่เป็นไปได้เขียนได้ดังสมการที่ 2.8 โดยปัจจัยส่วนลด $\gamma \in [0,1]$ เป็นตัวกำหนดระยะการพิจารณารางวัลในอนาคต ถ้า $\gamma = 1$ หมายความว่ารางวัลในอนาคตมีน้ำหนักเท่ากัน แต่ถ้า $\gamma = 0$ หมายความว่าพิจารณาเฉพาะรางวัลในปัจจุบันเท่านั้น

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.8}$$

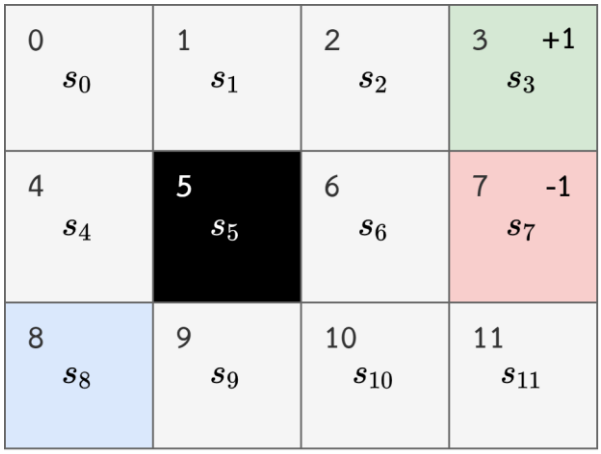
โดยที่ G คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนประกอบไปด้วย $\{G_1, G_2, \dots\}$

t คือ timestep ณ ปัจจุบัน

R_t คือ รางวัลที่จะได้รับใน timestep นั้น ๆ

γ คือ ปัจจัยส่วนลด

จากแผนภาพที่ 2.16 แสดงถึงสภาพแวดล้อมของเกม Grid World ในเกมนี้มีกฎคือตัวแทนจะเริ่มต้นในสถานะที่ 8 ก็คือช่องตำแหน่งที่ 8 (s_8) จากแผนภาพและถ้าหาทางออกได้ จะได้รางวัลเป็น +1 และจบเกมทันทีในที่นี่คือสถานะที่ 3 ก็คือช่องตำแหน่งที่ 3 (s_3) แต่ถ้าตกกับดักคือสถานะที่ 7 ก็คือช่องตำแหน่งที่ 7 (s_7) จะได้รางวัลเป็น -1 และจบเกมทันที แต่ถ้าหากตัวแทนเลือกการการกระทำที่ติดขอบก็จะอยู่ที่ตำแหน่งเดิมเช่น ตัวแทนอยู่ตำแหน่งสถานะที่ 9 ก็คือช่องตำแหน่งที่ 9 (s_9) และเลือกเดินขึ้นไปสถานะที่ 5 ก็คือช่องตำแหน่งที่ 5 (s_5) ซึ่งเป็นตำแหน่งที่ไม่สามารถไปได้ ดังนั้นตัวแทนจะคงอยู่ที่ตำแหน่งที่ 9 เช่นเดิม ดังนั้นตัวแทนที่ปฏิสัมพันธ์กับสภาพแวดล้อมควรจะหาเส้นทางไปยังทางออกให้ได้เร็วที่สุดและไม่ตกลงบนสถานะที่เป็นกับดัก เนื่องจากถ้าตัวแทนไม่รีบหาทางออก Timestep ของตัวแทนก็จะเพิ่มขึ้นส่งผลให้ค่า γ จะมีค่าน้อยลงเรื่อย ๆ ซึ่งส่งผลให้รางวัลที่คาดหวังสะสมที่ถูกลดทอนในระยะยาวลดน้อยลงนั่นเอง



แผนภาพที่ 2.16 สภาพแวดล้อมของเกม Grid World

2.6.3 Episodic และ Continuous Tasks

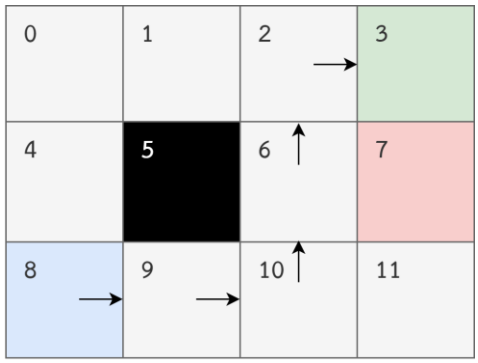
งานของการเรียนรู้แบบเสริมแรงสามารถแบ่งออกได้เป็น 2 ประเภทหลัก ๆ คือ Episodic task และ Continuous task โดย Episodic task คืองานที่สามารถฝึกสอนออกเป็นตอน (Episode) ได้ กล่าวคือเป็นงานที่มีจุดจบหรือมีสถานะสุดท้ายเช่น เกมแข่งรถที่เริ่มจากจุดเริ่มต้นและเข้าเส้นชัย ดังนั้นเมื่องานจบ สภาพแวดล้อมก็จะถูกรีเซ็ตและตัวแทนจะเริ่มต้นใหม่ในตอนถัดไป ในสถานะสุดท้ายจะถือว่ารางวัลเป็น 0 และถือว่าเป็นเวลา T ของเวลาที่จำกัด ส่วน Continuous task คือ คือปัญหาที่ไม่สามารถกำหนดเป็นตอนได้นั่นคือเป็นปัญหาแบบต่อเนื่องดังนั้น $\mathrm{T} = \infty$

2.6.4 Policy Function และ Value Function

ฟังก์ชันนโยบาย (Policy Function) π จะเป็นการจำลองการแจกแจงความน่าจะเป็น $\pi(a|s)$ บนจำนวนการกระทำที่สามารถทำได้ $a \in \mathrm{A}(s)$ ของแต่ละสถานะ s เช่น $\pi(A_t|S_t)$ คือความน่าจะเป็นที่จะเลือกการกระทำ A_t ณ สถานะ S_t ดังนั้นตัวแทนก็จะทำการสุ่มตัวอย่างการกระทำจากการแจกแจงความน่าจะเป็น $\pi(a|s)$ โดยเป้าหมายของเราคือการหานโยบายที่เหมาะสมที่สุด เพื่อให้ได้รางวัลที่คาดหวังที่ถูกลดทอนมากที่สุดดังแผนภาพที่ 2.17 แสดงถึงนโยบายที่เหมาะสมที่สุดในแต่ละสถานะ

Optimal Policy

State	Action
2	Right
6	Up
8	Right
9	Right
10	Up



แผนภาพที่ 2.17 นโยบายที่เหมาะสมที่สุดในแต่ละสถานะของเกม Grid World

ฟังก์ชันค่า (Value Function) $v_\pi(s)$ เป็นค่าประมาณความเหมาะสมของตัวแทนที่จะไปยังสถานะ s ดังนั้น $v_\pi(s)$ คือรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π ดังสมการที่ 2.9

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s\right], \text{for all } s \in S \qquad (2.9)$$

โดยที่ $v_\pi(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

π คือ นโยบาย

t คือ timestep ณ ปัจจุบัน

G_t คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนใน timestep นั้น ๆ

R_t คือ รางวัลที่จะได้รับใน timestep นั้น ๆ

γ คือ ปัจจัยส่วนลด

s คือ สถานะปัจจุบัน

S_t คือ สถานะที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

จากสมการที่ 2.9 สามารถเขียนให้อยู่ในรูปแบบการวนซ้ำ (Recursive) ได้ รูปแบบการวนซ้ำเรียกว่าสมการ Bellman ของ v_π สามารถเขียนได้ดังสมการที่ 2.10 สมการ Bellman ค่าของสถานะ s จะขึ้นอยู่กับสถานะถัดไปที่เป็นไปได้เท่านั้น s' ในขณะที่แต่ละสถานะจะถูกถ่วงน้ำหนักตามการแจกแจงความน่าจะเป็น $P^a_{s,s'}$ ดังนั้นการแก้ปัญหาของการเรียนรู้แบบเสริมแรงส่วนใหญ่จะเป็นการหาการประมาณสมการ Bellman ที่เหมาะสมจากการประมาณค่าของสถานะถัดไป

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\right] \\ &= \sum_a \pi(a|s) \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma v_\pi(s')\right] \end{aligned} \qquad (2.10)$$

โดยที่ $v_\pi(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

π คือ นโยบาย

a คือ การกระทำปัจจุบัน

s คือ สถานะปัจจุบัน

s' คือ สถานะถัดไป

γ คือ ปัจจัยส่วนลด

G_t คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนใน timestep นั้น ๆ

R_t คือ รางวัลที่จะได้รับใน timestep นั้น ๆ

$\pi(a|s)$ คือ ความน่าจะเป็นที่จะเลือกการกระทำ a ณ สถานะ s

$P^a_{s,s'}$ คือ ความน่าจะของการเปลี่ยนแปลงสถานะจาก s เป็น s' เมื่อทำการกระทำ a ในสถานะ s

$R^a_{s,s'}$ คือ รางวัลที่คาดหวังจะได้รับเมื่อทำการกระทำ a ในสถานะ s และเปลี่ยนสถานะไปเป็น s'

$v_\pi(s')$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s' เมื่อตัวแทนปฏิบัติตามนโยบาย π

ฟังก์ชัน State-Action value $q_\pi(s,a)$ เป็นการประมาณหาความเหมาะสมของการกระทำ a ณ สถานะ s โดย $q_\pi(s,a)$ คือรางวัลที่คาดหวังสะสมที่ถูกลดทอนจากการกระทำ a ณ สถานะ s ตามนโยบาย π ดังสมการที่ 2.11

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,|S_t = s, A_t = a\right] \quad \text{for all } s \in S \text{ and } a \in A \tag{2.11}$$

โดยที่ $q_\pi(s,a)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนจากการกระทำ a ณ สถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

π คือ นโยบาย

t คือ timestep ณ ปัจจุบัน

G_t คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนใน timestep นั้น ๆ

R_t คือ รางวัลที่จะได้รับใน timestep นั้น ๆ

γ คือ ปัจจัยส่วนลด

s คือ สถานะปัจจุบัน

a คือ การกระทำปัจจุบัน

S_t คือ สถานะที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

A_t คือ การกระทำที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

เป้าหมายของการเรียนรู้แบบเสริมกำลังคือการหานโยบายที่เหมาะสม π_* นั่นก็คือนโยบายที่ดีคือนโยบายที่ $\pi \geq \pi'$ เมื่อค่าฟังก์ชันของนโยบายใหม่ดีกว่า $v_\pi(s) \geq v'_\pi(s)$ สำหรับทุกสถานะ $s \in S$ ถ้าฟังก์ชัน State-Action value เหมาะสมแล้วนโยบายที่เหมาะสมก็จะถูกใช้โดยตัวแทน โดยมีความเป็นไปได้ที่จะมีนโยบายที่เหมาะสมหลายนโยบายซึ่งก็จะนำไปสู่ฟังก์ชัน State-

Action value ฟังก์ชันที่เหมาะสมเช่นเดียวกัน ดังนั้นฟังก์ชัน State-Action value ที่เหมาะสมสามารถเขียนได้ดังสมการที่ 2.12

$$v_*(s) = \max_{\pi} v_\pi(s) \text{ for all } s \in S \tag{2.12}$$

โดยที่ $v_*(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบายที่ดีที่สุด (Optimal Policy)

π คือ นโยบาย

$v_\pi(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

ดังนั้นผลลัพธ์จากนโยบายที่เหมาะสมก็จะได้ฟังก์ชัน State-Action value ที่เหมาะสม q_* ดังสมการที่ 2.13

$$\begin{aligned} q_*(s,a) &= \max_{\pi} q_\pi(s,a) \text{ for all } s \in S \text{ and } a \in A(s) \\ &= \mathbb{E}[R_{t+1} + \gamma v_*(s') | S_t = s, A_t = a] \end{aligned} \tag{2.13}$$

โดยที่ $q_*(s,a)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนจากการกระทำ a ณ สถานะ s เมื่อตัวแทนปฏิบัติตามนโยบายที่ดีที่สุด (Optimal Policy)

π คือ นโยบาย

t คือ timestep ณ ปัจจุบัน

s คือ สถานะปัจจุบัน

a คือ การกระทำปัจจุบัน

s' คือ สถานะถัดไป

S_t คือ สถานะที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

A_t คือ การกระทำที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

R_{t+1} คือ รางวัลที่จะได้รับใน timestep ถัดไป

γ คือ ปัจจัยส่วนลด

$q_\pi(s,a)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนจากการกระทำ a ณ สถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

$v_*(s')$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดไป s' เมื่อตัวแทนปฏิบัติตามนโยบายที่ดีที่สุด (Optimal Policy)

ดังนั้นสมการที่ 2.14 เป็นสมการเหมาะสมของ Bellman ที่สร้างมาจากสมการก่อนหน้า

$$
\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(s') | S_t = s, A_t = a] \\
&= \max_a \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma v_*(s') \right] \\
&= \max_a \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma \max_{a'} q_{\pi_*}(s',a') \right] \qquad (2.14)
\end{aligned}
$$

โดยที่ $v_*(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบายที่ดีที่สุด (Optimal Policy)

π_* คือ นโยบายที่ดีที่สุด

t คือ timestep ณ ปัจจุบัน

s คือ สถานะ

a คือ การกระทำ

s' คือ สถานะถัดไป

a' คือ การกระทำที่จะทำในสถานะถัดไป

S_t คือ สถานะที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

A_t คือ การกระทำที่เป็นไปได้ทั้งหมดใน timestep นั้น ๆ

γ คือ ปัจจัยส่วนลด

$P^a_{s,s'}$ คือ ความน่าจะของการเปลี่ยนแปลงสถานะจาก s เป็น s' เมื่อทำการกระทำ a ในสถานะ s

$R^a_{s,s'}$ คือ รางวัลที่จะได้รับเมื่อทำการกระทำ a ในสถานะ s และเปลี่ยนสถานะไปเป็น s'

$v_*(s')$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s' เมื่อตัวแทนปฏิบัติตามนโยบายที่ดีที่สุด (Optimal Policy)

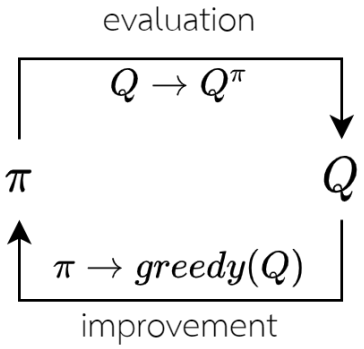
$q_{\pi_*}(s',a')$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนจากการกระทำ a' ณ สถานะถัดไป s' เมื่อตัวแทนปฏิบัติตามนโยบาย π ที่ดีที่สุด

2.6.5 วิธีการ Monte Carlo

วิธีการ Monte Carlo เป็นวิธีการหนึ่งในการแก้ปัญหางานแบบ Episodic โดยที่ไม่รู้โมเดลของ Environment กล่าวคือไม่รู้ค่าของ Transition Probability โดยการแก้ปัญหาจะเป็นการวนทำซ้ำเพื่อหาจุดลู่เข้าของจำนวนตอนที่เพิ่มมากขึ้นเพื่อให้ได้นโยบายที่เหมาะสมก็คือการหาค่าเฉลี่ยรางวัลที่ได้จากการเล่นนั้นเอง

วิธีการ Monte Carlo สามารถแบ่งการแก้ปัญหาได้เป็น 2 รูปแบบคือ 1. Monte Carlo Prediction และ 2. Monte Carlo Control สิ่งที่แตกต่างระหว่างการแก้ปัญหาทั้ง 2 รูปแบบคือแนวทางในการหา Optimal Policy วิธีการ Monte Carlo Prediction จะเป็นการประมาณค่า

รางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละสถานะ ดังนั้นนโยบายในการเลือกการกระทำจะถูกกำหนดไว้ตายตัวเราต้องหาหาค่าประมาณของแต่ละสถานะออกมาแทนว่าแต่ละสถานะดีมากน้อยเพียงใด แต่ในทางกลับกันวิธีการ Monte Carlo Control จะเป็นการหานโยบายที่ดีที่สุดที่จะทำให้ได้รับรางวัลที่คาดหวังสะสมที่ถูกลดทอนมากที่สุด ดังนั้นกระบวนการจะเป็นการประเมินประสิทธิภาพของนโยบาย และทำการปรับปรุงนโยบายให้ดีขึ้นเรื่อย ๆ ดังแผนภาพที่ 2.18



แผนภาพที่ 2.18 กระบวนการประเมินและปรับปรุงนโยบายของวิธี Monte Carlo Control

จากที่กล่าวไปวิธีการ Monte Carlo Control เป็นการเรียนรู้เพื่อหานโยบายที่ดีที่สุด ดังนั้นกระบวนการหานโยบายที่ดีที่สุดสามารถแบ่งได้เป็น 2 ประเภทคือ 1. วิธีการแบบ On-Policy และ 2. วิธีการแบบ Off-Policy ซึ่งสิ่งที่แตกต่างระหว่างทั้ง 2 วิธีการคือ วิธีการ On-Policy จะประเมิน (Evaluate) และปรับปรุง (Improvement) นโยบายเดียวกันกับที่ใช้ในการตัดสินใจ แต่ Off-Policy จะประเมินและปรับปรุงนโยบายอีกตัว ที่ไม่ได้ใช้ในการตัดสินใจเพื่อเก็บรวบรวมข้อมูล

2.6.6 วิธีการ Monte Carlo Prediction

วิธีการ Monte Carlo Prediction นั้นเป็นการประมาณการค่าของ State หรือ State-value ซึ่งกระบวนการที่จะสามารถประมาณค่าออกมาได้จำเป็นต้องมีข้อมูลที่เพียงพอในการคำนวณ ซึ่งข้อมูลเหล่านั้นมาจากการปฏิบัติสัมพันธ์ของตัวแทนตามนโยบายกับสภาพแวดล้อมจนจบเกมเพื่อเก็บเกี่ยวข้อมูลรางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละ Episode โดยเราต้องมีการกำหนดนโยบายให้กับตัวแทนเพื่อเป็นตัวเลือกการกระทำในแต่ละ Timestep ซึ่งนโยบายที่นิยมใช้กันมีดังนี้ 1. นโยบายแบบสุ่มหรือ Random Policy เป็นนโยบายที่จะทำการสุ่มการกระทำจากการกระทำที่เป็นไปได้ทั้งหมดในแต่ละ Timestep และ 2. นโยบายละโมบด้วยค่า Epsilon โดยจะกล่าวถึงในหัวข้อถัด ๆ ไป ซึ่งการคำนวณหารางวัลที่คาดหวังสะสมที่ถูกลดทอนจากสมการที่ 2.8 นั้นเราสามารถนำมาคำนวณได้ 2 วิธีคือ 1. First-visit Monte Carlo และ 2. Every-visit Monte Carlo สิ่งที่แตกต่างระหว่างการคำนวณทั้ง 2 แบบคือ จำนวนครั้งที่ไปอยู่ในสถานะนั้น ๆ ใน 1 Episode ว่าจะนับเพียงครั้งแรกที่ไปอยู่หรือนับทุกครั้งที่ไปอยู่ ดังนั้นจากสมการที่ 2.9 เราสามารถใช้การประมาณการค่าจากประสบการณ์แทนได้ดังสมการที่ 2.15

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$
$$v_\pi(s) \approx \frac{1}{N}\sum_{t=1}^{N} G_t(s) \quad (2.15)$$

โดยที่ $v_\pi(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s เมื่อตัวแทนปฏิบัติตามนโยบาย π

N คือ จำนวนรางวัลที่คาดหวังสะสมที่ถูกลดทอนทั้งหมด

$G_t(s)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละ Episode ของแต่ละ State

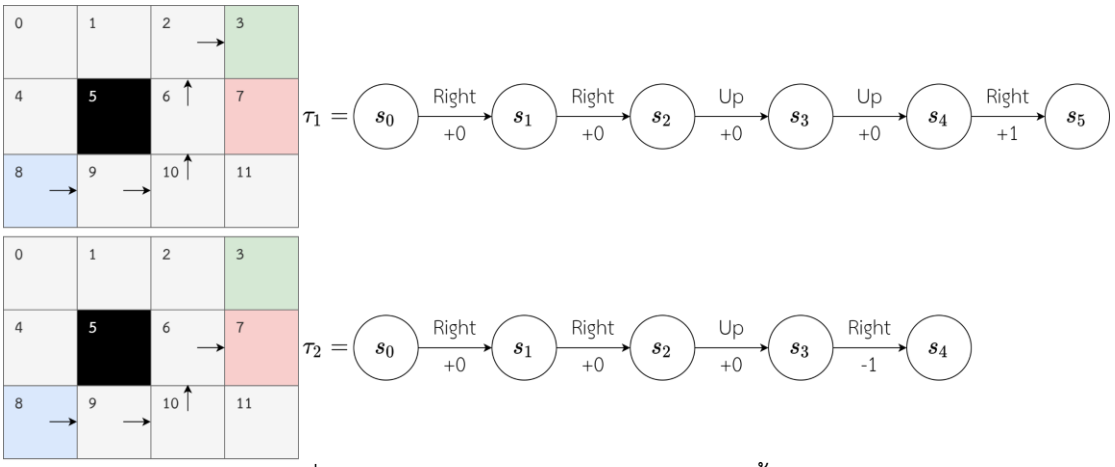
เมื่อเราสามารถหารางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละ Episode ได้แล้วโดยปกติของการฝึกสอนเรามักจะไม่อัพเดทบ่อยครั้ง ดังนั้นเราจะนำค่ารางวัลที่คาดหวังสะสมที่ถูกลดทอนไปเก็บไว้ใน List ของแต่ละสถานะจนถึงจำนวนรอบของการอัพเดทจึงค่อยนำค่ารางวัลที่คาดหวังสะสมที่ถูกลดทอนเหล่านี้มาเฉลี่ยดังสมการที่ 2.15 และนำค่ามาแทนที่ในตารางค่า V (V-table) ดังแผนภาพที่ 2.19

V-table $V(s)$

State:	0	1	2	3	4	5	6	7	8	9	10	11
	0	0	0	0	0	0	0	0	0	0	0	0

แผนภาพที่ 2.19 ตารางค่า V ของแต่ละสถานะของเกม Grid World

จากตัวอย่างสภาพแวดล้อมของเกม Grid World เราสามารถนำวิธีการ Monte Carlo มาแก้ปัญหานี้ได้ เพื่อให้การแสดงตัวอย่างเป็นไปได้ง่าย จึงขอกำหนดให้ค่า $\gamma = 0.9$ และตัวแทนมีการปฏิสัมพันธ์กับสภาพแวดล้อมจำนวน 2 Episode โดยมีลำดับการปฏิสัมพันธ์ (Trajectory) ทั้ง 2 Episode ดังแผนภาพที่ 2.20



แผนภาพที่ 2.20 ลำดับการปฏิสัมพันธ์ของตัวแทนทั้ง 2 Episode

จากแผนภาพที่ 2.20 ในการปฏิสัมพันธ์ของ Episode ที่ 1 สามารถคำนวณหารางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละสถานะสามารถคำนวณได้จากสมการที่ 2.8 ดังนี้

$$\begin{aligned}
G_0 &= R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5 \\
&= 0 + (0.9)(0) + (0.9)^2(0) + (0.9)^3(0) + (0.9)^4(1) \\
&= 0.6561 \\
G_1 &= R_2 + \gamma R_3 + \gamma^2 R_4 + \gamma^3 R_5 \\
&= 0 + (0.9)(0) + (0.9)^2(0) + (0.9)^3(1) \\
&= 0.729 \\
G_2 &= R_3 + \gamma R_4 + \gamma^2 R_5 \\
&= 0 + (0.9)(0) + (0.9)^2(1) \\
&= 0.81 \\
G_3 &= R_4 + \gamma R_5 \\
&= 0 + (0.9)(1) \\
&= 0.9 \\
G_4 &= R_5 \\
&= 1
\end{aligned}$$

การปฏิสัมพันธ์ของ Episode ที่ 2 สามารถคำนวณหารางวัลที่คาดหวังสะสมที่ถูกลดทอนในแต่ละสถานะสามารถคำนวณได้จากสมการที่ 2.8 ดังนี้

$$\begin{aligned}
G_0 &= R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 \\
&= 0 + (0.9)(0) + (0.9)^2(0) + (0.9)^3(-1) \\
&= -0.729 \\
G_1 &= R_2 + \gamma R_3 + \gamma^2 R_4 \\
&= 0 + (0.9)(0) + (0.9)^2(-1) \\
&= -0.81 \\
G_2 &= R_3 + \gamma R_4 \\
&= 0 + (0.9)(-1) \\
&= -0.9 \\
G_3 &= R_4 \\
&= -1
\end{aligned}$$

เมื่อจบ 2 Episode สถานะที่มีการปฏิสัมพันธ์ทั้ง 2 Episode ก็จะมีรางวัลที่คาดหวังสะสมที่ถูกลดทอนจำนวน 2 ค่าดังนั้นเราสามารถหาค่าประมาณของสถานะได้ดังสมการที่ 2.15 ดังนี้

$$\begin{aligned}
v_\pi(s_2) &\approx \frac{1}{1}\sum_{t=1}^{1} G_t(s_2) \\
&\approx 1 \\
v_\pi(s_6) &\approx \frac{1}{2}\sum_{t=1}^{2} G_t(s_6) \\
&\approx \frac{1}{2}(0.9 - 1) \\
&\approx -0.05
\end{aligned}$$

$$
\begin{aligned}
v_\pi(s_8) &\approx \frac{1}{2}\sum_{t=1}^{2} G_t(s_8) \\
&\approx \frac{1}{2}(0.6561 - 0.729) \\
&\approx -0.03645 \\
v_\pi(s_9) &\approx \frac{1}{2}\sum_{t=1}^{2} G_t(s_9) \\
&\approx \frac{1}{2}(0.729 - 0.81) \\
&\approx -0.0405 \\
v_\pi(s_{10}) &\approx \frac{1}{2}\sum_{t=1}^{2} G_t(s_{10}) \\
&\approx \frac{1}{2}(0.81 - 0.9) \\
&\approx -0.045
\end{aligned}
$$

หลังจากการคำนวณค่าประมาณสถานะเสร็จก็นำค่าประมาณสถานะกลับไปแทนที่ในตารางค่า V จะได้ดังแผนภาพที่ 2.21 และเมื่อวนรอบทำซ้ำจนลู่เข้าสู่จุด Optimal ตารางค่า V จะมีค่าดังแผนภาพที่ 2.22

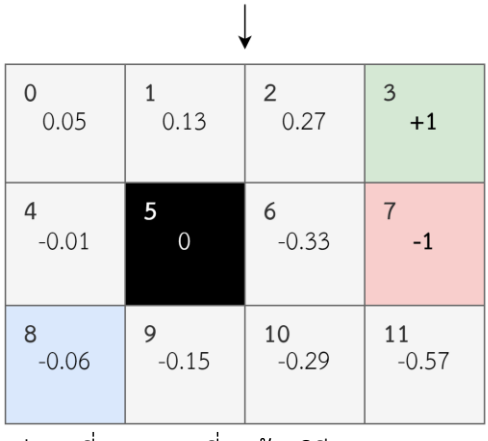
V-table $V(s)$

State:	0	1	2	3	4	5	6	7	8	9	10	11
	0	0	1	1	0	0	-0.05	-1	-0.03	-0.04	-0.04	0

แผนภาพที่ 2.21 ตารางค่า V ที่ถูกอัพเดทหลังปฏิสัมพันธ์ 2 Episode

V-table $V(s)$

State:	0	1	2	3	4	5	6	7	8	9	10	11
	0.05	0.13	0.27	1	-0.01	0	-0.33	-1	-0.06	-0.15	-0.29	-0.57



แผนภาพที่ 2.22 ตารางค่า V ที่เหมาะสมที่สุดด้วยวิธีการ Monte Carlo Prediction

2.6.7 วิธีการ Temporal Difference

วิธีการ Temporal Difference (TD) ถูกคิดค้นเพื่อแก้ปัญหาแบบต่อเนื่องหรือ Continuous Task ที่วิธีการแบบ Monte Carlo ไม่สามารถแก้ปัญหาได้ เพราะวิธีการ Monte Carlo จะทำการเรียนรู้เมื่อจบ Episode เท่านั้น แต่วิธีการ Temporal Difference นั้นนโยบายจะถูกอัพเดทในแต่ละเวลา t ดังนั้นไม่จำเป็นที่ต้องเล่นจนจบ Episode เพื่อทำการอัพเดทเหมือนกับวิธีการ Monte Carlo มากกว่านั้นวิธีการนี้ไม่จำเป็นต้องรู้ค่า Transition Probability ของสภาพแวดล้อม จากการทดลองพบว่าวิธีการ Temporal Difference จะลู่เข้าจุดเหมาะสมได้เร็วกว่าวิธีการ Monte Carlo

เนื่องจากวิธีการของ Temporal Difference ไม่จำเป็นต้องเล่นจนจบตอน ดังนั้นรางวัลที่คาดหวังสะสมที่ถูกลดทอน G_t จะไม่สามารถหาได้ จึงใช้การประมาณการด้วยวิธีการ TD-target แทน วิธีการนี้จะเป็นการประมาณสมการ Bellman ที่ 2.14 โดย TD-target หาได้จากการนำผลรวมของรางวัลในปัจจุบันและค่ารางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดไป $R_{t+1} + \gamma V(S_{t+1})$ เรียกว่า TD-target ดังสมการที่ 2.16

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.16}$$

โดย $V(S_t)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ S ณ Timestep t

t คือ timestep ณ ปัจจุบัน

γ คือ ปัจจัยส่วนลด

α คือ อัตราการเรียนรู้

R_{t+1} คือ รางวัลที่จะได้รับใน timestep ถัดไป

โดยกระบวนการทำงานจะมีดังนี้ ในแต่ละรอบการทำงานของแต่ละตอน การกระทำ A_t ได้มาจากนโยบาย π ณ สถานะ S_t เมื่อทำการกระทำ A_t ตัวแทนก็จะได้รับรางวัล R_{t+1} และย้ายไปอยู่สถานะถัดไป S_{t+1} จากนั้นทำการคำนวณหาค่า TD-error ซึ่งหามาจากส่วนต่างระหว่าง TD-target และ $V(S_t)$ จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า V (V-table) จนกระทั่งตารางค่า V ลู่เข้าค่าที่เหมาะสม

จากตัวอย่างสภาพแวดล้อมของเกม Grid World เราสามารถนำวิธีการ Temporal Difference มาแก้ปัญหานี้ได้ เพื่อให้การแสดงตัวอย่างเป็นไปได้ง่าย จึงขอกำหนดให้ค่า $\gamma = 0.9$ $\alpha = 0.5$ และกำหนดให้ทุก Timestep มีรางวัลเป็น -0.1 เพื่อให้เห็นถึงการคำนวณในแต่ละครั้งและกำหนดให้ตารางค่า V มีค่าเริ่มต้นดังแผนภาพที่ 2.23

V-table $V(s)$

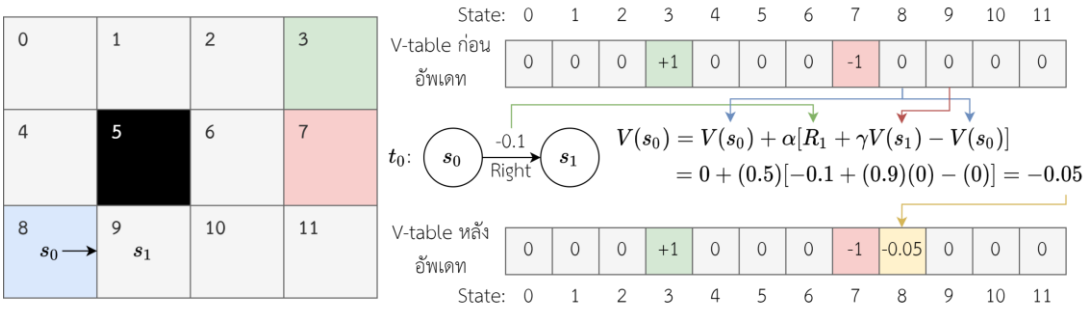
State:	0	1	2	3	4	5	6	7	8	9	10	11
	0	0	0	+1	0	0	0	-1	0	0	0	0

แผนภาพที่ 2.23 ตารางค่า V เริ่มต้นของวิธีการ Temporal Difference

สมมุติให้ตัวแทนเริ่มต้นที่ตำแหน่งสถานะที่ 8 และมีการกระทำไปทางขวาทำให้ไปอยู่ในตำแหน่งสถานะที่ 9 เราสามารถคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep แรกได้จากสมการที่ 2.16 ดังนี้

$$\begin{aligned} V(S_0) &= V(S_0) + \alpha[R_1 + \gamma V(S_1) - V(S_0)] \\ &= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\ &= 0 + (0.5)(-0.1) \\ &= -0.05 \end{aligned}$$

จากการคำนวณหาค่ารางวัลที่คาดหวังของสถานะ S_0 สามารถแสดงได้ดังแผนภาพที่ 2.24



แผนภาพที่ 2.24 วิธีการคำนวณหาค่ารางวัลที่คาดหวังของสถานะของ timestep แรก

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า V จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 9 มีการกระทำไปทางขวาทำให้ไปอยู่ในตำแหน่งสถานะที่ 10 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$\begin{aligned} V(S_1) &= V(S_1) + \alpha[R_2 + \gamma V(S_2) - V(S_1)] \\ &= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\ &= 0 + (0.5)(-0.1) \\ &= -0.05 \end{aligned}$$

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า V จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 10 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 6 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$\begin{aligned} V(S_2) &= V(S_2) + \alpha[R_3 + \gamma V(S_3) - V(S_2)] \\ &= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\ &= 0 + (0.5)(-0.1) \\ &= -0.05 \end{aligned}$$

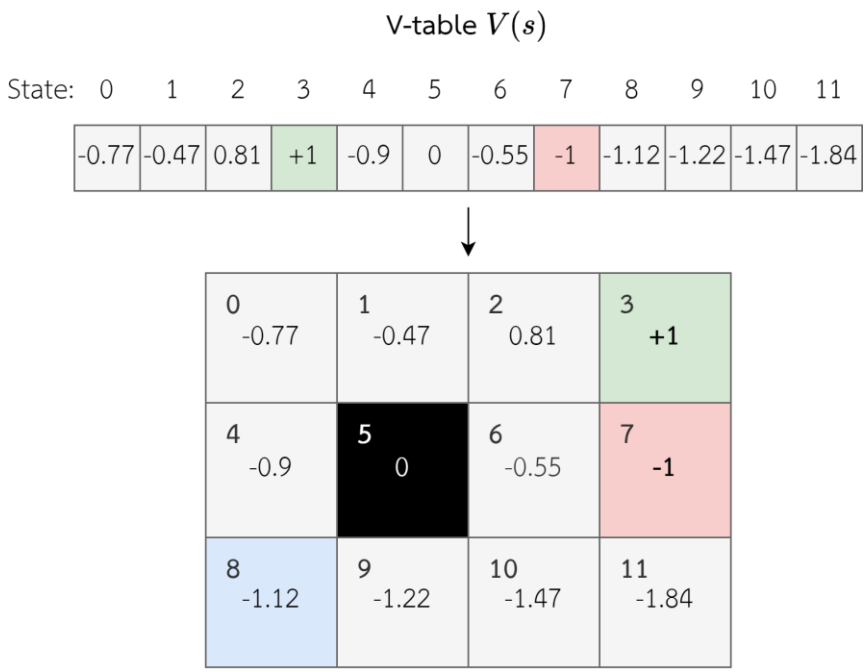
จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า V จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 6 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 2 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$\begin{aligned} V(S_3) &= V(S_3) + \alpha[R_4 + \gamma V(S_4) - V(S_3)] \\ &= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\ &= 0 + (0.5)(-0.1) \\ &= -0.05 \end{aligned}$$

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า V จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 2 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 3 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$\begin{aligned} V(S_4) &= V(S_4) + \alpha[R_5 + \gamma V(S_5) - V(S_4)] \\ &= 0 + (0.5)[1 + (0.9)(1) - 0] \\ &= 0 + (0.5)(1.9) \\ &= 0.95 \end{aligned}$$

ดังนั้นใน Episode ถัด ๆ ไปก็จะมีกระบวนการทำเช่นนี้ในแต่ละ Timestep คือนำค่าประมาณมาจากตาราง V มาใช้คำนวณในแต่ละ Timestep และอัพเดทค่ากลับไป ทำให้วิธีการนี้เกิดการเรียนรู้ในทุก ๆ Timestep จึงสามารถแก้ปัญหาแบบต่อเนื่องได้ และเมื่อเรียนรู้ไปเรื่อย ๆ ตารางค่า V ก็จะลู่เข้าสู่จุดเหมาะสมดังแผนภาพที่ 2.25



แผนภาพที่ 2.25 ตารางค่า V ที่เหมาะสมที่สุดด้วยวิธีการ Temporal Difference

2.6.8 วิธีการ Q-Learning

วิธีการ Q-Learning เป็นวิธีการยอดนิยมซึ่งเป็นวิธีการประเภท Off-policy Temporal Difference Control กล่าวคือไม่ใช้นโยบาย π ในการอัพเดทตารางค่า Q แต่จะใช้ค่า Q สูงสุดของสถานะถัดไป S_{t+1} ในการอัพเดทแทน โดยสมการ State-Action value ที่ใช้อัพเดทค่า Q แสดงดังสมการที่ 2.17

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (2.17)$$

โดย $Q(S_t, A_t)$ คือ รางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ S เมื่อทำการกระทำ A ณ Timestep t

t คือ timestep ณ ปัจจุบัน

γ คือ ปัจจัยส่วนลด

α คือ อัตราการเรียนรู้

R_{t+1} คือ รางวัลที่จะได้รับใน timestep ถัดไป

จากตัวอย่างสภาพแวดล้อมของเกม Grid World เราสามารถนำวิธีการ Q-Learning มาแก้ปัญหานี้ได้ เพื่อให้การแสดงตัวอย่างเป็นไปได้ง่าย จึงขอกำหนดให้ค่า $\gamma = 0.9$ $\alpha = 0.5$ และกำหนดให้ทุก Timestep มีรางวัลเป็น -0.1 เพื่อให้เห็นถึงการคำนวณในแต่ละครั้งและกำหนดให้ตารางค่า Q มีค่าเริ่มต้นดังแผนภาพที่ 2.26

Q-table

$Q(s,a)$

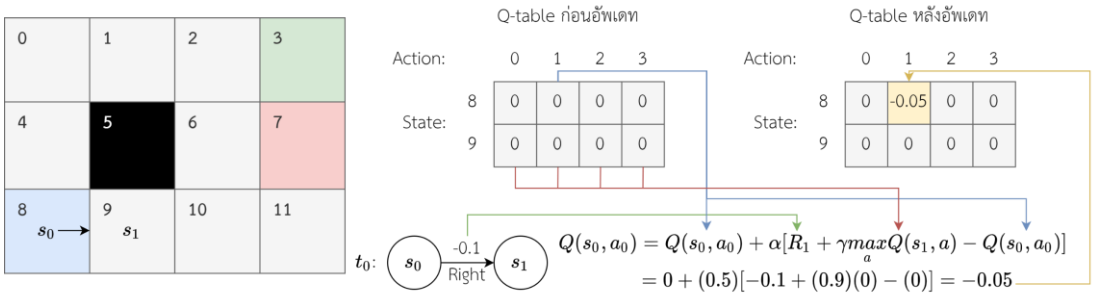
State (s)	Action (a) 0 ↑	1 →	2 ↓	3 ←
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
...	...	...	...	...
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0

แผนภาพที่ 2.26 ตารางค่า Q เริ่มต้นของวิธีการ Q-Learning

สมมุติให้ตัวแทนเริ่มต้นที่ตำแหน่งสถานะที่ 8 และมีการกระทำไปทางขวาทำให้ไปอยู่ในตำแหน่งสถานะที่ 9 เราสามารถคำนวณหาค่ารางวัลที่คาดหวังของสถานะใน Timestep แรกได้จากสมการที่ 2.17 ดังนี้

$$
\begin{aligned}
Q(S_0, A_0) &= Q(S_0, A_0) + \alpha\left[R_1 + \gamma \max_a Q(S_1, a) - Q(S_0, A_0)\right] \\
&= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\
&= 0 + (0.5)(-0.1) \\
&= -0.05
\end{aligned}
$$

จากการคำนวณหาค่ารางวัลที่คาดหวังของสถานะ S_0 เมื่อทำการกระทำ A_0 สามารถแสดงได้ดังแผนภาพที่ 2.27



แผนภาพที่ 2.27 วิธีการคำนวณหาค่ารางวัลที่คาดหวังของสถานะและการกระทำของ timestep แรก

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า Q ของสถานะและการกระทำนั้น ๆ จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 9 มีการกระทำไปทางขวาทำให้ไปอยู่ในตำแหน่งสถานะที่ 10 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$
\begin{aligned}
Q(S_1, A_1) &= Q(S_1, A_1) + \alpha \left[R_2 + \gamma \max_a Q(S_2, a) - Q(S_1, A_1) \right] \\
&= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\
&= 0 + (0.5)(-0.1) \\
&= -0.05
\end{aligned}
$$

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า Q จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 10 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 6 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$
\begin{aligned}
Q(S_2, A_2) &= Q(S_2, A_2) + \alpha \left[R_3 + \gamma \max_a Q(S_3, a) - Q(S_2, A_2) \right] \\
&= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\
&= 0 + (0.5)(-0.1) \\
&= -0.05
\end{aligned}
$$

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า Q จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 6 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 2 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

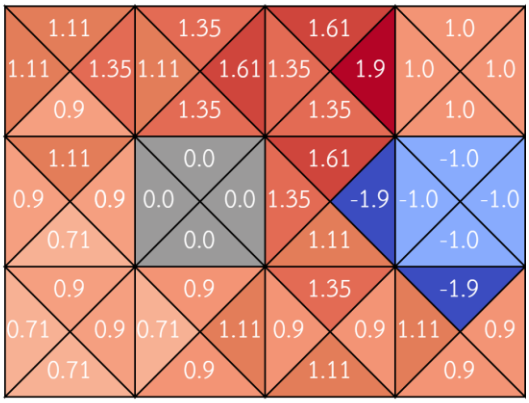
$$
\begin{aligned}
Q(S_3, A_3) &= Q(S_3, A_3) + \alpha \left[R_4 + \gamma \max_a Q(S_4, a) - Q(S_3, A_3) \right] \\
&= 0 + (0.5)[-0.1 + (0.9)(0) - 0] \\
&= 0 + (0.5)(-0.1) \\
&= -0.05
\end{aligned}
$$

จากนั้นนำค่าที่ได้ไปอัพเดทตารางค่า Q จากนั้นสมมุติให้ตัวแทนที่อยู่ในตำแหน่งสถานะที่ 2 มีการกระทำไปทางบนทำให้ไปอยู่ในตำแหน่งสถานะที่ 3 สามารถคำนวณคำนวณหาค่ารางวัลที่คาดหวังของสถานะ ใน Timestep ที่สองได้ดังนี้

$$
\begin{aligned}
Q(S_4, A_4) &= Q(S_4, A_4) + \alpha \left[R_5 + \gamma \max_a Q(S_5, a) - Q(S_4, A_4) \right] \\
&= 0 + (0.5)[1 + (0.9)(1) - 0] \\
&= 0 + (0.5)(1.9) \\
&= 0.95
\end{aligned}
$$

ดังนั้นใน Episode ถัด ๆ ไปก็จะมีกระบวนการทำเช่นนี้ในแต่ละ Timestep คือนำค่าประมาณมาจากตาราง Q มาใช้คำนวณในแต่ละ Timestep และอัพเดทค่ากลับไป ทำให้วิธีการนี้เกิดการเรียนรู้ในทุก ๆ Timestep จึงสามารถแก้ปัญหาแบบต่อเนื่องได้ และเมื่อเรียนรู้ไปเรื่อย ๆ ตารางค่า Q ก็จะลู่เข้าสู่จุดเหมาะสมดังแผนภาพที่ 2.28

Q-table	0 ↑	1 →	2 ↓	3 ←
0	1.11	1.35	0.9	1.11
1	1.35	1.61	1.35	1.11
2	1.61	1.9	1.35	1.35
3	1	1	1	1
...	...	...	...	...
8	0.9	0.9	0.71	0.71
9	0.9	1.11	0.9	0.71
10	1.35	0.9	1.11	0.9
11	-1.9	0.9	0.9	1.11



แผนภาพที่ 2.28 ตารางค่า Q ที่เหมาะสมที่สุดด้วยวิธีการ Q-Learning

2.6.9 นโยบายละโมบด้วยค่า Epsilon (Epsilon Greedy)

จากวิธีการที่กล่าวไปในข้อ 2.6.5 - 2.6.8 จะทำงานไม่ได้เลยหากขาดส่วนนโยบายที่ใช้สำหรับการเลือกการทำ เพราะที่กล่าวมาขั้นตอนจะเป็นกระบวนการเรียนรู้เพื่อประเมินคุณภาพของค่าสถานะ $V(S)$ หรือคุณภาพของค่า State-Action $Q(s,a)$ โดยใช้เพียงนโยบายแบบสุ่มเท่านั้น (Random Policy) ซึ่งในความเป็นจริงไม่ควรทำการสุ่มตลอดเวลา แต่ควรสุ่มในช่วงแรก (Exploration) เพราะในช่วงแรกตารางจะยังไม่ดีมากพอ แต่พอผ่านไปในช่วงเวลาหนึ่งเมื่อตารางมีการเรียนรู้ มีการปรับปรุงมากพอแล้วควรใช้ความรู้ที่มีจากตารางมากกว่าการสุ่ม (Exploitation) โดยเรียกปัญหานี้ว่า Exploration-Exploitation Dilemma

ดังนั้นจึงมีคนคิดค้นนโยบายในการสมดุลกันระหว่างการสำรวจและการใช้ความรู้ที่มี ซึ่งนโยบายนี้เรียกว่านโยบายละโมบด้วยค่า Epsilon ซึ่งเป็นนโยบายที่นิยมใช้กันเป็นจำนวนมากโดยกระบวนการของนโยบายละโมบด้วยค่า Epsilon มีการทำงานดังสมการที่ 2.18

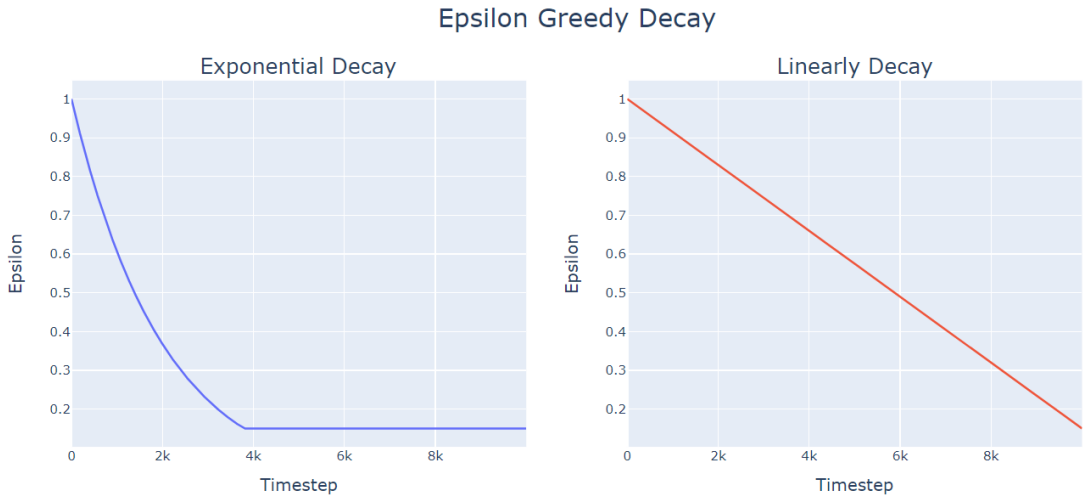
$$\pi(a|s) \leftarrow \begin{cases} 1-\epsilon & \text{if } a = \underset{a \in A(s)}{\operatorname{argmax}}\, Q(s,a) \\ \epsilon & \text{otherwise} \end{cases} \tag{2.18}$$

จากสมการที่ 2.18 สามารถตีความได้ดังนี้ จะเลือกการกระทำละโมบด้วยความน่าจะเป็น $(1-\epsilon)$ และความน่าจะเป็น ϵ จะเป็นการสุ่มการกระทำเพื่อสำรวจขอบเขตการกระทำ (Action Space) ที่เป็นไปได้ทั้งหมด โดยค่า ϵ จะอยู่ระหว่าง 0 ถึง 1 เพื่อเป็นการถ่วงน้ำหนักระหว่างการใช้ความรู้ที่มี (Exploitation) และการสำรวจ (Exploration)

ดังนั้นค่า Epsilon ในช่วงแรกจะสูงเพื่อที่จะสำรวจความเป็นไปได้ทั้งหมดให้ได้มากที่สุดเพราะยังไม่มีความรู้มากพอ ซึ่งหลังจากเกิดการเรียนรู้ค่า Epsilon จะลดลงตามช่วงเวลาจนถึงค่าต่ำสุดของ ϵ_{min} โดยส่วนมากการกำหนดค่าต่ำสุดของ Epsilon นั้นจะกำหนดไว้ที่ 0.15 หรือ 15 เปอร์เซ็นต์เพราะในบางครั้งถึงแม้เราจะเชื่อว่าเรามีข้อมูลที่มากพอแล้ว หรือมีตารางที่ดีแล้วแต่ใน

ความเป็นจริงอาจจะไม่ได้ครอบคลุมทุกเหตุการณ์ที่เป็นไปได้ทั้งหมด ดังนั้นจึงตั้งให้มีโอกาสในการสุ่มไว้เพียงเล็กน้อยในแต่ละเหตุการณ์นั้นเอง

จากตัวอย่างสภาพแวดล้อมของเกม Grid World ที่ใช้วิธีการ Q-Learning ในการแก้ปัญหาในหัวข้อที่ 2.6.8 นั้นเราสามารถนำนโยบายละโมบด้วยค่า Epsilon มาใช้งานได้ ซึ่งในช่วงเวลาแรกของการเรียนรู้เราควรตั้งค่า Epsilon ไว้ที่ 1 หรือ 100 เปอร์เซ็นต์ของการสุ่ม จากนั้นกระบวนการเรียนรู้ก็จะเป็นการสุ่มการกระทำ คำนวณหาค่ารางวัลที่คาดหวังที่ถูกลดในแต่ละ Timestep และอัพเดทค่านั้นกลับไปยังตาราง Q ซึ่งกระบวนการเช่นนี้ก็จะวนทำไปเรื่อย ๆ ดังนั้นทุก ๆ Timestep ตาราง Q ก็จะมีการปรับปรุงให้ดียิ่งขึ้นเรื่อย ๆ จากการกระทำที่เราได้ทำการสุ่มมา จากนั้นเราต้องทำการลดค่า Epsilon ลงเพื่อลดปริมาณการสำรวจซึ่งวิธีการลดค่า Epsilon นั้นสามารถลดได้หลากหลายวิธีขึ้นอยู่กับความเหมาะสมของแต่ละโจทย์ปัญหา แต่หลัก ๆ สามารถจำแนกได้ดังนี้ 1. Linear decay เป็นการลดแบบคงที่ และ 2. Exponential Decay เป็นการลดแบบเท่าตัว ซึ่งวิธีการลดแบบ Exponential Decay นิยมใช้กันมากที่สุด ดังแผนภาพที่ 2.29



แผนภาพที่ 2.29 อัตราการลดของค่า Epsilon ของวิธีการ Linear และ Exponential

2.7 การเรียนรู้แบบเสริมแรงเชิงลึก (Deep Reinforcement Learning)

ในหัวข้อ 2.6 ก่อนหน้านี้จะเป็นกล่าวถึงการเรียนรู้แบบเสริมแรงดั้งเดิม จะพบว่าทุกวิธีการจะมีตารางในการเก็บค่า ซึ่งนำไปสู่ข้อเสียคือปัญหาที่ต้องการจะแก้นั้นจะต้องมีจำนวนการกระทำและสถานะที่น้อย เพราะขนาดของตารางจะขึ้นอยู่กับจำนวนการกระทำและสถานะที่เป็นไปได้ทั้งหมด ซึ่งในชีวิตจริงขอบเขตของสถานะมีขนาดใหญ่ ทำให้เป็นไปไม่ได้เลยที่จะไปอยู่ในทุกสถานะเพื่อหา State-Action value ทั้งหมดออกมา และยิ่งกว่านั้นคือหากตารางมีขนาดใหญ่มากเท่าไร ก็จะมีปัญหาเรื่องหน่วยความจำของคอมพิวเตอร์ที่ใช้ในการประมวลผลมากเท่านั้น

เพื่อแก้ปัญหาข้อจำกัดเหล่านี้จึงนำโครงข่ายประสาทเชิงลึก (Deep Neural Network) มาเป็นฟังก์ชันประมาณการแทนการใช้ตาราง ด้วยความสามารถในการประมาณฟังก์ชันไม่เป็นเชิงเส้น

และความสามารถในการหาปัจจัยที่เกี่ยวข้องจากข้อมูลดิบทำให้สามารถประมาณค่าได้ถึงแม้สถานะนั้นจะไม่เคยมาก่อนก็ตาม

2.7.1 วิธีการเรียนรู้แบบเสริมแรงเชิงลึก

วิธีการเรียนรู้แบบเสริมแรงเชิงลึกสามารถแบ่งได้เป็น 3 ประเภทหลัก ๆ ได้แก่

2.7.1.1 วิธีการแบบ Value

วิธีการ Value-based ถูกสร้างขึ้นมาจากวิธีการ Temporal Difference ที่กล่าวถึงในหัวข้อ 2.6.7 โดยหลักการคือแทนที่ตารางค่า V หรือ Q เป็นการประมาณโดยใช้โครงข่ายประสาทเทียมแทน กล่าวคือใช้โครงข่ายประสาทเทียมในการเรียนรู้ฟังก์ชัน Value นั้นเอง และจะใช้นโยบายจำพวกละโมบด้วยค่า Epsilon ในการเลือกการกระทำจากค่าเหล่านั้นอีกทีหนึ่ง ซึ่งการเรียนรู้แบบนี้ถูกใช้ใน Q-learning เป็นต้น

2.7.1.2 วิธีการแบบ Policy

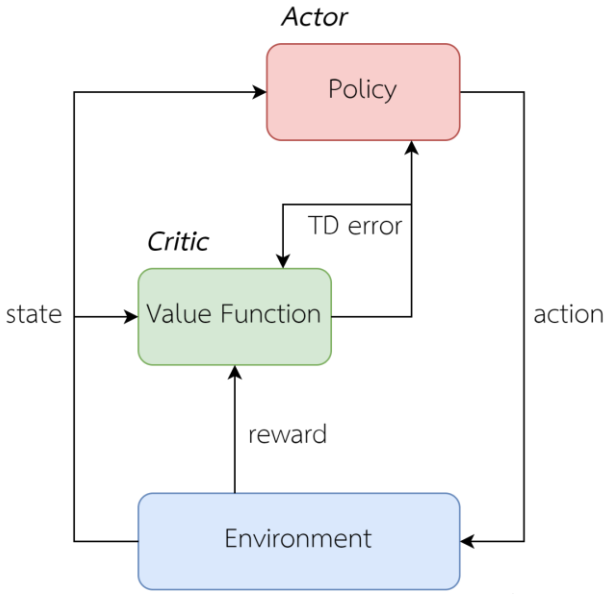
วิธีการ Policy-based จะเป็นการเรียนรู้ฟังก์ชัน Policy $\pi(a|s,\theta)$ โดยตรงแทนการเรียนรู้จากฟังก์ชัน Value และเลือกการกระทำที่ดีที่สุด คุณภาพของนโยบายจะถูกวัดประสิทธิภาพจากตัววัด $J(\theta)$ โดยฟังก์ชันเป้าหมาย (Objective Function) ของวิธีการนี้คำนวณได้ดังสมการที่ 2.19 คือการเพิ่มค่าของ $J(\theta)$ ให้สูงที่สุด

$$\theta^* = \underset{\theta}{\operatorname{argmax}} J(\theta) \tag{2.19}$$

โดย θ คือ ค่าพารามิเตอร์ที่เราต้องการเรียนรู้
$J(\theta)$ คือ ฟังก์ชันเป้าหมายที่เราใช้ในการเรียนรู้ ขึ้นอยู่แต่ละอัลกอริทึม
θ^* คือ ค่าพารามิเตอร์ที่ดีที่สุดที่ทำให้ฟังก์ชันเป้าหมายมีค่าสูงสุด

2.7.1.3 วิธีการแบบ Actor-Critic

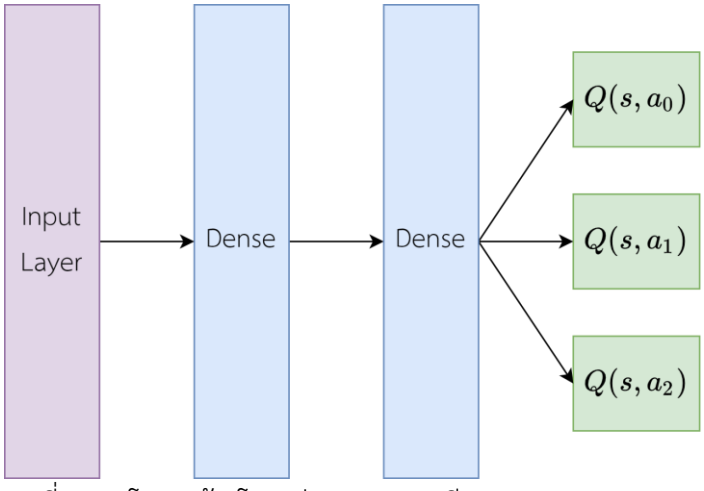
วิธีการแบบ Actor-Critic เป็นวิธีการที่นำวิธี Value-based และ Policy-based มาปรับใช้ร่วมกัน ดังนั้นต้องมีเพียงหนึ่งวิธีการในการเลือกการกระทำและอีกหนึ่งวิธีในการประเมินการกระทำที่เลือกมาว่าดีมากน้อยเพียงใด ซึ่งในที่นี่จะใช้วิธี Policy-based เป็นตัวเลือกการกระทำจึงเรียกตำแหน่งนี้ว่า Actor โดยจะใช้นโยบายปัจจุบันเพื่อเลือกการกระทำ a ณ สถานะ s และเมื่อทำการกระทำนั้นกับสภาพแวดล้อมและได้รางวัลกลับมาจะนำส่วนของรางวัลและการกระทำที่ทำไปประเมินว่าดีมากน้อยเพียงใด ในส่วนนี้จะเป็นหน้าที่ของวิธี Value-based จึงเรียกตำแหน่งนี้ว่า Critic โดยจะใช้ฟังก์ชันค่า $v(s)$ และคำนวณค่ารางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะที่ได้รับมาและคำนวณออกมาเป็นค่าสูญเสียหรือ TD-error ดังที่กล่าวไปในหัวข้อ 2.6.7 จากนั้นจะนำค่า TD-error ไปอัพเดททั้ง 2 โครงข่ายประสาทเทียมทั้ง Actor และ Critic ดังแผนภาพที่ 2.30



แผนภาพที่ 2.30 สถาปัตยกรรม Actor-Critic โดย Actor ทำหน้าที่เป็นนโยบายและแปลงค่าจากสถานะที่รับเข้ามาไปเป็นการกระทำ ในส่วนของ Critic ทำหน้าที่เป็นฟังก์ชันค่า โดยทั้งสองเครือข่ายสามารถอัพเดทผ่านค่า TD-error ได้เหมือนกัน ดังนั้น Actor จะใช้ Critic ในขั้นตอนการเรียนรู้

2.7.2 Deep Q-Learning (DQN)

จากที่กล่าวมาวิธีการ Deep Q-Learning เป็นวิธีการประเภท Temporal Difference Control ที่สามารถเรียนรู้เพื่อแก้ปัญหาที่เป็นแบบต่อเนื่องได้และมีการใช้ตารางค่า Q ในการเก็บรางวัลที่คาดหวังสะสมที่ถูกลดทอนของแต่ละคู่สถานะและการกระทำที่เป็นไปได้ทั้งหมด ด้วยเหตุนี้ทำให้วิธีการ Q-Learning มีข้อจำกัดเพราะเราไม่สามารถสร้างตารางที่เก็บสถานะหรือการกระทำที่มีจำนวนมากเกินไปได้ ดังนั้นเราจึงนำโครงข่ายประสาทเทียมมาใช้ในการประมาณค่าของตารางเหล่านั้นแทน ซึ่งวิธีการ Deep Q-Learning ถูกคิดค้นต่อยอดมาจากวิธีการ Q-Learning เพื่อแก้ปัญหาเหล่านี้ [6] โดยวิธีการ Deep Q-Learning ถือเป็นหนึ่งในวิธีการประเภท Value-based ดังแผนภาพที่ 2.31



แผนภาพที่ 2.31 โครงสร้างโครงข่ายประสาทเทียมแบบ Deep Q-Network

ซึ่งผู้ที่เสนอวิธีการ Deep Q-Learning ได้บอกถึงปัญหาอยู่ 2 ประการของการนำโครงข่ายประสาทเทียมมาใช้ดังนี้

1. ข้อมูลที่ไม่อิสระต่อกันและมีการแจกแจงที่ไม่เหมือนกัน (Independent and Identically Distributed) ซึ่งปัญหานี้เกิดจากที่ข้อมูลที่เราใช้ในการเรียนรู้นั้นเกิดจากนโยบายตัวเดียวกันอีกทั้งยังมีความเกี่ยวข้องกันระหว่างชุดข้อมูลเพราะข้อมูลเกิดเป็นลำดับ สามารถแก้ไขได้โดยการสร้างถังเก็บข้อมูลประสบการณ์จากการเล่น และใช้การสุ่มแบบเท่าเทียมกัน (Uniform Sampling) เพื่อนำประสบการณ์เหล่านั้นมาเรียนรู้

2. เป้าหมายในการเรียนรู้มีความไม่คงที่ (Non-stationary target) จากที่กล่าวมาในส่วนแรกคือเราไม่รู้ค่า TD-target ที่แท้จริง ดังนั้นเราจึงใช้การประมาณแทนทำให้เมื่อเรามีการอัพเดทโครงข่ายประสาทเทียมค่า TD-target ที่เราประมาณเหล่านี้ก็จะขยับไปด้วย ซึ่งเป้าหมายของเราคือเราต้องการให้ค่าประมาณของเราเข้าใกล้ TD-target มากที่สุด ดังนั้นวิธีแก้ปัญหาคือการใช้โครงข่ายประสาทเทียม 2 โครงข่าย หนึ่งตัวสำหรับการเรียนรู้ (Online Network) และอีกหนึ่งตัวสำหรับการประเมินผล (Target Network) โดยค่าน้ำหนักของโครงข่ายประสาทเทียมสำหรับการประเมินผลจะถูกคัดลอกมาจากน้ำหนักของโครงข่ายประสาทเทียมสำหรับการเรียนรู้ทุก ๆ N รอบขึ้นอยู่กับความเหมาะสมเพื่อไม่ให้โครงข่ายประสาทเทียมสำหรับการประเมินไม่เก่าเกินไป

โดยค่า TD-target ที่มีการใช้โครงข่ายประสาทเทียมมาช่วยประมาณการสามารถเขียนได้ดังสมการที่ 2.20

$$Y_k^{DQN} = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k^-) \tag{2.20}$$

โดย $Q(s', a'; \theta_k^-)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดไปของทุก ๆ การกระทำที่เป็นไปได้โดยใช้โครงข่ายประสาทเทียมสำหรับการประเมิน

r คือ รางวัล ณ ขณะนั้น

γ คือ ปัจจัยส่วนลด

θ_k^- คือ ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมสำหรับการประเมิน

ดังนั้นเราสามารถคำนวณหาค่าสูญเสียสำหรับการเรียนรู้ของโครงข่ายของประสาทเทียมได้แล้ว จากสมการที่ 2.17 เดิมของ Q-Learning สามารถนำมาเขียนใหม่ได้ดังสมการที่ 2.21 โดยเราสามารถใช้ได้ทั้ง L1 Loss (MAE) หรือ L2 Loss (MSE) หรือ Huber Loss (L1+L2) ในการคำนวณปริมาณค่าสูญเสีย

$$loss = \left(r + \gamma \max_{a' \in A} Q(s', a'; \theta_k^-) - Q(s, a; \theta_k)\right)^2 \tag{2.21}$$

โดย $Q(s, a; \theta_k)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะปัจจุบันและการกระทำที่เลือกโดยใช้โครงข่ายประสาทเทียมสำหรับการเรียนรู้

$Q(s', a'; \theta_k^-)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดไปของทุก ๆ การกระทำที่เป็นไปได้โดยใช้โครงข่ายประสาทเทียมสำหรับการประเมิน

r คือ รางวัล ณ ขณะนั้น

γ คือ ปัจจัยส่วนลด

θ_k คือ ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมสำหรับการเรียนรู้

θ_k^- คือ ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมสำหรับการประเมิน

2.7.3 Double Deep Q-Learning (DDQN)

Double Deep Q-Learning เป็นอัลกอริทึมที่ได้รับความนิยมและอีกทั้งยังเป็นอัลกอริทึมที่มีการพัฒนาต่อยอดมาจาก Deep Q-Learning ที่แก้ปัญหาการ Overestimate [7] จากการเลือก Action ที่มากที่สุดจากทุก ๆ Action เนื่องจากค่า Q-values นั้นมีความแปรปรวนมากดังสมการที่ 2.20 ดังนั้นเพื่อแก้ปัญหานี้จึงใช้โครงข่ายประสาทเทียมในการเรียนรู้ในการทำนายค่า Q-value ปัจจุบันและเลือก Action ที่ดีที่สุด จากนั้นใช้โครงข่ายประสาทเทียมในการประเมินในการประเมิน Action นั้นใน State ถัดไปแทนดังสมการที่ 2.22 ดังนั้นผู้พัฒนาจึงคิดว่าอัลกอริทึม Double Deep Q-Learning จะมีความสามารถในการเรียนรู้ได้รวดเร็วกว่า

$$Y_k^{DDQN} = r + \gamma Q\left(s', \operatorname*{argmax}_{a \in A} Q(s', a; \theta_k); \theta_k^-\right) \tag{2.22}$$

โดย $\operatorname*{argmax}_{a \in A} Q(s', a; \theta_k)$ คือ การกระทำในสถานะถัดไปที่ให้ค่าประมาณ Q ของโครงข่ายประสาทเทียมสำหรับการเรียนรู้สูงที่สุด

$Q\left(s', \underset{a \in A}{\operatorname{argmax}}\, Q(s', a; \theta_k); \theta_k^-\right)$ คือ ค่าประมาณ Q ของโครงข่ายประสาทเทียม โดยใช้สถานะถัดไปและการกระทำที่ให้ค่าประมาณ Q ของโครงข่ายประสาทเทียมสำหรับการเรียนรู้สูงที่สุดมาประมาณ

r คือ รางวัล ณ ขณะนั้น

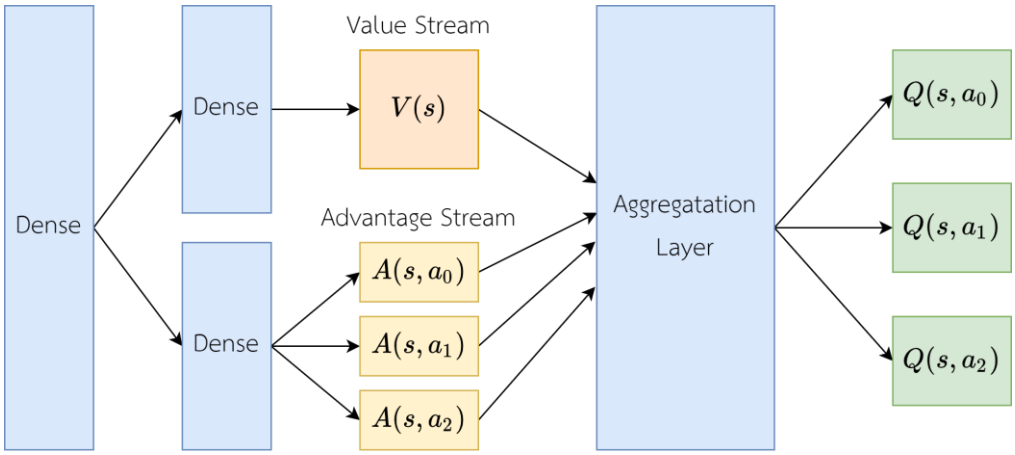
γ คือ ปัจจัยส่วนลด

θ_k คือ ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมสำหรับการเรียนรู้

θ_k^- คือ ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมสำหรับการประเมิน

2.7.4 Dueling Double Deep Q-Learning (Dueling DDQN)

Dueling Double Deep Q-Learning เป็นอัลกอริทึมที่ได้รับความนิยมและอีกทั้งยังเป็นอัลกอริทึมที่มีการพัฒนาต่อยอดมาจาก Deep Q-Learning ที่นำมาแก้ปัญหา State ที่บางครั้งไม่ว่าจะเป็น Action ไหนก็ให้ผลลัพธ์ที่เหมือน ๆ กัน ซึ่งส่งผลให้การปรับ Q-value ของ State นั้นให้ครบทุก Action ทำได้ช้ากว่า [8] เพื่อที่จะแก้ปัญหานี้จึงมีการปรับแก้ไขโครงสร้างของโครงข่ายประสาทเทียมให้แยกเป็น 2 สายแทน โดยสายหนึ่งจะทำหน้าที่ทำนายค่าของ State $v(s)$ ว่าการไปอยู่ State นั้นดีมากน้อยแค่ไหน และอีกสายหนึ่งจะทำหน้าที่ทำนายค่าความได้เปรียบของแต่ละ Action หรือค่า Advantage ว่า Action ไหนที่ได้เปรียบกว่าหรือเสียเปรียบกว่า Action อื่น แล้วจึงนำผลลัพธ์ของทั้งสองสายมารวมกันอีกทีเป็นค่า Q-value ดังแผนภาพที่ 2.32 โดยสามารถเขียนอยู่ในรูปสมการได้ดังสมการที่ 2.23 ซึ่งผู้พัฒนาคิดว่าเราสามารถนำข้อดีของ Double Deep Q-Learning ที่แก้ปัญหา Overestimate มาใช้ร่วมกันได้



แผนภาพที่ 2.32 โครงสร้างโครงข่ายประสาทเทียมแบบ Dueling Deep Q-Network

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|}\sum_{a'} A(s,a')\right) \tag{2.23}$$

โดย $Q(s,a)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะปัจจุบันและการกระทำที่เลือก

$V(s)$ คือ ค่ารางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะ s

$A(s,a)$ คือ ค่าประมาณความได้เปรียบของการกระทำที่เลือกสถานะ s

$\frac{1}{|A|}\sum_{a'} A(s,a')$ คือ ค่าเฉลี่ยของค่าประมาณความได้เปรียบของการกระทำทั้งหมด

2.7.5 Dueling Double Deep Q-Learning with Prioritized Experience Replay (Dueling DDQN + PER)

จากเดิมอัลกอริทึมที่ 2.7.2 - 2.7.4 จะมีการใช้ Experience Replay ในการเก็บประสบการณ์ และใช้การสุ่มแบบ Uniform จากประสบการณ์เหล่านั้นมาเรียนรู้ ทำให้การเรียนรู้นั้นช้าเพราะบางครั้งตัวอย่างประสบการณ์เหล่านั้นอาจจะคงที่หรือ Stable อยู่แล้ว ดังนั้นคงจะดีกว่าถ้าเราจัดความสำคัญให้กับประสบการณ์ที่มักจะทำผิดพลาด (Error มากกว่า) มากกว่าประสบการณ์ที่ทำได้ดีอยู่แล้ว (Error น้อยกว่า) ก็คือเพิ่มความน่าจะเป็นที่จะสุ่มประสบการณ์ที่มักจะทำผิดพลาดมากกว่าประสบการณ์ที่ทำได้ดีอยู่แล้ว โดยเราจะเรียกถังเก็บข้อมูลประสบการณ์แบบนี้ว่าถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ ซึ่งวิธีการนี้จะทำให้อัลกอริทึมของเราสามารถเรียนรู้ได้เร็วมากขึ้น

2.7.6 ถังเก็บข้อมูลประสบการณ์

จากปัญหาข้อมูลที่ไม่อิสระต่อกันและมีการแจกแจงที่เหมือนกันที่กล่าวไปในวิธีการ Deep Q-Learning จึงจำเป็นต้องมีถังสำหรับเก็บรวบรวมประสบการณ์เหล่านี้ เพื่อนำมาสุ่มหยิบประสบการณ์เหล่านี้เพื่อแก้ปัญหาในข้างต้น โดยเราสามารถพัฒนาถังเก็บประสบการณ์ได้ 2 แบบหลัก ๆ ได้แก่

2.7.6.1 ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay)

ส่วนของการถังเก็บข้อมูลประสบการณ์ทั่วไปจะใช้โครงสร้างข้อมูลประเภท Deque เนื่องจากประสบการณ์ที่ถูกเพิ่มในส่วนแรกจะเป็นประสบการณ์ที่เก่าที่สุด ดังนั้นเมื่อเรามีการสะสมประสบการณ์เต็ม Deque และข้อมูลในส่วนแรกจะถูกลบออกไป ดังนั้นประสบการณ์ที่เหลือก็จะเป็นประสบการณ์ที่ใหม่กว่า โดยประสบการณ์จะถูกเก็บในรูปแบบของ Tuple โดยมีลำดับดังนี้ (s,a,r,d,s') กล่าวคือเก็บสถานะปัจจุบัน การกระทำปัจจุบัน รางวัลที่ได้จากการกระทำปัจจุบันในสถานะปัจจุบัน สถานะว่าข้อมูลสิ้นสุดหรือยัง และสถานะถัดไปหลังจากทำการกระทำปัจจุบันในสถานะปัจจุบันตามลำดับ

2.7.6.2 ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay)

ส่วนของถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญจะมีการจัด Priorities หรือความสำคัญของประสบการณ์ในการเรียนรู้แทนการสุ่มด้วยความน่าจะเป็นเท่า ๆ กัน (Uniform) เพื่อเพิ่มความเร็วในการเรียนรู้ของอัลกอริทึมให้รวดเร็วยิ่งขึ้น [9]

โดยที่ความน่าจะเป็นของการสุ่มแต่ละ sample นั้นจะอยู่ในช่วงของ [0, 1] ซึ่งสามารถหาได้จากสมการที่ 2.24

$$P(t_i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{2.24}$$

โดย $P(t_i)$ คือ ความน่าจะเป็นของการสุ่มแต่ละ sample

α คือ ตัวกำหนดความเชื่อมั่นต่อประสบการณ์

p_i คือ ค่าความสำคัญ (Priority) ของแต่ละประสบการณ์

โดยพารามิเตอร์ Alpha (α) จะมีค่าอยู่ในช่วง [0, 1] ซึ่งจะเป็นตัวกำหนดความเชื่อมั่นต่อประสบการณ์นั้น ๆ ว่าเชื่อถือได้มากน้อยแค่ไหน ถ้าเชื่อถือได้มากค่า Priority นั้นที่ถูกยกกำลังก็จะมาก แต่ถ้า Alpha มีค่าเป็น 0 เท่ากับว่าทุกตัวมีค่าเท่ากันหมด ค่า Priority ก็จะถูกตัดทิ้ง ดังนั้นโดยเริ่มต้นเราจะให้ค่า Alpha มากแล้วค่อย ๆ ลดลงจนถึง 0

ส่วนค่า Priority นั้นสามารถหาได้จากสมการที่ 2.25

$$\begin{aligned} \delta_i &= Q_\pi(s,a) - [r + \gamma Q_\pi(s',a')] \\ p_i &= |\delta_i| + \epsilon \end{aligned} \tag{2.25}$$

โดย δ_i คือ ปริมาณของ TD-error

$Q_\pi(s,a)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะปัจจุบันและการกระทำที่เลือกโดยนโยบาย π

$Q_\pi(s',a')$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดไปและการกระทำที่ทำในสถานะถัดไปที่เลือกโดยนโยบาย π

r คือ รางวัล ณ ขณะนั้น

γ คือ ปัจจัยส่วนลด

p_i คือ ค่าความสำคัญ (Priority) ของแต่ละประสบการณ์

ϵ คือ ค่าคงที่ที่กำหนด

จากสมการที่ 2.25 สามารถแปลความได้ดังนี้หาก TD-error มากแสดงว่าสามารถเรียนรู้ได้มาก ดังนั้นค่า Priority ก็จะสูงตาม และที่ใส่ Absolute เพื่อป้องกันไม่ให้ค่าติดลบ

เพราะเราสนใจที่ปริมาณและที่มีการบวกด้วยค่า Epsilon เล็กน้อยเพื่อไม่ให้มีปัญหาตัวหารเป็น 0 เวลานำไปคำนวณความน่าจะเป็น

การที่เรามีการจัด Priority ของประสบการณ์ทำให้ Distribution ของ State จริง ๆ เปลี่ยนไปจึงเกิด Bias ขึ้นดังนั้นเพื่อจัดการกับปัญหาเหล่านี้จึงมีค่า Weight เข้ามาคูณกับค่าสูญเสียเพื่อแก้ปัญหา Bias ในส่วนนี้ โดยค่า Weight หาได้จากสมการที่ 2.26

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta} \tag{2.26}$$

โดย w_i คือ ค่าน้ำหนักของแต่ละ Sample

N คือ ปริมาณประสบการณ์ทั้งหมดใน Buffer

$P(i)$ คือ ความน่าจะเป็นของการสุ่มแต่ละ Sample

β คือ ค่าควบคุมน้ำหนักของ Sample

โดยพารามิเตอร์ N แทนจำนวนประสบการณ์ทั้งหมดใน Buffer และพารามิเตอร์ Beta (β) จะเป็นตัวกำหนดปริมาณในการปรับ Bias ซึ่งจะมีค่าอยู่ในช่วง [0, 1] ดังนั้นค่าสูญเสียจะหาได้จากสมการที่ 2.27

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[w_i \cdot \left(r_i + \gamma \max_{a_i' \in A} Q(s_i', a_i'; \theta_k^-) - Q(s_i, a_i; \theta_k)\right)^2\right] \tag{2.27}$$

โดย L คือ ค่าสูญเสียที่โครงข่ายประสาทเทียมจะนำไปเรียนรู้

N คือ ปริมาณ Sample ทั้งหมดของ Training Batch

w_i คือ ค่าน้ำหนักของแต่ละ Sample

r_i คือ รางวัลที่ได้รับของแต่ละ Sample

γ คือ ปัจจัยส่วนลด

$Q(s_i, a_i; \theta_k)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะปัจจุบันและการกระทำที่เลือกโดยใช้โครงข่ายประสาทเทียมสำหรับการเรียนรู้ของแต่ละ Sample

$Q(s_i', a_i'; \theta_k^-)$ คือ ค่าประมาณรางวัลที่คาดหวังสะสมที่ถูกลดทอนของสถานะถัดของทุก ๆ การกระทำที่เป็นไปได้โดยใช้โครงข่ายประสาทเทียมสำหรับการประเมินของแต่ละ Sample

2.8 งานวิจัยที่เกี่ยวข้อง

Yang Hongyang, Liu Xiao-Yang, Zhong Shan, Walid Anwar, 2020 ได้ทำการศึกษาค้นคว้าบทความงานวิจัยเกี่ยวกับการซื้อขายหุ้นอัตโนมัติด้วยวิธีการเรียนรู้แบบเสริมแรงเชิงลึก [10] โดยงานศึกษาวิจัยครั้งนี้มีการนำหลักการกลยุทธทั้งมวล (Ensemble) เข้ามาปรับใช้ ข้อมูลที่ใช้ในงานวิจัยนี้เป็นข้อมูลหุ้น 30 หุ้นของดาวโจนส์ตั้งแต่วันที่ 01/01/2009 ถึง 05/08/2020 โดยงานวิจัยนี่ได้ออกแบบสภาพแวดล้อมสำหรับตัวแทนที่จะฝึกสอนออกเป็น 2 ส่วนหลักคือขอบเขตสถานะ (State Space) และขอบเขตการกระทำ (Action Space) ในส่วนของขอบเขตสถานะประกอบไปด้วย 4 ส่วนหลักคือ 1. ยอดเงินคงเหลือในบัญชี 2. ราคาปิดของแต่ละหุ้น 3. จำนวนหุ้นที่ถือ และ 4. ตัวชี้วัดเชิงเทคนิค ในส่วนของขอบเขตการกระทำได้ออกแบบให้ค่าเริ่มตั้งแต่ -k ถึง k โดย k บ่งบอกถึงปริมาณที่สามารถซื้อหรือขายได้ของหุ้นตัวนั้น ในส่วนของตัวแทน (Agent) งานวิจัยนี้ได้นำกลยุทธทั้งมวลมาคัดเลือกอัลกอริทึมที่ดีที่สุดจากสามอัลกอริทึมในสถานการณ์ที่แตกต่างกัน โดยตัวที่ใช้เป็นเกณฑ์ในการคัดเลือกอัลกอริทึมของงานวิจัยนี้ได้ใช้อัตราส่วนชาร์ป (Sharpe Ratio) เป็นตัวคัดเลือกอัลกอริทึมเนื่องจากอัตราส่วนชาร์ปมีความสมดุลระหว่างผลตอบแทนและความเสี่ยง อีกทั้งยังมีการนำตัวชี้วัด Turbulence เข้ามาวัดสถานการณ์ของตลาดเพื่อให้ตัวแทนสามารถหลีกเลี่ยงความเสี่ยงจากสถานการณ์ของตลาดที่ไม่ปกติได้ โดยงานวิจัยนี้ได้คัดเลือกอัลกอริทึมที่ใช้มาสามอัลกอริทึมคือ 1. Advantage Actor Critic (A2C) 2. Deep Deterministic Policy Gradient (DDPG) และ 3. Proximal Policy Optimization (PPO) ซึ่งผลลัพธ์ที่ได้จากงานวิจัยนี้พบว่าวิธีการทั้งมวลช่วยให้ดึงจุดแข็งของแต่ละอัลกอริทึมออกมาได้ดีในสถานการณ์ต่าง ๆ โดยอัลกอริทึม A2C ทำงานได้ดีเมื่อราคาเป็นแนวโน้มขาลง อัลกอริทึม PPO ทำงานได้ดีเมื่อราคาเป็นแนวโน้มขาขึ้น และอัลกอริทึม DDPG ทำงานได้ดีเมื่อราคาที่ไม่มีทิศทางที่แน่นอน และตัวแทนสามารถตัดขาดทุนได้ในกรณีที่สถานการณ์ของตลาดไม่ปกติจากการนำตัวชี้วัด Turbulence เข้ามา ซึ่งผลลัพธ์จากการทดลองพบว่าวิธีกลยุทธทั้งหมดให้ค่าอัตราส่วนชาร์ปได้ดีที่สุด เมื่อเทียบกับการใช้แต่ละอัลกอริทึมเพียงตัวเดียวหรือวิธีการ Min-variance และ DJIA

Koya Ishikawa, Kazuhide Nakata, 2021 ได้ทำการค้นคว้าศึกษาบทความงานวิจัยเกี่ยวกับการซื้อขายสกุลเงินในตลาด Forex โดยคำนึงถึงค่าธรรมเนียมที่ต้องเสียในการซื้อขายเป็นปัจจัยหลัก [11] เนื่องจากตลาด Forex มีค่าธรรมเนียมที่หลากหลาย งานศึกษาวิจัยนี้จึงได้ออกแบบฟังก์ชันการให้รางวัล (Reward Function) ขึ้นมาใหม่เพื่อให้ตัวแทน (Agent) สามารถคำนึงถึงค่าธรรมเนียมในการซื้อขายเป็นหลัก ดังนั้นตัวแทนจะพยายามทำการซื้อขายในจำนวนครั้งที่น้อยและใช้เวลาในการถือแต่ละออร์เดอร์ที่นานขึ้นเพื่อหลีกเลี่ยงการเสียค่าธรรมเนียมจากการซื้อขายที่สั้นและมีจำนวนครั้งในการซื้อขายที่ถี่กว่า กล่าวคือตัวแทนต้องพยายามเพิ่มผลกำไรให้ได้สูงที่สุดในแต่ละคำสั่งซื้อ และเนื่องจากตัวแทนจะถือคำสั่งซื้อที่นานขึ้นจึงมีการปรับใช้กระบวนการเรียนรู้แบบออนไลน์ (Online learning) เข้ามาใช้เพื่อให้ระบบยังคงสามารถเรียนรู้ได้อย่างต่อเนื่องและได้รับข้อมูลใหม่ ๆ เข้ามาแทนการเรียนรู้จากข้อมูลเก่าเพียงอย่างเดียว โดยงานวิจัยนี้ได้ใช้โมเดล Long

Short-Term Memory (LSTM) ในการสกัดคุณสมบัติบางอย่างของข้อมูลออกมาเนื่องจากข้อมูลเป็นอนุกรมเวลา ดังนั้นช่วงเวลาและลำดับการเกิดจึงมีความสำคัญต่อการเรียนรู้ของตัวแทน จากนั้นนำผลลัพธ์ไปประมวลผลกับ Fully Connected Neural Network เพื่อหาค่า Q-value ออกมา 3 ค่า คือ 1. ค่า Q สำหรับ Positive Position 2. ค่า Q สำหรับ No Position และ 3. ค่า Q สำหรับ Negative Position เพื่อมาเป็นข้อมูลตัดสินใจให้กับตัวแทน ซึ่งผู้พัฒนาได้เรียกชื่ออัลกอริทึมนี้ว่า Deep Q-Network for Online Trade โดยงานวิจัยนี้ใช้ข้อมูลเป็นคู่สกุลเงิน USD/JPY โดยมีช่วงเวลาของข้อมูลเป็นทุก 1 นาทีตั้งแต่วันที่ 01/01/2020 ถึงวันที่ 12/31/2020 ซึ่งผลลัพธ์ที่ได้จากงานวิจัยพบว่าตัวแทนสามารถลดปริมาณการซื้อขายลงได้ เมื่อมีการกำหนดค่าธรรมเนียมในการซื้อขายที่สูงขึ้นในขณะที่ตัวแทนยังคงสามารถทำกำไรกลับมาได้ แต่ถ้าหากค่าใช้จ่ายสูงถึง 0.1% ของเงินที่ใช้ลงทุนตัวแทนของงานวิจัยนี้ยังคงทำงานได้ไม่ดี และมีการขาดทุน

Uk Jo, Taehyun Jo, Wanjun Kim, Iljoo Yoon, Dongseok Lee, Seungho Lee, 2019 ได้ทำการค้นคว้าศึกษาบทความงานวิจัยเกี่ยวกับการซื้อขายหุ้นด้วยวิธีการซื้อขายแบบ Scalping โดยใช้การร่วมมือกันของตัวแทนหลายตัวแทน (Cooperative Multi-Agent) ที่ถูกฝึกสอนการซื้อขายด้วยวิธีการเรียนรู้แบบเสริมแรงเชิงลึก (Deep Reinforcement Learning) [12] การซื้อขายแบบ Scalping คือการซื้อขายในระยะเวลาสั้น ๆ ที่ใช้เวลาเพียงไม่กี่นาทีในการเปิดและปิดออร์เดอร์การซื้อขายนั้น ๆ ดังนั้นตัวแทนต้องมีข้อมูลที่ใช้ในการตัดสินใจมากพอที่จะเลือกตำแหน่งเปิดและปิดออร์เดอร์ โดยงานศึกษาวิจัยครั้งนี้มีการนำหลักการของโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network) มาใช้ในการประมวลผลข้อมูล ในที่นี่คือใช้ Conv3D สำหรับข้อมูลคำสั่งซื้อและ Conv1D สำหรับข้อมูลการชำระเนื่องจากงานวิจัยนี้มองว่าการทำคอนโวลูชันสามารถสกัดคุณสมบัติที่สำคัญจากข้อมูลที่เป็นอนุกรมเวลาได้ดี ซึ่งงานวิจัยนี้ได้ใช้ตัวแทนทั้งหมดถึง 4 ตัวแทนซึ่งแต่ละตัวแทนมีจุดประสงค์ที่แตกต่างกันคือ 1. ตัวแทนที่ทำหน้าที่ค้นหาสัญญาณการเข้าซื้อ (Buy Signal Agent) เป็นตัวแทนที่ทำหน้าที่คาดการณ์ราคาหุ้นว่าจะเพิ่มขึ้นหรือไม่ในอีก 2 นาทีข้างหน้านับจากเวลาปัจจุบัน 2. ตัวแทนที่ทำหน้าเข้าซื้อ (Buy Order Agent) จะดูจากราคาคาดการณ์ที่ได้จากตัวแทนที่หนึ่งมาตัดสินใจว่าควรเข้าซื้อหรือไม่ ถ้าควรเข้าซื้อจะเข้าซื้อที่จุดไหน ที่จะได้ราคาที่ต่ำที่สุด 3. ตัวแทนที่ทำหน้าที่ค้นหาสัญญาณการสั่งขาย (Sell Signal Agent) หลังจากเกิดการเข้าซื้อจากตัวแทนที่สอง ตัวแทนที่สามจะทำหน้าที่คาดการณ์ราคาหุ้นว่าจะลดลงหรือไม่ในอีก 2 นาทีข้างหน้านับจากเวลาปัจจุบัน และ 4. ตัวแทนที่ทำหน้าที่ขาย (Sell Order Agent) จะดูจากราคาคาดการณ์ที่ได้จากตัวแทนที่สามมาตัดสินใจว่าควรขายหรือไม่ ถ้าควรขายจะขายที่จุดไหน ที่จะได้ราคาที่สูงที่สุด และเมื่อทำการขายเสร็จสิ้นก็จะวนกลับไปที่ตัวแทนที่หนึ่งเพื่อค้นหาสัญญาณการเข้าซื้อใหม่อีกครั้ง ดังนั้นตัวแทนแต่ละตัวจะมีปฏิสัมพันธ์ซึ่งกันและกัน โดยอัลกอริทึมที่ตัวแทนใช้ในการเรียนรู้จะใช้อัลกอริทึมที่ชื่อว่า Double Deep Q-network (DDQN) ซึ่งในงานศึกษาวิจัยนี้ได้ใช้ข้อมูลราคาหุ้นของ KOSPI และ KOSDAQ ตั้งแต่เดือนเมษายนในปี 2018 ถึงเดือนกรกฎาคมในปี 2018 ซึ่งผลลัพธ์ที่ได้จากการทดลองพบว่าตลาดหุ้นที่งานวิจัยนี้ได้เลือกมานั้นไม่ค่อยเหมาะกับวิธีการ

ซื้อขายแบบ Scalping เนื่องจากค่าธรรมเนียมที่เสียในแต่ละการซื้อขายที่สูงและอีกปัจจัยหนึ่งก็คือตลาดหุ้นเกาหลีไม่ได้มีความผันผวนที่สูงมากพอในการซื้อขายหุ้นในระยะเวลาสั้น ๆ แต่ถึงกระนั้นผลลัพธ์ที่ได้จากการทดลองพบว่าตัวแทนยังสามารถทำกำไรได้ถึงแม้จะไม่สูงมาก แต่ยังคงตรงกับจุดประสงค์ของการวิจัยในครั้งนี้

Francisco Caio Lima Paiva, Leonardo Kanashiro Felizardo, Reinaldo Augusto da Costa Bianchi, Anna Helena Reali Costa, 2021 ได้ทำการค้นคว้าศึกษาบทความงานวิจัยเกี่ยวกับการซื้อขายหุ้นด้วยวิธีการเรียนรู้แบบเสริมแรงที่มีการตระหนักถึงความรู้สึกหรืออารมณ์ของตลาด (Sentiment-Aware Reinforcement Learning) [13] โดยงานวิจัยนี้ได้กล่าวถึงปัญหาหลัก ๆ 2 ปัญหาที่งานวิจัยอื่น ๆ ไม่ได้แก้ไขหรือกล่าวถึงคือ 1. การสกัดแรงเหวี่ยงของความรู้สึกของตลาด ที่จะสะท้อนถึงสภาวะตลาดในแต่ละช่วงเวลา กล่าวคือความรู้สึกของตลาด ณ ช่วงเวลานั้นมีความรุนแรงมากน้อยเพียงใด และ 2. ความเสถียรของวิธีการเรียนรู้แบบเสริมแรงในสถานการณ์ต่าง ๆ กล่าวคือตัวแทนสามารถทำงานได้ดีมากน้อยเพียงใด (Generalization) ในสถานการณ์ที่ไม่อาจจะไม่เคยพบเห็นมาก่อน ดังนั้นนักวิจัยจึงได้คิดค้นวิธีการเรียนรู้แบบเสริมแรงที่มีการตระหนักถึงความรู้สึกของตลาดขึ้นมา โดยตั้งชื่อว่า Sentiment-Aware RL เพื่อแก้ปัญหา 2 ข้อจากที่กล่าวไป ในงานวิจัยนี้นักวิจัยได้เลือกใช้ข้อมูลหุ้นในระดับชั่วโมงจำนวน 3 ปีตั้งแต่ปี 2018 – 2020 ของหุ้นจำนวน 20 หุ้นในแต่ละอุตสาหกรรมที่แตกต่างกันเช่น โรงงาน เทคโนโลยี การเงินหรือพลังงานเป็นต้น ซึ่งข้อมูลจะประกอบไปด้วยราคาปิดของแต่ละหุ้นและข่าวที่เกี่ยวข้องของแต่ละหุ้น ดังนั้นนักวิจัยจึงออกแบบระบบออกมาเป็น 2 ส่วนหลักคือ 1. โมเดลที่ทำหน้าที่สกัดความรู้สึกออกมาจากข่าวในช่วงเวลานั้น ๆ และ 2. ส่วนที่เป็นตัวแทนในการตัดสินใจจากข้อมูลความรู้สึกและข้อมูลราคาในช่วงเวลาขณะนั้น ๆ ในส่วนของโมเดลสกัดความรู้สึกนั้น ผู้วิจัยได้เลือกใช้อัลกอริทึม Convolutional Neural Network (CNN) ในการเรียนรู้การสกัดความรู้สึกออกมาจากข่าวทองคำ โดยผลลัพธ์ที่ได้จากโมเดลจะเป็นคะแนนอารมณ์ในช่วง -1 ถึง 1 ซึ่งจะต้องมีกระบวนการในการรวบรวมคะแนนในช่วงเวลานั้น ๆ ให้อยู่ในระดับชั่วโมง โดยนักวิจัยพบว่าวิธีการรวมคะแนนโดยการเลือกค่าต่ำที่สุดในชั่วโมงนั้น ๆ ให้ผลลัพธ์ที่ดีที่สุด กว่าวิธีการหาค่าเฉลี่ยหรือค่าสูงสุด ในส่วนของตัวแทนที่ทำหน้าที่ตัดสินใจจากข้อมูลความรู้สึกและข้อมูลราคาหุ้นนั้นนักวิจัยได้เลือกใช้อัลกอริทึมที่มีชื่อว่า Advantage Actor-Critic (A2C) ในการฝึกสอนจากข้อมูลเหล่านี้เพื่อเลือกการกระทำที่ดีที่สุดในแต่ละชั่วโมงนั้น ๆ ว่าควรเข้าซื้อ ถือรอ หรือควรออกเป็นต้น จากการทดลองพบว่าตัวแทนที่มีการตระหนักถึงอารมณ์ของตลาดสามารถทำกำไรโดยเฉลี่ยได้มากกว่าตัวแทนที่ไม่ได้นำอารมณ์ของตลาดในตัดสินใจสูงถึง 43% อีกทั้งเมื่อเทียบกับกลยุทธแบบเก่าคือกลยุทธ Buy-and-Hold ตัวแทนที่ตระหนักถึงอารมณ์ของตลาดสามารถทำกำไรโดยเฉลี่ยได้มากกว่าถึง 17%

บทที่ 3
วิธีการดำเนินงานวิจัย

ในชีวิตจริงการลงทุนซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ มักจะมีการวางแผนและการวิเคราะห์ต่าง ๆ เพื่อหาจุดเข้าซื้อขาย และจุดออกที่เหมาะสม ซึ่งปกตินักลงทุนจะทำการวิเคราะห์ปัจจัยพื้นฐาน (Fundamental Analysis) หรือการวิเคราะห์เชิงเทคนิค (Technical Analysis) หรือทั้ง 2 อย่างขึ้นอยู่กับประสบการณ์ของนักลงทุนแต่ละคน ซึ่งในการทดลองนี้จะเป็นการจำลองนักลงทุนที่วิเคราะห์เชิงเทคนิคเท่านั้น โดยปกติแล้วนักลงทุนจะใช้ตัวชี้วัดเชิงเทคนิคหลากหลายประเภทในการวิเคราะห์ ซึ่งค่าชี้วัดเชิงเทคนิคนี้ขึ้นอยู่กับกรอบของเวลาที่นักลงทุนต้องนำมาพิจารณาด้วย โดยกรอบเวลาที่มักถูกนำมาใช้ในการวิเคราะห์คือ กรอบเวลา 1 ชั่วโมง กรอบเวลา 6 ชั่วโมง กรอบเวลา 1 วันและอื่น ๆ แล้วแต่ความชำนาญของแต่ละคน จากนั้นนักลงทุนก็จะใช้ประสบการณ์ในการตัดสินใจจากพฤติกรรมของราคาและค่าชี้วัดเชิงเทคนิคเหล่านี้ในการวางแผนจุดเข้าซื้อขายและจุดออกที่เหมาะสมนั่นเอง ซึ่งเราจะจำลองกระบวนการเหล่านี้ให้กับตัวแทน (Agent) ของเราในการตัดสินใจวางแผนจุดเข้าซื้อขายและจุดออกที่เหมาะสม

วิธีการดำเนินงานวิจัยการศึกษาเปรียบเทียบประสิทธิภาพของการซื้อขายเงินตราต่างประเทศด้วยอัลกอริทึมที่แตกต่างกัน เพื่อให้การวิจัยครั้งนี้บรรลุวัตถุประสงค์ที่ตั้งไว้ ผู้วิจัยได้กำหนดวิธีดำเนินการวิจัยซึ่งมีรายละเอียดในการดำเนินงานวิจัยดังนี้

3.1 ขั้นตอนในการวิจัย

ในการค้นคว้าอิสระนี้เป็นการศึกษาอัลกอริทึมการเรียนรู้แบบเสริมแรงเชิงลึกเพื่อนำไปประยุกต์กับการซื้อขายชุดข้อมูลคู่สกุลเงิน โดยมีขั้นตอนในการทดลองดังต่อไปนี้

ขั้นตอนที่ 1 ศึกษาความรู้พื้นฐานเกี่ยวกับตลาดแลกเปลี่ยนเงินตราต่างประเทศ การซื้อขายคู่สกุลเงิน และศึกษาการทำงานของอัลกอริทึมการเรียนรู้แบบเสริมแรงเชิงลึก

ขั้นตอนที่ 2 กำหนดขอบเขตและดำเนินการรวบรวมข้อมูลราคาคู่สกุลเงินในตลาดแลกเปลี่ยนเงินตราต่างประเทศในช่วงปี 2558 - 2565

ขั้นตอนที่ 3 ออกแบบสภาพแวดล้อมสำหรับการฝึกสอนและฟังก์ชันการให้รางวัล

ขั้นตอนที่ 4 ดำเนินการสร้าง Agent ฝึกสอน ปรับปรุงและทดสอบอัลกอริทึมในการซื้อขายคู่สกุลเงินกับสภาพแวดล้อมสำหรับการทดสอบ

ขั้นตอนที่ 5 นำผลมาเปรียบเทียบการซื้อขายคู่สกุลเงินว่า Agent ตัวไหนให้ผลลัพธ์ที่ดีที่สุดในแต่ละรอบการปรับค่าพารามิเตอร์

ขั้นตอนที่ 6 สรุปผลและนำเสนอ

3.2 ชุดข้อมูลและวิธีเก็บรวบรวมข้อมูลที่ใช้ในการทดลอง

ชุดข้อมูลที่ใช้ในการทดลองจะเป็นข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ตั้งแต่ปี 2558 – 2565 ซึ่งเป็นคู่สกุลเงินหลักที่นักลงทุนมีการซื้อขายมากที่สุด โดยจะเก็บรวบรวมผ่านทางผู้ให้บริการข้อมูลทางการเงินคือ Yahoo โดยสามารถดาวน์โหลดผ่าน Library ของผู้พัฒนาได้โดยตรงที่มีชื่อว่า yfinance ซึ่งข้อมูลที่ได้จะอยู่ในรูปแบบดังตารางที่ 3.1

ตารางที่ 3.1 ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD

Date	Open	High	Low	Close
2015-01-01	1.209863	1.209863	1.209863	1.209863
2015-01-02	1.208868	1.208956	1.201080	1.208941
2015-01-03	1.195500	1.197590	1.188909	1.194643
2015-01-04	1.193830	1.197000	1.188693	1.193902

จากนั้นจะมีการคำนวณหาค่าชี้วัดเชิงเทคนิคเพื่อเป็นข้อมูลสนับสนุนในการหาจุดซื้อขายเช่น RSI ATR หรือ Parabolic SAR เป็นต้น แต่เนื่องจากแต่ละตัวชี้วัดเชิงเทคนิคมีการใช้ปริมาณวันในการคำนวณที่แตกต่างกันดังนั้นข้อมูลในช่วงแรกบางส่วนอาจไม่มีค่าได้ โดยการแก้ปัญหานี้จึงใช้วิธีการเติมค่าย้อนหลังโดยใช้ค่าล่าสุดก่อนไม่มีค่า (Backward Filling) เพื่อแก้ปัญหาในส่วนนี้ ซึ่งผลลัพธ์ที่ได้จะเป็นดังตารางที่ 3.2

ตารางที่ 3.2 ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ที่ถูกเพิ่มค่าชี้วัดเชิงเทคนิค

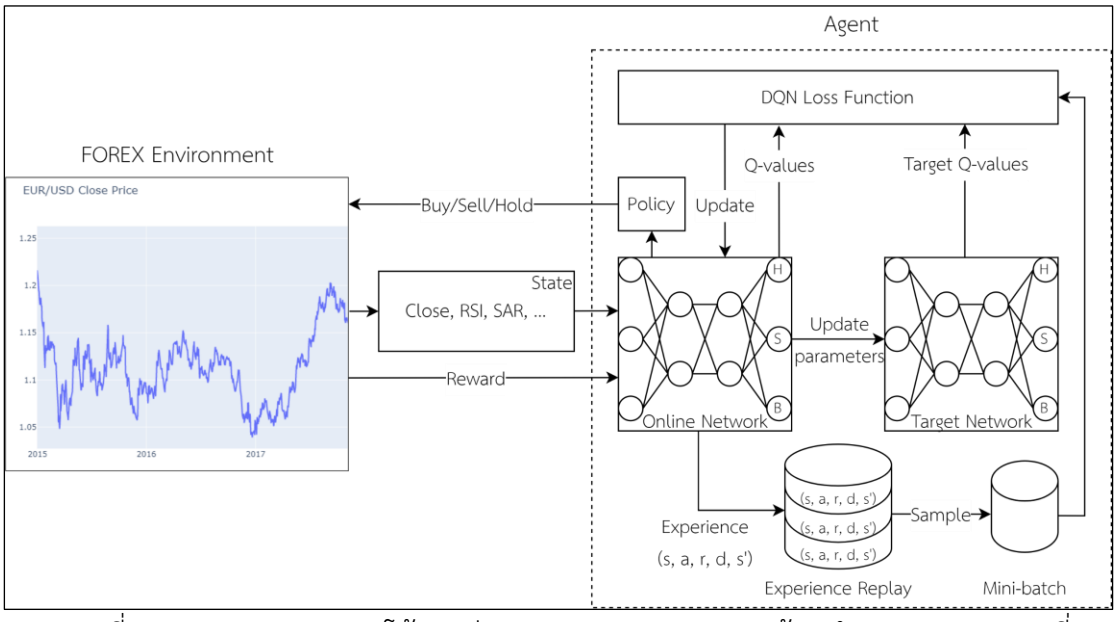
Date	Open	High	Low	Close	RSI	ADX
2015-01-01	1.209863	1.209863	1.209863	1.209863	14.41016	64.122363
2015-01-02	1.208868	1.208956	1.201080	1.208941	14.41016	64.122363
2015-01-03	1.195500	1.197590	1.188909	1.194643	14.41016	64.122363
2015-01-04	1.193830	1.197000	1.188693	1.193902	14.41016	64.122363

จากนั้นข้อมูลในส่วนนี้จะถูกนำไปใช้ในการสร้างสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศเพื่อใช้ในการฝึกสอน Agent ต่อไป

3.3 ภาพรวมของระบบ

ในส่วนของภาพรวมของระบบจะอธิบายถึงการออกแบบและองค์ประกอบของเครื่องมือที่ใช้ในการทดลอง ซึ่งประกอบไปด้วย 2 ส่วนหลักคือ ส่วนของสภาพแวดล้อมจำลองตลาดแลกเปลี่ยน

เงินตราต่างประเทศ (Forex Environment) และส่วนของอัลกอริทึมที่ใช้ในการแก้ปัญหาในสภาพแวดล้อมนั้น โดยการปฏิสัมพันธ์ระหว่าง Agent และสภาพแวดล้อมสามารถแสดงได้ดังแผนภาพที่ 3.1 โดย Agent จะทำหน้าที่ปฏิสัมพันธ์กับสภาพแวดล้อมเพื่อเก็บข้อมูลประสบการณ์และนำประสบการณ์เหล่านี้มาอัพเดทค่าพารามิเตอร์ของโครงข่ายประสาทเทียมให้ดีขึ้นเรื่อย ๆ



แผนภาพที่ 3.1 ภาพรวมการตอบโต้ระหว่าง Agent และสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ

3.3.1 สภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ (Forex Environment)

ในส่วนของสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ เราใช้สภาพแวดล้อมที่ชื่อว่า gym-anytrading เป็นตัวตั้งต้น โดยสภาพแวดล้อม gym-anytrading ถูกพัฒนาและออกแบบโดยคุณ Mohammad Amin Haghpanah ซึ่งมีการพัฒนาออกมา 2 สภาพแวดล้อมคือ 1. ตลาดหุ้น และ 2. ตลาดแลกเปลี่ยนเงินตราต่างประเทศ เพื่อนำมาใช้ในการฝึกสอนอัลกอริทึมการเรียนรู้แบบเสริมแรงอย่างง่ายในการซื้อขายหุ้นหรือซื้อขายคู่สกุลเงิน ในที่นี่เราจะนำสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศมาทำการปรับปรุง โดยทางผู้พัฒนาได้ออกแบบตำแหน่ง (Position) ของการซื้อขายมาอย่างง่ายเพียง 2 ตำแหน่งคือเข้าซื้อ (Long) และเข้าขาย (Short) แต่ในความเป็นจริงควรจะมีตำแหน่งที่รอการเข้าซื้อหรือขายเพิ่มขึ้นมา และทางผู้พัฒนาได้ออกแบบการกระทำ (Action) มาเพียง 2 การกระทำคือซื้อ (Buy) และขาย (Sell) เท่านั้น แต่ในความเป็นจริงควรจะมีการกระทำสำหรับการถือต่อ (Hold) ในกรณีที่มีการซื้อขายแล้วอีกด้วย ทั้งนี้เพื่อให้สภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศมีความเหมาะสมมากยิ่งขึ้น จึงจำเป็นต้องพัฒนาสิ่งเหล่านี้เพิ่มเติมคือ

1. เพิ่มตำแหน่งการรอเข้าซื้อหรือเข้าขาย (Flat) ไปยังสภาพแวดล้อม
2. เพิ่มการกระทำถือต่อ (Hold) ของราคาที่เข้าซื้อหรือขายไปยังสภาพแวดล้อม
3. ปรับปรุงข้อมูลสถานะ (State) ที่จะส่งจากสภาพแวดล้อมให้กับอัลกอริทึมนำไปตัดสินใจเลือกการกระทำ โดยจะเพิ่มตำแหน่งปัจจุบันเข้าข้อมูลสถานะเพื่อให้อัลกอริทึมสามารถรู้ตำแหน่งการซื้อขาย ณ ปัจจุบันได้
4. ปรับปรุงกลไกการดำเนินการซื้อขายให้สามารถทำงานร่วมกับตำแหน่งและการกระทำที่เพิ่มเข้ามาได้
5. ปรับปรุงฟังก์ชันการให้รางวัล เพื่อให้อัลกอริทึมทำงานในรูปแบบที่ควรจะเป็น เช่น ไม่มีการเข้าซื้อซ้ำ หรือไม่มีการเข้าขายซ้ำเป็นต้น
6. ปรับปรุงวิธีการคำนวณกำไรขาดทุน เพื่อให้ง่ายต่อการทำความเข้าใจจึงคิดคำนวณเป็นราคาที่แตกต่างจากราคาที่ซื้อขายคูณกับปริมาณที่ทำการซื้อขาย
7. ปรับปรุงช่วงของรางวัลเพื่อเพิ่มประสิทธิภาพการเรียนรู้ของ Agent ให้มีความเสถียรมากยิ่งขึ้น
8. สร้างสภาพแวดล้อมสำหรับการฝึกสอนและสภาพแวดล้อมสำหรับการทดสอบ

3.3.2 Agent ที่ใช้ในการแก้ปัญหา

ในส่วนของ Agent ที่ใช้ในการแก้ปัญหาจะต้องทำหน้าที่รับข้อมูลที่ได้จากการตอบโต้กับสภาพแวดล้อม และค่ารางวัลที่ได้ในแต่ละการกระทำ มาปรับปรุงสมองของ Agent หรือค่าพารามิเตอร์ต่าง ๆ ของโครงข่ายประสาทเทียม ซึ่งการพัฒนาจะเขียนตามหลักการเขียนโปรแกรมเชิงวัตถุ (Object-oriented Programming) เพื่อให้มอง Agent ที่มีอัลกอริทึมในการแก้ปัญหา 1 แบบเป็น Agent 1 ตัวและเพื่อให้ง่ายต่อการสืบทอดคุณสมบัติที่คล้ายกันเพื่อลดปริมาณและความซับซ้อนในการพัฒนาอีกด้วย ซึ่งสิ่งที่ต้องพัฒนาจะประกอบไปด้วยดังนี้

1. นโยบายสำหรับการเลือกการกระทำเช่น การเลือกการกระทำแบบละโมบด้วยค่า Epsilon หรือการเลือกการกระทำแบบละโมบ
2. ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) หรือถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay)
3. การค้นหาและเปรียบเทียบค่าพารามิเตอร์ของโครงข่ายประสาทเทียมเพื่อให้ได้โครงข่ายประสาทเทียมที่ดีที่สุด (Hyperparameter Tuning)
4. พัฒนา Agent ที่ใช้อัลกอริทึมและถังเก็บข้อมูลประสบการณ์ที่แตกต่างกันดังตารางที่ 3.3 และมีการติดตามค่าพารามิเตอร์ ค่ารางวัล ค่าผลตอบแทนหรือค่าความผิดพลาดต่าง ๆ ลงบน Tensorboard

ตารางที่ 3.3 การพัฒนา Agent สำหรับการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศ

ลำดับ	ประเภทของ Agent	ประเภทถังเก็บข้อมูลประสบการณ์
1.	Double Deep Q-Learning	Experience Replay
2.	Dueling Double Deep Q-Learning	Experience Replay
3.	Double Deep Q-Learning	Prioritized Experience Replay
4.	Dueling Double Deep Q-Learning	Prioritized Experience Replay

การสร้างเครื่องมือในส่วนนี้จะเขียนด้วยภาษา Python และใช้ Library ต่าง ๆ ดังแสดงในตารางที่ 3.4

ตารางที่ 3.4 Library ที่ใช้ในการพัฒนา

Library ที่ใช้ในการพัฒนา	รายละเอียด
gym==0.23.1	ใช้ในการเป็นพื้นฐาน (Base class) ในการพัฒนาสภาพแวดล้อมต่าง ๆ เพื่อให้เป็นมาตรฐานการพัฒนาในรูปแบบเดียวกัน
gym-anytrading==1.2.0	ใช้ในการสร้างสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ
yfinance==0.1.70	ใช้ในการดึงข้อมูลอัตราแลกเปลี่ยนคู่สกุลเงินรายวัน
TA-Lib==0.4.24	ใช้ในการคำนวณค่าชี้วัดเชิงเทคนิคต่าง ๆ เช่น RSI ATR หรือ ADX เป็นต้น
pytorch==1.11.0	ใช้ในการพัฒนาโครงข่ายประสาทเทียม
pytorch-lightning==1.6.2	ใช้ในการควบคุมโครงสร้างการเขียนให้เป็นระบบ การทำ Checkpoint หรือการทำ Logging เป็นต้น
tensorboard==2.9.0	ใช้ในการติดตามสถานะค่าพารามิเตอร์ ค่ารางวัล ค่าผลตอบแทน หรือค่าความผิดพลาดต่าง ๆ
optuna==2.10.0	ใช้ในการค้นหาและปรับปรุงพารามิเตอร์ต่าง ๆ ของโครงข่ายประสาทเทียม
plotly==5.7.0	ใช้สำหรับแสดงผลข้อมูลในรูปแบบกราฟ

3.4 การออกแบบการทดลอง

การทดลองฝึกอบรม Agent ในการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศจะเป็นการเปรียบเทียบประสิทธิภาพระหว่าง Agent ในแง่มุมต่าง ๆ ดังตารางที่ 3.5

ตารางที่ 3.5 ตัวชี้วัดสำหรับการเปรียบเทียบประสิทธิภาพระหว่าง Agent

ลำดับ	ตัวชี้วัด	ตำแหน่งการวัดผล
1.	Loss หรือ Q-Error	Episode สุดท้ายของการฝึกสอน
2.	ผลตอบแทนสะสมของการลงทุน (Investment Return) เฉลี่ย 50 Episode	Episode สุดท้ายของการฝึกสอน
3.	ผลตอบแทนสะสมของการลงทุน (Investment Return)	Episode สุดท้ายของการทดสอบ

ในส่วนของการทดลองจะแบ่งการทดลองออกเป็น 3 การทดลองได้แก่

1. การทดลองที่ 1 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับปริมาณข้อมูลในการสุ่มมาเรียนรู้ (Batch Size) ในการทดลองนี้จะเป็นการศึกษาผลกระทบต่อปริมาณข้อมูลในการสุ่มมาเรียนรู้ในแต่ละครั้งว่ามีผลต่อการตัดสินใจของแต่ละ Agent มากน้อยเพียงใดระหว่างปริมาณข้อมูลในการเรียนรู้ที่ละน้อย ๆ กับปริมาณข้อมูลในการเรียนรู้ทีละมาก ๆ
2. การทดลองที่ 2 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับอัลกอริทึมในการเรียนรู้ (Optimizer) และอัตราการเรียนรู้ (Learning Rate) ในการทดลองนี้จะเป็นการศึกษาความเร็วในการเรียนรู้ของแต่ละอัลกอริทึมในการเรียนรู้และอัตราการเรียนรู้ที่แตกต่างกันว่า Agent สามารถเรียนรู้ได้เร็วมากน้อยเพียงใดและมีประสิทธิภาพในการซื้อขายต่างกันมากน้อยเพียงใดในปริมาณรอบการฝึกสอนที่เท่ากัน
3. การทดลองที่ 3 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับการให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคต (Gamma) ในการทดลองนี้จะเป็นการศึกษาผลกระทบต่อความสำคัญของรางวัลที่คาดหวังในอนาคตว่ามีผลต่อการตัดสินใจของแต่ละ Agent มากน้อยเพียงใด

โดยในแต่ละการทดลองจะมีการกำหนดค่าพารามิเตอร์ดังตารางที่ 3.6

ตารางที่ 3.6 ค่าพารามิเตอร์สำหรับการทดลอง

Hyperparameters	Description	Value
window_size	ปริมาณข้อมูลย้อนหลัง	4

ตารางที่ 3.6 ค่าพารามิเตอร์สำหรับการทดลอง (ต่อ)

Hyperparameters	Description	Value
gamma	ค่าน้ำหนักรางวัลที่คาดหวังในอนาคต	0.1, 0.3, 0.5, 0.7, 0.9
capacity	ขนาดของถังเก็บข้อมูลประสบการณ์	100,000
batch_size	ปริมาณข้อมูลในการสุ่มมาเรียนรู้	64, 128, 256
optimizer	อัลกอริทึมในการเรียนรู้	AdamW, RMSprop
lr	อัตราการเรียนรู้	0.001, 0.0005
hidden_size	ปริมาณ Hidden Node	512
loss_fn	Loss function ที่ใช้เป็น Smooth L1 Loss หรือ Huber Loss	smooth_l1_loss
sync_rate	ปริมาณรอบในการคัดลอกน้ำหนักจากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ให้กับโครงข่ายประสาทเทียมสำหรับการประเมิน	50
samples_per_epoch	ปริมาณ sample ที่จะถูกหยิบสุ่มไว้ในแต่ละครั้ง โดย Agent จะสุ่มหยิบตามขนาด batch_size จาก samples ที่ถูกหยิบออกมาจากถังเก็บข้อมูลประสบการณ์	1,000
eps_start	ค่า Epsilon เริ่มต้น	1.0
eps_end	ค่า Epsilon สิ้นสุด	0.15
eps_last_episode	จำนวนรอบที่จะปรับค่า Epsilon เริ่มต้นลดลงจนถึงค่าสิ้นสุด	300
alpha_start	ค่า Alpha เริ่มต้น	1.0
alpha_end	ค่า Alpha สิ้นสุด	0.0
alpha_last_episode	จำนวนรอบที่จะปรับค่า Alpha เริ่มต้นลดลงจนถึงค่าสิ้นสุด	200
beta_start	ค่า Beta เริ่มต้น	0.5
beta_end	ค่า Beta สิ้นสุด	1.0
beta_last_episode	จำนวนรอบที่จะปรับค่า Beta เริ่มต้นเพิ่มขึ้นจนถึงค่าสิ้นสุด	200
max_epochs	จำนวนรอบในการฝึกสอน	600

3.5 รายละเอียดการพัฒนาสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ

ในส่วนสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศนั้นจะนำข้อมูลในส่วนของหัวข้อ 3.2 มาใช้งาน โดยเราสามารถเลือกได้ว่าเราอยากใช้ข้อมูลอะไรบ้างในการฝึกสอน ในที่นี่ขอกำหนดเป็นใช้ราคาปิด (Close Price) ในการซื้อขายและใช้ค่าชี้วัดเชิงเทคนิคดังนี้ 1. RSI 2. ADX 3. SAR 4. ATR 5. EMA 6. MACD และ 7. Bollinger Band เป็นข้อมูลให้กับ Agent ในการตัดสินใจเลือกการกระทำที่ดีที่สุดในแต่ละ Timestep และกำหนดปริมาณการซื้อขาย (lot size) เป็น 10,000 เพื่อให้เมื่อมีการคำนวณกำไรหรือขาดทุน ค่าผลลัพธ์จะได้มีค่าไม่น้อยเกินไป

3.5.1 กำหนดการกระทำที่เป็นไปได้ทั้งหมดในสภาพแวดล้อม

ในส่วนของการกระทำจะเป็นการเพิ่มการถือ (Hold) เข้าไปยังสภาพแวดล้อม โดยจะกำหนดเป็นประเภท Enum และกำหนดค่าดังตารางที่ 3.7

ตารางที่ 3.7 การกระทำที่เป็นไปได้ทั้งหมด

ชื่อการกระทำ	คำอธิบายการกระทำ	ค่าของการกระทำ
Hold	ถืออัตราแลกเปลี่ยนต่อหรือกำลังรอเข้าซื้อหรือขาย	0
Sell	ทำการเทขาย	1
Buy	ทำการเข้าซื้อ	2

3.5.2 กำหนดตำแหน่งที่เป็นไปได้ทั้งหมดในสภาพแวดล้อม

ในส่วนของตำแหน่งจะเป็นการเพิ่มการรอเข้าซื้อหรือขาย (Flat) เข้าไปยังสภาพแวดล้อม โดยจะกำหนดเป็นประเภท Enum และกำหนดค่าดังตารางที่ 3.8

ตารางที่ 3.8 ตำแหน่งที่เป็นไปได้ทั้งหมด

ชื่อของตำแหน่ง	คำอธิบายตำแหน่ง	ค่าของตำแหน่ง
Short	ทำการเข้าขาย	-1
Flat	ทำการรอเพื่อหาตำแหน่งเข้าซื้อหรือเข้าขาย	0
Long	ทำการเข้าซื้อ	1

3.5.3 ปรับปรุงข้อมูลสถานะ (State)

ในส่วนของข้อมูลสถานะจะมีการเพิ่มตำแหน่งเข้าไปยังข้อมูลสถานะเพื่อให้ Agent นำข้อมูลส่วนนี้ไปตัดสินใจเลือกการกระทำ โดยเราสามารถกำหนดขนาดกรอบเวลาของข้อมูลย้อนหลัง (window_size) เพื่อเป็นข้อมูลสำหรับการตัดสินใจใน 1 สถานะได้ ตัวอย่างเรากำหนดกรอบเวลาข้อมูลย้อนหลังเป็น 3 วันกล่าวคือเราต้องการข้อมูล 3 วันย้อนหลังนับจากวันปัจจุบัน เช่น ปัจจุบันวันที่ 2015-01-03 ดังนั้นข้อมูลสถานะในวันนี้ก็จะมีตั้งแต่วันที่ 2015-01-01 ถึงวันที่ 2015-01-03 นั้นเอง ดังตารางที่ 3.9

ตารางที่ 3.9 ข้อมูล 3 วันย้อนหลังในกรณีที่วันปัจจุบันเป็นวันที่ 2015-01-03

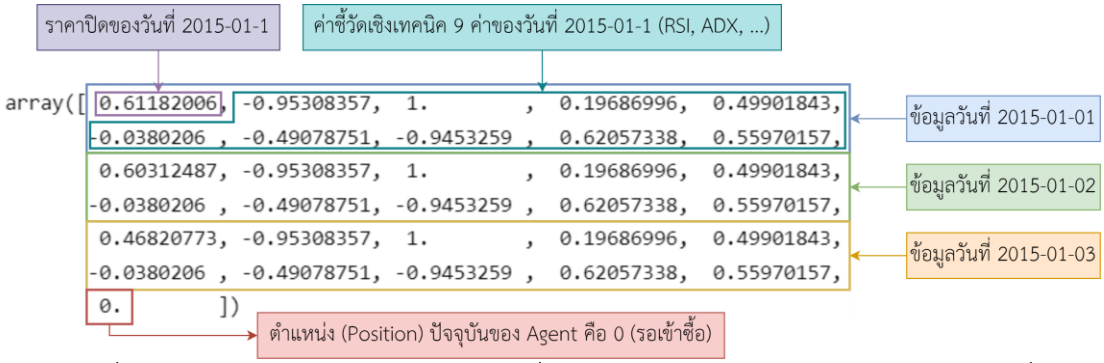
Date	Close	RSI	ADX	...	B_LowerBand
2015-01-01	1.209863	14.41016	64.122363	...	1.18124
2015-01-02	1.208941	14.41016	64.122363	...	1.18124
2015-01-03	1.194643	14.41016	64.122363	...	1.18124

จากนั้นจะมีการนำข้อมูลในส่วนนี้ไปประมวลผลต่อในการปรับช่วงของข้อมูลให้เหมาะสมกัน โดยในที่นี่จะปรับให้อยู่ในช่วง -1 ถึง 1 เพื่อให้สอดคล้องกับตำแหน่งของการซื้อขายที่ทำการออกแบบมา โดยใช้วิธีการ Min-Max Normalization ดังตารางที่ 3.10

ตารางที่ 3.10 ข้อมูล 3 วันย้อนหลังนับตั้งแต่วันปัจจุบันที่ถูกปรับช่วงข้อมูล

Date	Close	RSI	ADX	...	B_LowerBand
2015-01-01	0.611820	-0.95308	1.0	...	0.45481
2015-01-02	0.603124	-0.95308	1.0	...	0.45481
2015-01-03	0.468207	-0.95308	1.0	...	0.45481

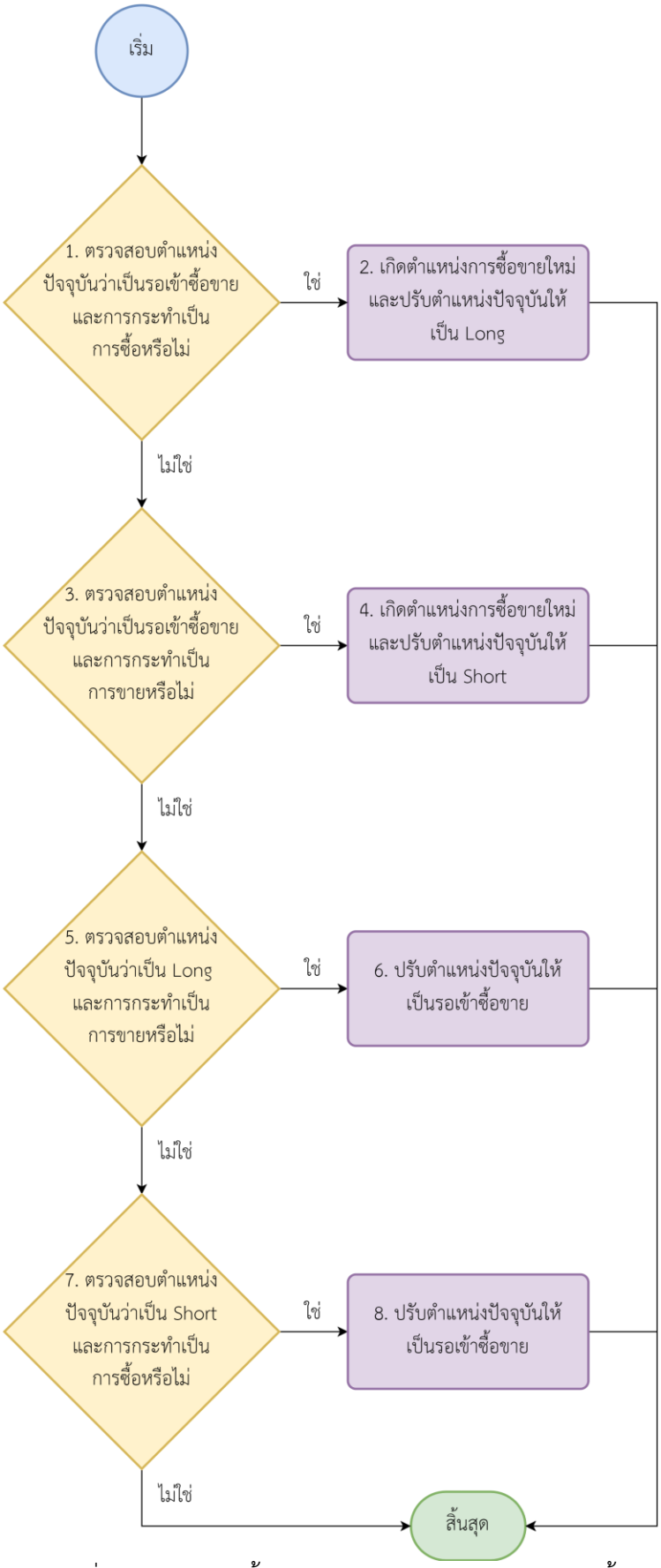
เนื่องจากโครงข่ายประสาทเทียมต้องการรับข้อมูลที่อยู่ในรูปของอาเรย์ 1 มิติดังนั้นเราต้องมีการปรับโครงสร้างของข้อมูล จากนั้นก็จะทำการเติมตำแหน่งล่าสุดเข้าไปยังตำแหน่งสุดท้ายของอาเรย์ดังนั้นข้อมูลที่ Agent จะได้รับในแต่ละ Timestep จะอยู่ในรูปแบบดังแผนภาพที่ 3.2 เป็นต้น



แผนภาพที่ 3.2 แสดงถึงตัวอย่างข้อมูลสถานะที่ Agent จะได้รับใน 1 Timestep ในกรณีที่ขนาดกรอบข้อมูลย้อนหลังเป็น 3 วัน และมีค่าชี้วัดเชิงเทคนิค 9 ตัว

3.5.4 ปรับปรุงกลไกการดำเนินการซื้อขายเงินตราต่างประเทศ

จากเดิมที่มีเพียง 2 การกระทำและ 2 ตำแหน่งการซื้อขายแต่ในที่นี่เรามีการเพิ่มการกระทำถือต่อ (Hold) และตำแหน่งการรอเข้าซื้อขาย (Flat) ดังนั้นเงื่อนไขการเกิดการซื้อขายจะต้องมีการปรับปรุงสามารถเขียนเป็นลำดับการทำงานได้ดังแผนภาพที่ 3.3



แผนภาพที่ 3.3 ลำดับการซื้อขายและการปรับปรุงตำแหน่งการซื้อขาย

จากแผนภาพที่ 3.3 เมื่อมีการกระทำซื้อ (Buy) และตำแหน่งปัจจุบันเป็นการรอเข้าซื้อ (Flat) เท่ากับว่าจะเกิดการซื้อขายดังนั้นจึงให้ trade=True และปรับตำแหน่งการซื้อขายเป็นตำแหน่ง Long แต่ถ้าการกระทำเป็นการขาย (Sell) และตำแหน่งปัจจุบันเป็นการรอเข้าขาย (Flat) เท่ากับว่าจะเกิดการขายดังนั้นจึงให้ trade=True และปรับตำแหน่งการซื้อขายเป็นตำแหน่ง Short แต่ในทางกลับกันถ้าการกระทำเป็นขาย (Sell) และตำแหน่งปัจจุบันเป็น Long เท่ากับว่าเป็นการขายราคาที่ถืออยู่ดังนั้นจะถือว่ายังไม่เกิดการ trade ใหม่และตำแหน่งปัจจุบันจะถูกปรับกลับไปเป็นรอเข้าซื้อขาย (Flat) และสุดท้ายหากการกระทำเป็นซื้อ (Buy) และตำแหน่งปัจจุบันเป็น Short เท่ากับว่าเป็นการเทซื้อกับราคาที่สั่งขายมาดังนั้นจะถือว่ายังไม่เกิดการ trade ใหม่และตำแหน่งปัจจุบันจะถูกปรับไปเป็นรอเข้าซื้อขาย (Flat) สามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.4

```
trade = False
# ทำการเข้าซื้อ (Buy)  แสดงว่า Positions  จะเปลี่ยนเป็น Long
if action == Actions.Buy.value and self._position == Positions.Flat:
    trade = True
    self._position = Positions.Long
# ทำการเข้าขาย (Sell)  แสดงว่า Positions  จะเปลี่ยนเป็น Short
elif action == Actions.Sell.value and self._position == Positions.Flat:
    trade = True
    self._position = Positions.Short
# ทำการเทขายจากที่ซื้อ แสดงว่า Positions  จะกลับไปเป็น Flat  (ไม่ถือว่าเกิด Position  ราคาใหม่)
elif action == Actions.Sell.value and self._position == Positions.Long:
    self._position = Positions.Flat
# ทำการเทซื้อจากที่ขาย แสดงว่า Positions  จะกลับไปเป็น Flat  (ไม่ถือว่าเกิด Position  ราคาใหม่)
elif action == Actions.Buy.value and self._position == Positions.Short:
    self._position = Positions.Flat
```

แผนภาพที่ 3.4 เงื่อนไขการซื้อขายและตำแหน่งการซื้อขาย

3.5.5 ปรับปรุงฟังก์ชันการให้รางวัล (Reward Function)

ฟังก์ชันการให้รางวัลจะเป็นส่วนที่กำหนดพฤติกรรมอันควรจะเป็นของ Agent เพราะจะเป็นตัวที่บ่งบอกถึงคุณภาพของการกระทำว่าเหมาะสมหรือไม่อย่างไร เนื่องจากตลาดแลกเปลี่ยนเงินตราต่างประเทศสามารถทำการซื้อขายได้ทั้งขาขึ้น (Long) และขาลง (Short) ดังนั้นการให้รางวัลถึงการกระทำเหล่านี้จึงต้องครอบคลุมการกระทำเหล่านี้ให้ได้มากที่สุด

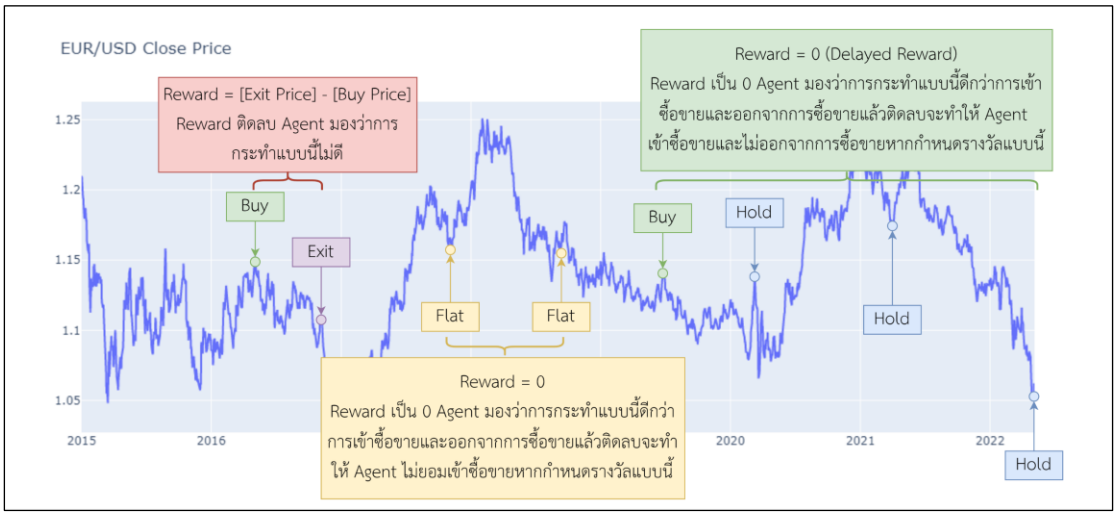
ตัวอย่างการกำหนดฟังก์ชันการให้รางวัลที่ไม่เหมาะสมที่ทาง OpenAI ได้หยิบยกขึ้นมาเป็นกรณีศึกษาคือ การฝึกสอน Agent ในการเล่นเกม CoastRunner [14] ซึ่งเป้าหมายของเกมที่แท้จริงคือ การบังคับเรือให้เข้าเส้นชัยให้เร็วที่สุดเท่าที่จะทำได้และถ้าเป็นไปได้ควรนำหน้าผู้เล่นอื่นทั้งหมด โดยตัวเกมเองก็มีการให้คะแนนในแต่ละการแข่งขัน ซึ่งคะแนนนี้มาจากการทำลายสิ่งกีดขวางระหว่างทางและลำดับในการเข้าเส้นชัย ทำให้นักพัฒนาได้กำหนดคะแนนนี้เป็นเป้าหมายให้กับ Agent ดังนั้นเมื่อเราทำการฝึกสอน Agent เราก็จะคาดหวังว่ามันจะพยายามควบคุมเรือเพื่อเข้าเส้นชัยให้เร็วที่สุด แต่ Agent กลับพบว่าการทำลายสิ่งกีดขวางระหว่างทางสามารถทำคะแนนได้มากกว่าการรีบเข้าเส้นชัย เนื่องจากมันพบว่าเส้นทางที่แข่งกันนั้น มีแอ่งน้ำขนาดใหญ่ที่มันสามารถวนเรือเป็นวงกลมเพื่อทำลายเรือ 3 ลำนั้นได้เรื่อย ๆ จนสุดท้ายมันก็วนทำลายแค่เรือ 3 ลำนั้น โดยไม่มีการเข้า

เส้นชัยใด ๆ ผลจากการทดลองนี้พบว่า Agent สามารถทำคะแนนได้สูงที่สุดและมากกว่าผู้เล่นอื่นโดยเฉลี่ย 20% ดังแผนภาพที่ 3.5 จะพบว่าด้านซ้ายบนของแผนภาพจะเป็นเส้นทางที่ใช้ในการแข่งขัน จะพบว่าผู้เล่นอื่นทำการแข่งกันตามเส้นทางที่กำหนดไว้ ในขณะที่ Agent ของเรานั้นกลับวนทำลายเรืออย่างนี้ไปเรื่อย ๆ โดยไม่สนใจว่ามันจะออกนอกเส้นทางหรือจะไม่เข้าเส้นชัย ด้วยเหตุนี้การกำหนดฟังก์ชันการให้รางวัลนั้นเป็นสิ่งที่ควรคำนึงถึงมากที่สุด ว่าเราจะออกแบบฟังก์ชันการให้รางวัลอย่างไรให้ตรงกับจุดประสงค์ที่เราต้องการ

แผนภาพที่ 3.5 เกม CoastRunner ที่ Agent พยายามทำลายเรือเพื่อเก็บคะแนนในเกมให้ได้มากที่สุด หมายเหตุ. จาก https://www.youtube.com/watch?v=tlOlHko8ySg

สำหรับตลาดแลกเปลี่ยนเงินตราต่างประเทศ เราก็ต้องคำนึงถึงพฤติกรรมที่เราอยากให้ Agent กระทำและการกระทำที่เราไม่อยากให้เกิดขึ้น ในการวิจัยนี้ผู้วิจัยต้องการให้ Agent มีพฤติกรรมหลัก ๆ 3 ข้อดังนี้ 1. Agent ไม่ควรเปิดคำสั่งซื้อหรือคำสั่งขายซ้ำกับออร์เดอร์ปัจจุบันที่ทำอยู่ 2. Agent ไม่ควรรอการเข้าซื้ออย่างเดียวจนไม่เกิดออร์เดอร์ใด ๆ หรือทำการเปิดออร์เดอร์และทำการถือยาวจนไม่มีการปิดออร์เดอร์ใด ๆ 3. Agent ไม่ควรทำการซื้อขายที่ก่อให้เกิดการขาดทุน ด้วยเหตุนี้ถ้าหากเราออกแบบฟังก์ชันการให้รางวัลได้ไม่ดีพอ พฤติกรรมของ Agent ก็จะไม่เป็นไปตามที่เราต้องการ ตัวอย่างเช่นหากเรากำหนดฟังก์ชันการให้รางวัลเป็นการคำนวณจากราคาที่แตกต่างเฉพาะตอนปิดออร์เดอร์เท่านั้นและการกระทำอื่น ๆ ให้เป็น 0 จะเกิดเหตุการณ์ดังแผนภาพที่ 3.6 Agent เกิดการซื้อ (Buy) และทำการปิดออร์เดอร์คำสั่งซื้อ (Exit) ที่ราคาต่ำกว่า ดังนั้นรางวัลที่ Agent ได้รับจะติดลบ เพราะคิดมาจากส่วนต่างของราคาซื้อและราคาออก ทำให้ Agent มองว่าการกระทำแบบนี้เป็นการกระทำที่ไม่ดี เพราะรางวัลที่ได้รับติดลบ ซึ่งสถานการณ์นี้ถือเป็นปกติเพราะเราต้องการให้ Agent ซื้อต่ำแล้วขายสูงดังนั้นการกำหนดรางวัลแบบนี้ถือว่าปกติ แต่ถ้าหาก Agent ทำการรอเข้า

ซื้อขาย (Flat) ก็จะเกิดปัญหาเพราะ Agent ก็จะเริ่มมองแล้วว่า จากพฤติกรรมก่อนหน้าที่มันทำการเข้าซื้อแล้วออกนั้นได้รางวัลติดลบ อย่างนี้ Agent จึงคิดว่ารอการเข้าซื้อขายอย่างเดียวยังดีเสียกว่าเพราะอย่างน้อยรางวัลก็ยังเป็น 0 ดังนั้นการกำหนดรางวัลอย่างนี้ก็จะส่งผลให้พฤติกรรมของ Agent ทำแต่การกระทำเหล่านี้และไม่เกิดการซื้อขายขึ้นซึ่งขัดกับสิ่งที่เราอยากได้ในข้อที่ 2 หรือ Agent เข้าทำการซื้อขายก็จริงแต่ไม่ออกจากรายการซื้อขายทำให้รางวัลก็เป็น 0 เช่นเดียวกัน



แผนภาพที่ 3.6 ตัวอย่างฟังก์ชันการให้รางวัลที่ส่งผลให้พฤติกรรมของ Agent ไม่ตรงตามความต้องการ

ดังนั้นเพื่อให้ Agent สามารถทำงานได้ตรงตามความต้องการ ผู้วิจัยจึงได้ออกแบบฟังก์ชันการให้รางวัลใหม่ซึ่งสามารถแสดงได้ดังแผนภาพที่ 3.7 โดยสามารถแบ่งได้หลัก ๆ เป็น 4 เหตุการณ์ดังนี้

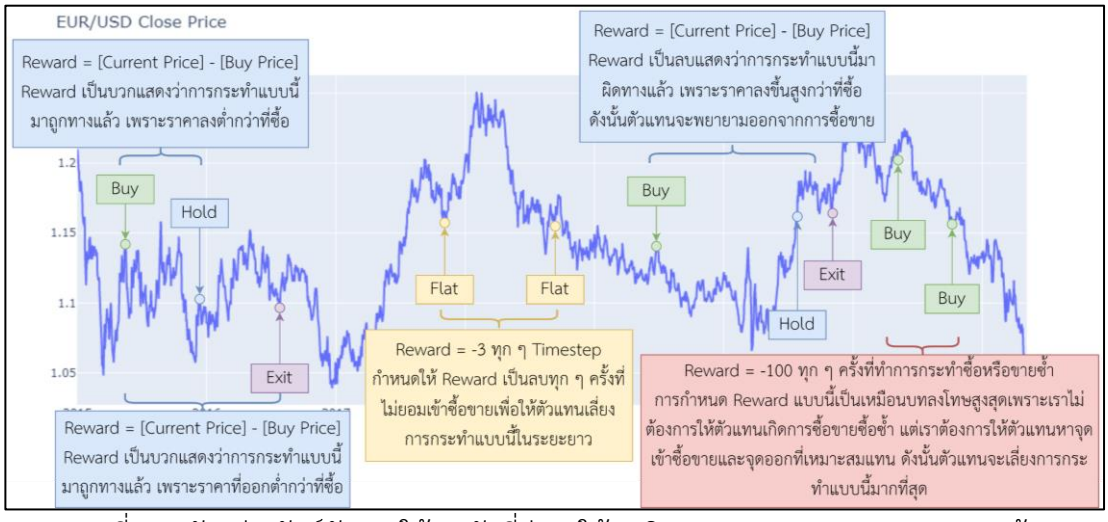
1. เมื่อ Agent มีการเข้าซื้อหรือมีการถือราคาต่อจากที่เข้าซื้อ จะทำการกำหนดรางวัลเป็นส่วนต่างของราคาที่เข้าซื้อและราคาปัจจุบันของทุก Timestep เนื่องจากถ้าเราไม่กำหนดแบบนี้ จะเกิดปัญหาที่เรียกว่า Delayed Reward กล่าวคือ Agent จะไม่รู้เลยว่าการกระทำที่ทำอยู่นั้นส่งผลดีมากน้อยเพียงใดต่ออนาคต เพราะจะรู้ตัวอีกทีก็ตอนที่ทำการปิดออร์เดอร์คำสั่งซื้อนั้น ๆ ด้วยเหตุนี้ถ้าหากรางวัลที่ได้รับล่าช้าเกินไปจะทำให้ Agent ของเราเรียนรู้ได้ยาก ดังนั้นเพื่อแก้ปัญหานี้จึงกำหนดเป็นราคาที่แตกต่างของราคาที่เข้าซื้อในทุก Timestep แทน เพื่อให้ Agent ได้รับ Feedback ตลอดเวลาว่า ณ ปัจจุบันที่ถืออยู่นั้น ราคายังคงกำไรหรือขาดทุนนั้นเอง

2. เมื่อ Agent มีการเข้าขายหรือมีการถือราคาต่อจากที่เข้าขาย จะทำการกำหนดรางวัลเป็นส่วนต่างของราคาที่เข้าขายและราคาปัจจุบันของทุก Timestep เช่นเดียวกันกับข้อที่ 1 เพื่อไม่ให้เกิดปัญหา Delayed Reward จะกำหนดในรูปแบบเดียวกัน

3. เมื่อ Agent ทำการรอเพื่อหาตำแหน่งเข้าซื้อหรือเข้าขาย เพื่อไม่ให้เกิดปัญหาจากที่กล่าวไปในตัวอย่างข้างต้นคือ Agent เลือกทำการรออย่างเดียวเพราะให้รางวัลเป็น 0 ดังนั้น

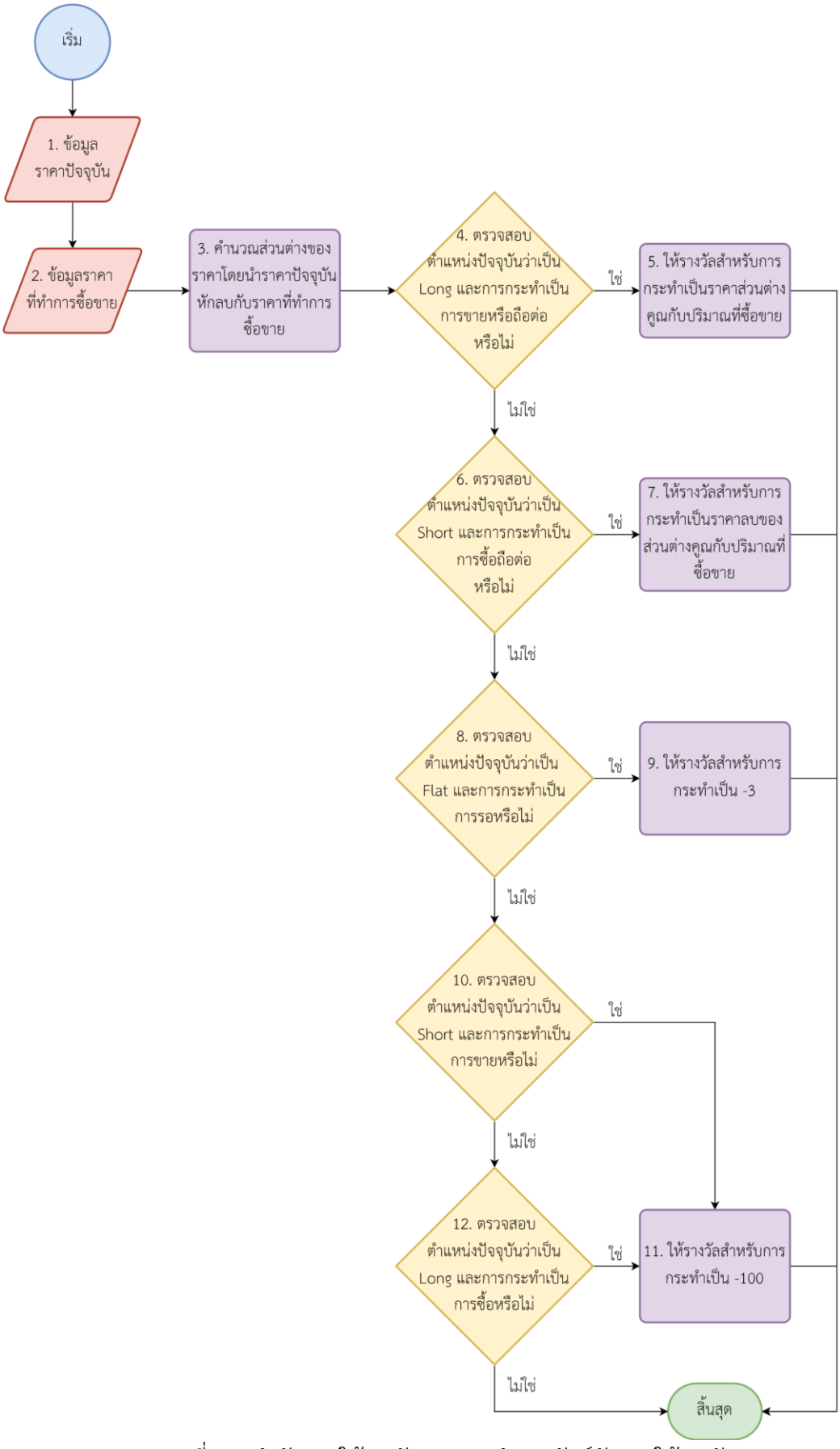
ผู้วิจัยจึงได้กำหนดรางวัลสำหรับการรอเป็น -3 ทุกครั้งที่ทำ เพื่อเป็นการกระตุ้น Agent เพียงเล็กน้อย ไม่ให้มากจนเกินไปจนพยายามเข้าซื้อหรือเข้าขายอย่างเดียวจนไม่สนใจการกระทำรอเข้าซื้อขาย หรือน้อยจนเกินไปจนทำการกระทำรอเข้าซื้อขายเพียงอย่างเดียว

4. เมื่อ Agent มีการเข้าซื้อซ้ำหรือเข้าขายซ้ำ ในการศึกษาครั้งนี้เราไม่มีกระบวนการในการเฉลี่ยราคา ดังนั้นเพื่อไม่ให้ราคาแรกที่ทำการเปิดออร์เดอร์เปลี่ยนไป จึงไม่อนุญาตให้เกิดการกระทำเหล่านี้ซ้ำ เปรียบเสมือนข้อห้ามดังนั้นผู้วิจัยจึงได้กำหนดรางวัลสำหรับการกระทำซ้ำเป็น -100 ทุกครั้งที่ทำ เพื่อเป็นการกระตุ้น Agent ให้เลี่ยงการกระทำแบบนี้มากที่สุดเท่าที่จะมากได้



แผนภาพที่ 3.7 ตัวอย่างฟังก์ชันการให้รางวัลที่ส่งผลให้พฤติกรรมของ Agent ตรงตามความต้องการ

จากเหตุการณทั้ง 4 เหตุการณ์สามารถแปลงออกมาให้อยู่ในรูปของผังงานได้ดังแผนภาพที่ 3.8



แผนภาพที่ 3.8 ลำดับการให้รางวัลการกระทำของฟังก์ชันการให้รางวัล

จากแผนภาพที่ 3.8 เมื่อมีการเทขาย (Sell) และตำแหน่งปัจจุบันเป็นการถือระยะยาว (Long) หมายความว่าราคาที่เราซื้อควรน้อยกว่าราคาที่เราทำการขาย ดังนั้นรางวัลจึงเป็นราคาที่แตกต่างระหว่างราคาปัจจุบันกับราคาที่ทำการซื้อคูณกับปริมาณที่ทำการซื้อ ในทางกลับกันเมื่อมีการเทซื้อ (Buy) และตำแหน่งปัจจุบันเป็นการถือระยะสั้น (Short) หมายความว่าราคาที่เราขายควรน้อยกว่าราคาที่เราทำการซื้อ ดังนั้นรางวัลจึงเป็นราคาที่ติดลบระหว่างราคาปัจจุบันกับราคาที่ทำการขายคูณกันปริมาณที่ทำการขาย

แต่ในกรณีที่มีการซื้อขายและมีการถือต่อเพื่อไม่ให้เกิดปัญหารางวัลล่าช้า (Delayed Reward) หรือก็คือ Agent ไม่ทราบว่าการกระทำปัจจุบันจะส่งผลต่อรางวัลสะสมในอนาคตอย่างไร จึงเพิ่มการคำนวณรางวัลให้กับการถือต่อไม่ว่าจะเป็นตำแหน่งเป็นการถือระยะยาว (Long) หรือระยะสั้น (Short) เป็นราคาที่เปลี่ยนแปลงในแต่ละช่วงเวลาเพื่อให้ Agent รับรู้อยู่ตลอดเวลา

ในส่วนของการตำแหน่งรอเข้าซื้อขาย (Flat) จะมีการกำหนดรางวัลเป็น -3 เนื่องจากเราไม่ต้องการให้ Agent ทำการกระทำแบบนี้มากจนเกินไป จนไม่เกิดการซื้อขายเลย เพราะในช่วงแรก Agent อาจจะเกิดการขาดทุนรางวัลจึงเป็นลบ ดังนั้นเพื่อไม่ให้ Agent มองว่าการรอเข้าซื้อขายได้รางวัลที่ดีกว่าจึงมีการกำหนดรางวัลของการรอเข้าซื้อขายเป็นลบเช่นกัน

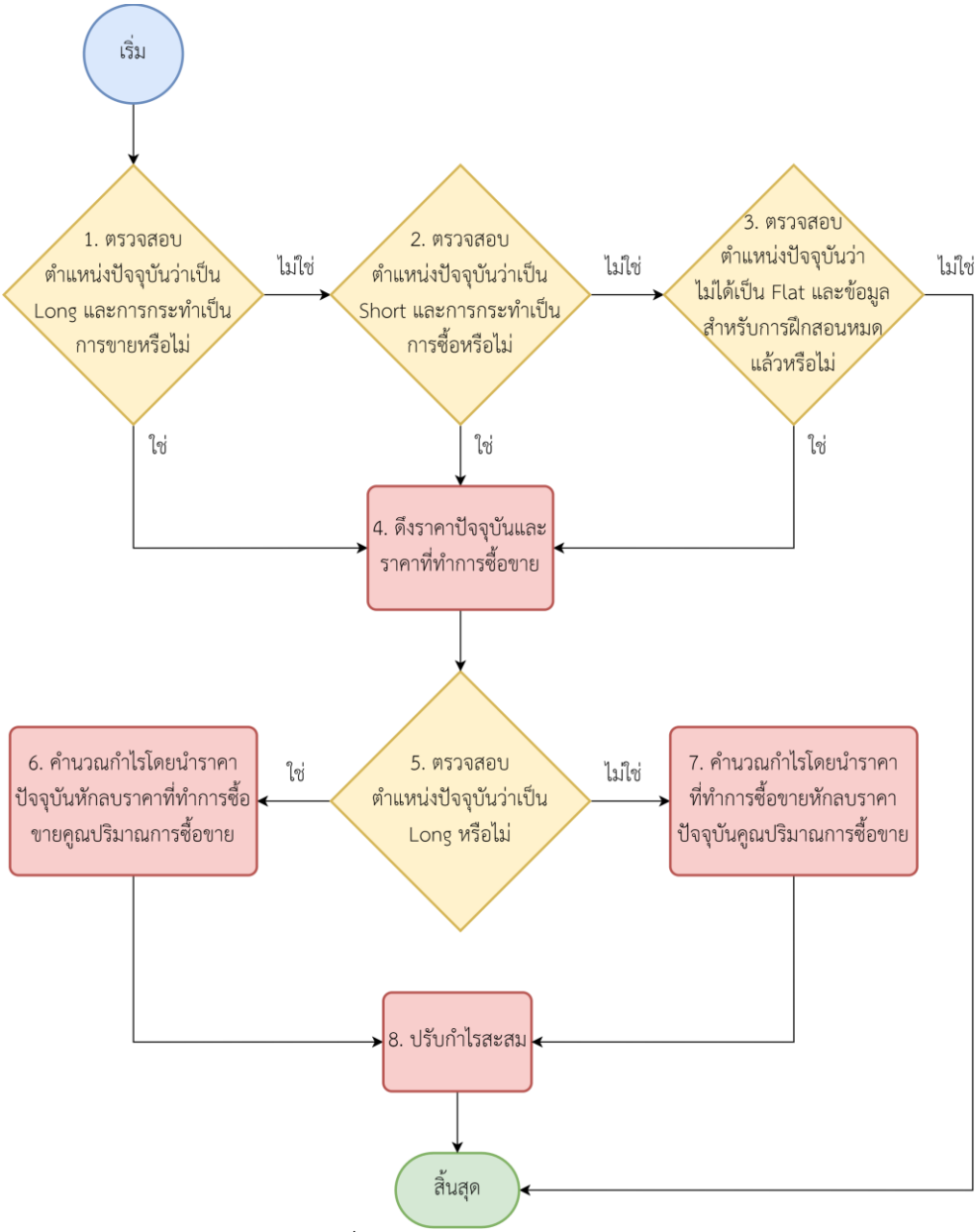
ในส่วนสุดท้ายจะเป็นส่วนที่เป็นการกระทำต้องห้ามที่สุดที่ไม่ต้องการให้ Agent ทำการกระทำเหล่านี้ก็คือเราไม่ต้องการให้ Agent ทำการเข้าถือระยะยาว (Long) ซ้ำหรือเข้าถือระยะสั้น (Short) ซ้ำจนกว่าจะมีการออกจากตำแหน่งเก่าเสียก่อน จึงกำหนดรางวัลให้เป็น -100 ที่สูงกว่าการกระทำอื่น ๆ ที่กล่าวมาทั้งหมด เป็นเสมือนบทลงโทษที่สูงที่สุดดังนั้น Agent จะพยายามเลี่ยงการกระทำแบบนี้มากที่สุด สามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.9

```
# ราคาปัจจุบัน
current_price = self.prices[self._current_tick]
# ราคาที่ทำการ Short หรือ Long
last_trade_price = self.prices[self._last_trade_tick]
# คำนวณความต่างของราคา
price_diff = current_price - last_trade_price

if (action == Actions.Sell.value and self._position == Positions.Long) or \
    (action == Actions.Hold.value and self._position == Positions.Long):
    # ถ้า Long แสดงว่า current_price > last_trade_price ดังนั้นไม่ต้องใส่ -
    step_reward += price_diff * self.lot_size

elif (action == Actions.Buy.value and self._position == Positions.Short) or \
     (action == Actions.Hold.value and self._position == Positions.Short):
    # ถ้า Short แสดงว่า last_trade_price > current_price จะเกิดการติดลบ ดังนั้นต้องใส่ -price_diff
    step_reward += -(price_diff * self.lot_size)

elif self._position == Positions.Flat and action == Actions.Hold.value:
    step_reward -= 3.

elif (action == Actions.Sell.value and self._position == Positions.Short ) or \
    (action == Actions.Buy.value and self._position == Positions.Long):
    step_reward -= 100
```

แผนภาพที่ 3.9 เงื่อนไขการให้รางวัลการกระทำ

3.5.6 ปรับปรุงวิธีการคำนวณกำไรขาดทุน

เพื่อให้การคำนวณกำไรขาดทุนทำความเข้าใจได้ง่าย จึงปรับการคำนวณกำไรขาดทุนให้อยู่ในรูปของราคาที่แตกต่างระหว่างการซื้อขายคูณกับปริมาณการซื้อขาย และกำหนดสกุลเงินหลัก (Base Currency) เป็นฝั่งซ้ายเสมอโดยกระบวนการคำนวณสามารถเขียนเป็นลำดับได้ดังแผนภาพที่ 3.10 กล่าวคือหากสกุลเงินเป็น EUR/USD และมีราคาเป็น 1.15300 สกุลเงินหลักจะเป็น EUR และสกุลเงินรอง (Quote Currency) เป็น USD และแปลความได้เป็น 1 EUR สามารถทำการซื้อขาย USD ได้ในราคา 1.15300 ดังนั้นหากเราทำการซื้อ 10000 EUR/USD ในราคาที่ 1.15300 และทำการขาย 10000 EUR/USD ในราคาที่ 1.15310 เราจะทำกำไรได้ (1.15310 – 1.15300) * 10000 ซึ่งเท่ากับได้กำไร 1 USD เป็นต้น โดยสามารถนำมาเขียนเป็นเงื่อนไขได้ดังแผนภาพที่ 3.11



แผนภาพที่ 3.10 ลำดับการคำนวณกำไรขาดทุน

```python
def _update_profit(self, action):
    trade = False
    if ((action == Actions.Sell.value and self._position == Positions.Long) or
        (action == Actions.Buy.value and self._position == Positions.Short)):
        trade = True
    # ถ้าเกิดการ Trade หรือข้อมูลหมดแล้ว โดยที่ถ้าข้อมูลหมดแล้วตำแหน่งต้องไม่ใช่ Flat
    if trade or (self._done and self._position != Positions.Flat):
        current_price = self.prices[self._current_tick]
        last_trade_price = self.prices[self._last_trade_tick]
        # คำนวณกำไรที่ได้จากราคาที่แตกต่าง
        if self._position == Positions.Long:
            self._total_profit += (current_price - last_trade_price) * self.lot_size
        else:
            self._total_profit += (last_trade_price - current_price) * self.lot_size
```

แผนภาพที่ 3.11 เงื่อนไขการคำนวณกำไรขาดทุน

3.5.7 ปรับปรุงช่วงของรางวัลเพื่อเพิ่มประสิทธิภาพการเรียนรู้ของ Agent ให้มีความเสถียรมากยิ่งขึ้น

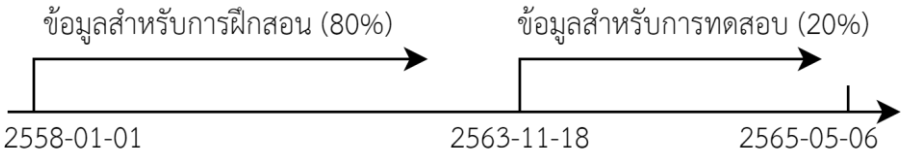
การที่เรากำหนดฟังก์ชันการให้รางวัลให้กับ Agent นั้นเป็นเหมือนกับการกำหนดพฤติกรรมอันควรจะเป็นของ Agent ซึ่งจะพบว่าค่าที่เรากำหนดมีช่วงที่หลากหลายตั้งแต่ค่าติดลบมากเมื่อการซื้อขายนั้นขาดทุนมากหรือค่าเป็นบวกมากเมื่อการซื้อขายทำกำไรได้มาก ทำให้ค่ารางวัลพวกนี้มีความแปรปรวนค่อนข้างสูงส่งผลให้ Agent เรียนรู้ได้ยากและมีความไม่เสถียรจากช่วงของรางวัลที่ไม่แน่นอน ดังนั้นเพื่อให้การฝึกสอน Agent มีประสิทธิภาพ และมีเสถียรภาพในการเรียนรู้มากยิ่งขึ้น จึงจำเป็นต้องนำรางวัลที่ได้ในแต่ละขั้นตอนนั้นมาปรับช่วงด้วยวิธีการ Welford กล่าวคือจะเป็นการปรับช่วงของรางวัลที่เกิดขึ้นในช่วงเวลาที่แตกต่างกันให้มีค่าเฉลี่ยเป็น 0 และมีส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation) เป็น 1 ซึ่งฟังก์ชันนี้ผู้พัฒนาสภาพแวดล้อม OpenAI Gym ได้มีการพัฒนาขึ้นมาให้พร้อมใช้งานแล้ว ดังนั้นเราสามารถนำฟังก์ชันนี้มาปรับใช้กับสภาพแวดล้อมที่เราสร้างขึ้นมาเองได้ทันที ดังแผนภาพที่ 3.12

```python
def create_environment(data, features, window_size=8):
    env = ForexEnv(df=data, window_size=window_size,
                   frame_bound=(window_size, len(data)),
                   unit_side = 'left', features=features)
    # Set Random Seed
    env.seed(RANDOM_SEED)
    env.action_space.seed(RANDOM_SEED)
    # Normalize Reward ให้มีค่าเฉลี่ยเป็น 0 และส่วนเบี่ยงเบนมาตรฐานเป็น 1
    env = NormalizeReward(env)
    # Wrapper เพื่อเก็บสถิติของแต่ละ Episode เช่น Return
    env = RecordEpisodeStatistics(env)
    return env
```

แผนภาพที่ 3.12 ฟังก์ชันสำหรับสร้างสภาพแวดล้อมที่มีการปรับปรุงช่วงของรางวัล

3.5.8 สร้างสภาพแวดล้อมสำหรับการฝึกสอนและสภาพแวดล้อมสำหรับการทดสอบ

เพื่อให้ Agent มีสภาพแวดล้อมสำหรับการทดสอบกับชุดข้อมูลที่ไม่เคยเห็นมาก่อน (Unseen Data) ดังนั้นจึงมีการแบ่งชุดข้อมูลออกเป็นชุดข้อมูลสำหรับฝึกสอนคิดเป็น 80% ของชุดข้อมูลตั้งแต่วันที่ 2558-01-01 ถึงวันที่ 2563-11-17 และชุดข้อมูลสำหรับทดสอบคิดเป็น 20% ของชุดข้อมูลตั้งแต่วันที่ 2563-11-18 ถึงวันที่ 2565-05-06 ดังแผนภาพที่ 3.13



แผนภาพที่ 3.13 ชุดข้อมูลสำหรับสภาพแวดล้อมฝึกสอนและชุดข้อมูลสำหรับสภาพแวดล้อมทดสอบ

3.6 รายละเอียดในการพัฒนา Agent ที่ใช้สำหรับการแก้ปัญหา

3.6.1 การพัฒนานโยบายสำหรับการเลือกการกระทำ

ในการพัฒนานโยบายสำหรับการเลือกการกระทำจะพัฒนาใน 2 รูปแบบคือ แบบที่หนึ่งวิธีการแบบละโมบจะเป็นการเลือกการกระทำที่ให้ค่าประมาณการ Q-value สูงที่สุดที่ได้จากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ (Online Network) ณ สถานะปัจจุบันที่รับเข้ามาดังอัลกอริทึมที่ 3.1 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.14

ALGORITHM 3.1: GREEDY ACTION SELECTION

Input: *S: current state, N: Online Neural Network*
Output: *Selected Action*
Function: *SELECT-ACTION(S, N)* ***is***
1 $Q \leftarrow$ *Estimate Q-value using* $N(S)$
2 $A \leftarrow argmax Q$
3 ***return*** *selected action A*
4 ***end***

```
def greedy(state, net):
    state = torch.tensor([state]).to(device)
    q_values = net(state)
    action = q_values.argmax(dim=-1)
    action = int(action.item())

    return action
```

แผนภาพที่ 3.14 วิธีการเลือกการกระทำแบบละโมบ

แบบที่สองจะเป็นวิธีการแบบละโมบด้วยค่า Epsilon จะประกอบด้วย 2 รูปแบบการกระทำคือ 1. หากเราสุ่มค่าขึ้นมาแล้วน้อยกว่าค่า Epsilon เราจะทำการสุ่ม Action แทน จะถือว่าเป็นการสำรวจและ 2. ถ้าหากเราสุ่มค่าขึ้นมาแล้วมากกว่าค่า Epsilon เราจะทำการเลือกเลือกการกระทำที่ให้ค่าประมาณการ Q-value สูงที่สุดที่ได้จากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ดังอัลกอริทึมที่ 3.2 ซึ่งค่า Epsilon จะถูกลดลงตามช่วงเวลาที่เรียนรู้เพราะในช่วงแรกที่เรายังไม่เกิดการเรียนรู้เราต้องมีการสำรวจมากแต่เมื่อมีการเรียนรู้เพิ่มขึ้นเราก็จะใช้ความรู้ที่มีมากขึ้นดังนั้นค่า Epsilon ก็จะลดลงตามช่วงเวลา และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.15

ALGORITHM 3.2: EPSILON-GREEDY ACTION SELECTION

Input: *S: current state, N: Online Neural Network, E: Environment, ε: Epsilon*
Output: *Selected Action*
Function: *SELECT-ACTION(S, N, E, ε)* ***is***
    n ← *uniform random number between 0 and 1*
    if *n* < *ε* ***then***
        A ← *random action from the action space from E*
    else
        Q ← *Estimate Q-value using N(S)*
        A ← *argmaxQ*
    end
    return *selected action A*
end

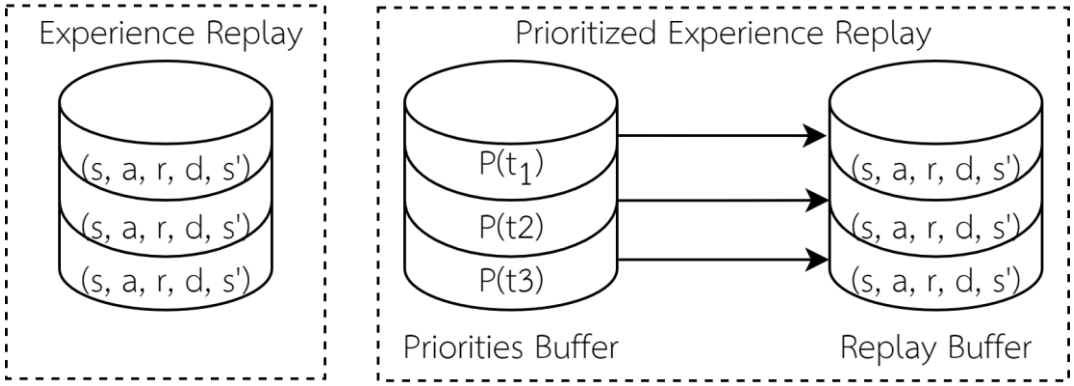
```python
def epsilon_greedy(state, net, env, epsilon=0.0):
    n = np.random.random()
    # ถ้า random แล้วน้อยกว่า Epsilon ให้สุ่ม action
    if n < epsilon:
        action = env.action_space.sample()
    else:
        state = torch.tensor([state]).to(device)
        q_values = net(state)
        action = q_values.argmax(dim=-1)
        action = int(action.item())

    return action
```

แผนภาพที่ 3.15 วิธีการเลือกการกระทำแบบละโมบด้วยค่า Epsilon

3.6.2 การพัฒนาถังเก็บข้อมูลประสบการณ์

ในการพัฒนาส่วนถังเก็บข้อมูลประสบการณ์ เราสามารถพัฒนารูปแบบการเก็บข้อมูลประสบการณ์ได้ 2 แบบคือ 1. ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) และ 2. ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay) ซึ่งสิ่งที่

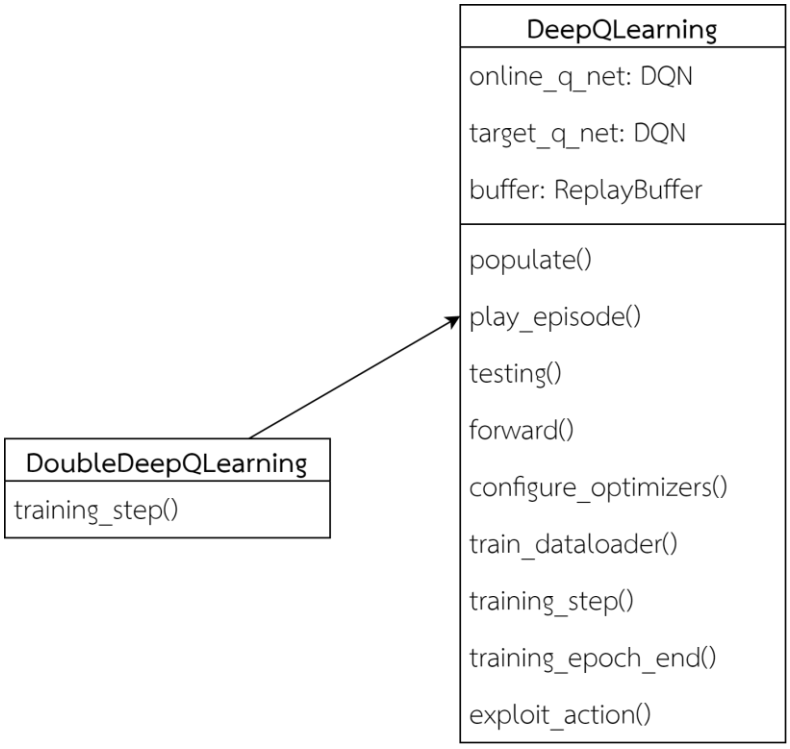
แตกต่างระหว่างทั้ง 2 แบบคือการเก็บข้อมูลประสบการณ์ที่มีการจัดความสำคัญจะใช้ Deque อีกตัวในการเก็บความสำคัญ (Priority) ของแต่ละประสบการณ์นั่นเอง ดังแผนภาพที่ 3.16



แผนภาพที่ 3.16 ความแตกต่างระหว่าง Experience Replay และ Prioritized Experience Replay

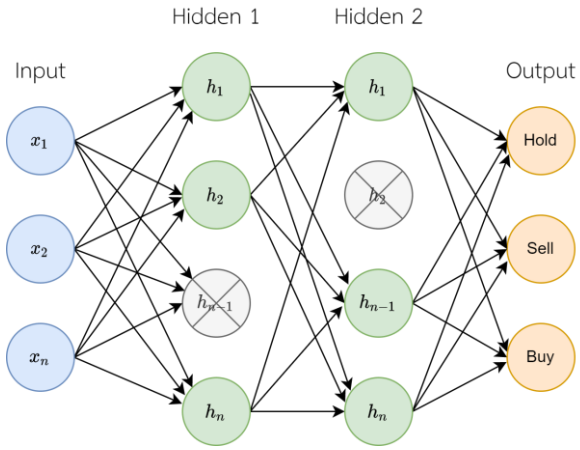
3.6.3 การพัฒนา Agent ประเภท Double Deep Q-Learning (DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay)

ในการพัฒนา Agent ประเภท Double Deep Q-Learning เราจะทำการเขียนส่วนประกอบต่าง ๆ ไม่ว่าจะเป็นส่วนสำหรับเก็บข้อมูลประสบการณ์ โครงข่ายประสาทเทียมสำหรับการเรียนรู้ โครงข่ายประสาทเทียมสำหรับการประเมิน ฟังก์ชันสำหรับการเรียนรู้ การติดตามประสิทธิภาพการเรียนรู้ต่าง ๆ ในรูปแบบของ Object ซึ่งเป็นคุณสมบัติหลักของ Agent ประเภท Deep Q-Learning จากนั้นเราถึงสืบทอดคุณสมบัติหลักมาจาก Agent ประเภท Deep Q-Learning เพื่อนำมาแก้ไขเพียงบางส่วน (Overriding) โดยสิ่งที่ต้องปรับแก้ไขคือส่วนของการคำนวณค่าประมาณ Q ดังที่กล่าวไปในบทที่ 2.7.3 สามารถแสดงได้ดังแผนภาพที่ 3.17



แผนภาพที่ 3.17 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Double Deep Q-Learning

ในส่วนของโครงข่ายประสาทเทียมสำหรับการเรียนรู้การกระทำที่เหมาะสมนั้นได้ใช้โครงข่ายประสาทเทียมที่มี Hidden Layer 2 ชั้นและมีการ Dropout Node ทิ้ง 20% เพื่อแก้ปัญหา Overfitting และมี Input รับข้อมูลเข้าตามขนาดของจำนวนวันย้อนหลัง จำนวนตัวชี้วัดเชิงเทคนิค และตำแหน่งปัจจุบัน และมี Output เป็น 3 Node คือ 1. Hold 2. Sell และ 3. Buy และมีการกำหนดฟังก์ชันกระตุ้นเป็น ReLU ดังแผนภาพที่ 3.18 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.19



แผนภาพที่ 3.18 โครงสร้างโครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent ประเภท Double Deep Q-Learning

```python
class DQN(nn.Module):

    def __init__(self, hidden_size, obs_size, n_actions):
        super(DQN, self).__init__()

        self.obs_size = obs_size

        self.net = nn.Sequential (
            nn.Linear(obs_size, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(p=0.2),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(p=0.2),
            # Predict Q-value ของแต่ละ state-action
            nn.Linear(hidden_size, n_actions)
        )

    def forward(self, x):
        return self.net(x.float())
```

แผนภาพที่ 3.19 โครงข่ายประสาทเทียมสำหรับการเรียนรู้ประเภท Deep Q-Network

ในส่วนของโครงข่ายประสาทเทียมสำหรับการประเมินการกระทำที่เหมาะสมนั้นจะใช้โครงสร้างเดียวกันกับโครงข่ายประสาทเทียมสำหรับการเรียนรู้ ดังนั้นเราสามารถคัดลอกโครงสร้างและน้ำหนักพารามิเตอร์เริ่มต้นของโครงข่ายประสาทเทียมออกมาเป็นอีกตัวได้

Agent ประเภท Double Deep Q-Learning นั้นต้องมีกระบวนการในการสำรวจสภาพแวดล้อมเพื่อเก็บรวบรวมประสบการณ์ในช่วงแรกดังนั้นนโยบายสำหรับการเลือกการกระทำจะเป็นการใช้วิธีการแบบละโมบด้วยค่า Epsilon และปรับปรุงค่า Epsilon ให้ลดลงตามจำนวนรอบที่ทำการเรียนรู้ (Epoch) และติดตามค่า Epsilon ลงบน Tensorboard ดังแผนภาพที่ 3.20

```python
def training_epoch_end(self, training_step_outputs):
    # หลังจบแต่ละ Epoch เราจะลดจำนวน Exploration (Epsilon)
    epsilon = max(
        self.hparams.eps_end,
        self.hparams.eps_start - (self.current_epoch / self.hparams.eps_last_episode)
    )
    self.log('episode/Epsilon', epsilon)
```

แผนภาพที่ 3.20 แสดงถึงวิธีการปรับปรุงค่า Epsilon ให้ลดลงตามจำนวนรอบการเรียนรู้

การฝึกสอน Agent นั้น จำเป็นต้องมีฟังก์ชันสำหรับการตอบโต้กับสภาพแวดล้อมในการเก็บข้อมูลประสบการณ์สำหรับการเรียนรู้ ซึ่งการตอบโต้กับสภาพแวดล้อมจะทำตามนโยบายแบบละโมบด้วยค่า Epsilon และเก็บข้อมูลประสบการณ์เหล่านั้นลง Experience Replay โดยกระบวนการตอบโต้ก็จะเหมือนกับกระบวนการของ Markov Decision Process คือ Agent รับสถานะปัจจุบันมาจากสภาพแวดล้อม จากนั้น Agent จะนำสถานะนั้นไปพิจารณาและทำตามนโยบายปัจจุบันเพื่อเลือกการกระทำออกมา และนำการกระทำนั้นไปตอบโต้กับสภาพแวดล้อมเพื่อ

รับรางวัลและสถานะถัดไป ซึ่งเราจะเก็บลำดับการทำงานเหล่านี้ลงใน Experience Replay ดังแผนภาพที่ 3.21

```python
@torch.no_grad()
def play_episode(self, policy=None, epsilon=0.):
    state = self.env.reset()
    done = False

    while not done:
        # ในกรณีส่ง Policy มาเช่น Epsilon Greedy
        if policy:
            action = policy(state, self.online_q_net, self.env, epsilon=epsilon)

        # ถ้าไม่ก็ Random Action เอา มีไว้เพื่อให้ช่วงแรกมีการสุ่ม Action ให้มากที่สุดในการเก็บเข้า ReplayBuffer
        else:
            action = self.env.action_space.sample()

        next_state, reward, done, info = self.env.step(action)

        # แปลงให้เป็น Experience ในรูปของ Tuple
        exp = (state, action, reward, done, next_state)

        # เก็บ Experience เข้า ReplayBuffer
        self.buffer.append(exp)

        state = next_state
```

แผนภาพที่ 3.21 ฟังก์ชันสำหรับการตอบโต้กับสภาพแวดล้อมในการเก็บข้อมูลประสบการณ์

กระบวนการเรียนรู้ของ Agent ประเภท Double Deep Q-Learning นั้นจะเป็นการเรียนรู้แบบ Mini-batch จากชุดประสบการณ์ที่เก็บสะสมไว้ และนำมาคำนวณหาค่าสูญเสียดังสมการ 2.22 และนำค่าส่วนนั้นกลับไปปรับปรุงโครงข่ายประสาทเทียมสำหรับการเรียนรู้ สามารถเขียนได้ดังแผนภาพที่ 3.22

```python
def training_step(self, batch, batch_idx):
    # Experience ที่ได้จาก Mini-batch
    states, actions, rewards, dones, next_states = batch
    actions = actions.unsqueeze(1)
    rewards = rewards.unsqueeze(1)
    dones = dones.unsqueeze(1)

    # Estimate Q-value จาก Online Network
    state_action_values = self.online_q_net(states).gather(1, actions)

    with torch.no_grad():
        # เอา action ที่ให้ค่า Q-value มากที่สุดมาจาก online network
        _, next_actions = self.online_q_net(next_states).max(dim=1, keepdim=True)
        # จากนั้นใช้ค่า Q-value ของ target network โดยใช้ action ที่ดีที่สุดที่ได้จาก online network
        next_action_values = self.target_q_net(next_states).gather(1, next_actions)
        # ในกรณีที่เป็น Terminal state จะให้ค่า Q-value เป็น 0
        next_action_values[dones] = 0.0

    # คำนวณตามสมการ Target Q-value
    expected_state_action_values = rewards + self.hparams.gamma * next_action_values

    # คำนวณค่าสูญเสีย ([Q-value] - [Target Q-value]) ^ 2
    loss = self.hparams.loss_fn(state_action_values, expected_state_action_values)
    self.log('episode/Q-Error', loss)
    return loss
```

แผนภาพที่ 3.22 วิธีการเรียนรู้โครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent

เมื่อ Agent ผ่านการเรียนรู้มาในระดับหนึ่ง เพื่อไม่ให้โครงข่ายประสาทเทียมสำหรับการประเมินนั้นเก่าเกินไป จำเป็นต้องมีการคัดลอกน้ำหนักพารามิเตอร์จากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ไปใส่ในโครงข่ายประสาทเทียมสำหรับการประเมินเป็นรอบ ๆ เพื่อให้การเรียนรู้ล่าสุดมากยิ่งขึ้นดังแผนภาพที่ 3.23

```
def training_epoch_end(self, training_step_outputs):
    # อัพเดท Weight ของ Target Q-Network ทุก sync_rate รอบ ในที่นี้คือ 10 รอบ (Default)
    if self.current_epoch % self.hparams.sync_rate == 0:
        self.target_q_net.load_state_dict(self.online_q_net.state_dict())
```

แผนภาพที่ 3.23 แสดงถึงวิธีการคัดลอกน้ำหนักพารามิเตอร์จากโครงข่ายประสาทเทียมสำหรับการเรียนรู้ไปใส่ในโครงข่ายประสาทเทียมสำหรับการประเมิน

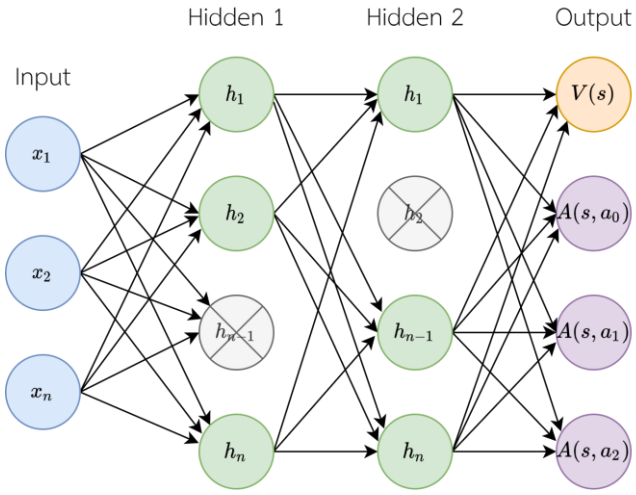
ส่วนสุดท้ายจะเป็นส่วนของการทดสอบประสิทธิภาพของ Agent กับสภาพแวดล้อมสำหรับการทดสอบ กล่าวคือทดสอบกับชุดข้อมูลที่ไม่เคยเห็นมาก่อนว่ามีประสิทธิภาพในการซื้อขายอย่างไร ซึ่งกระบวนการตอบโต้ก็จะเหมือนกับฟังก์ชัน play_episode() ที่กล่าวไปในข้างต้น แต่สิ่งที่แตกต่างคือจะใช้นโยบายละโมบในการเลือกการกระทำ ดังแผนภาพที่ 3.24

```
@torch.no_grad()
def testing(self, net):
    env = create_environment(data=test_df, features=features,
                             window_size=self.window_size)
    state = env.reset()
    done = False
    while not done:
        action = greedy(state, net)
        next_state, _, done, info = env.step(action)
        state = next_state
        if done:
            break
    return info['total_reward'], info['total_profit']
```

แผนภาพที่ 3.24 แสดงถึงวิธีการทดสอบของ Agent กับสภาพแวดล้อมสำหรับการทดสอบ

3.6.4 การพัฒนา Agent ประเภท Dueling Double Deep Q-Learning (Dueling DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay)

ในการพัฒนา Agent ประเภท Dueling Double Deep Q-Learning นั้นจะถูกพัฒนาต่อยอดมาจาก Agent ประเภท Double Deep Q-Learning ซึ่งส่วนที่แตกต่างคือส่วนของโครงสร้างของโครงข่ายประสาทเทียมที่มีการปรับโครงสร้างออกเป็น 2 สายคือ สาย State Value และสาย Advantage Value และมีกระบวนการรวมค่าเป็น Q-value จึงเรียกว่า Dueling ดังแผนภาพที่ 3.25 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.26

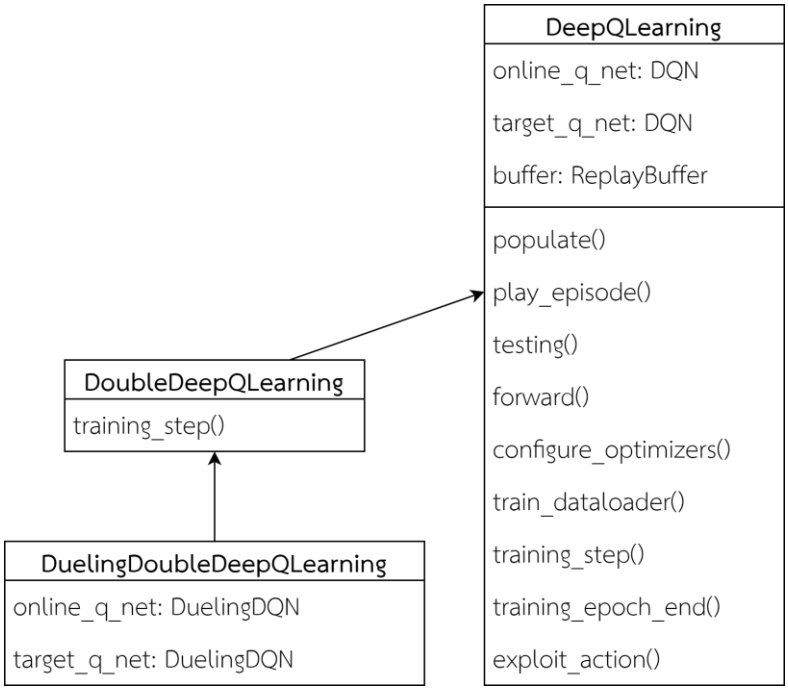


แผนภาพที่ 3.25 โครงสร้างโครงข่ายประสาทเทียมสำหรับการเรียนรู้ของ Agent ประเภท Dueling Double Deep Q-Learning

```python
class DuelingDQN(nn.Module):

    def __init__(self, hidden_size, obs_size, n_actions):
        super().__init__()

        self.obs_size = obs_size

        # FNN ที่แชร์Network ร่วมกัน
        self.net = nn.Sequential(
            nn.Linear(obs_size, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(p=0.2),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(p=0.2),
        )

        # Q(s, a) = V(s) + Adv(s, a)
        self.fc_value = nn.Linear(hidden_size, 1) # Value Stream
        self.fc_adv = nn.Linear(hidden_size, n_actions) # Advantage Stream

    def forward(self, x):
        x = self.net(x.float())
        value = self.fc_value(x) # V(s)
        adv = self.fc_adv(x) # A(s, a)
        return value + adv - torch.mean(adv, dim=1, keepdim=True)
```

แผนภาพที่ 3.26 โครงข่ายประสาทเทียมสำหรับการเรียนรู้ประเภท Dueling Deep Q-Network

จากนั้นเราสามารถสืบทอดคุณสมบัติมาจาก Agent ประเภท Double Deep Q-Learning ได้เลย โดยเปลี่ยนแค่โครงสร้างของโครงข่ายประสาทเทียมดังแผนภาพที่ 3.27 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.28



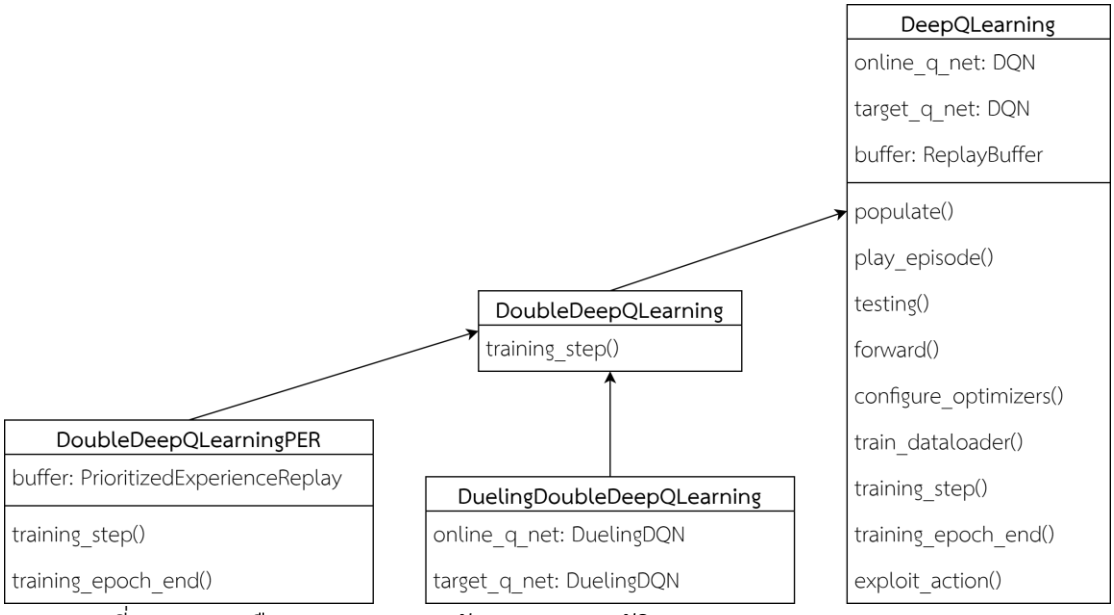
แผนภาพที่ 3.27 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Dueling Double Deep Q-Learning

```
class DuelingDoubleDeepQLearning(DoubleDeepQLearning):

    def __init__(self, policy=epsilon_greedy, buffer=ReplayBuffer,
                 capacity=100_000, batch_size=256, lr=1e-3,
                 hidden_size=128, gamma=0.99, network=DuelingDQN,
                 loss_fn=F.smooth_l1_loss, optim=AdamW,
                 eps_start=1.0, eps_end=0.15, eps_last_episode=150,
                 samples_per_epoch=1_000, sync_rate=10, window_size=8):

        super().__init__(policy=policy, buffer=buffer, capacity=capacity,
                         batch_size=batch_size, lr=lr, hidden_size=hidden_size,
                         gamma=gamma, network=network, loss_fn=loss_fn,
                         optim=optim, eps_start=eps_start, eps_end=eps_end,
                         eps_last_episode=eps_last_episode, window_size=window_size,
                         samples_per_epoch=samples_per_epoch, sync_rate=sync_rate)
```

แผนภาพที่ 3.28 การสร้าง Dueling Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนโครงข่ายประสาทเทียมเป็นประเภท Dueling

3.6.5 การพัฒนา Agent ประเภท Double Deep Q-Learning (DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay)

ในการพัฒนา Agent ประเภท Double Deep Q-Learning โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญนั้นเราสามารถนำ Double Deep Q-Learning เป็นตัวตั้งต้นแล้วสืบทอดคุณสมบัติมาได้เลย ซึ่งส่วนที่แตกต่างคือใช้ Prioritized Experience Replay และมีการกำหนดค่าพารามิเตอร์ Alpha และ Beta ดังแผนภาพที่ 3.29 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.30



แผนภาพที่ 3.29 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Double Deep Q-Learning ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ

```
class DoubleDeepQLearningPER(DoubleDeepQLearning):

    def __init__(self, policy=epsilon_greedy, buffer=PrioritizedExperienceReplay,
                 capacity=100_000, batch_size=256, lr=1e-3, hidden_size=128,
                 gamma=0.99, network=DQN, loss_fn=F.smooth_l1_loss, optim=AdamW,
                 eps_start=1.0, eps_end=0.15, eps_last_episode=150,
                 samples_per_epoch=1_000, sync_rate=10, window_size=8,
                 alpha_start=1.0, alpha_end=0.0, alpha_last_episode=150,
                 beta_start=0.5, beta_end=1.0, beta_last_episode=150):

        super().__init__(policy=policy, buffer=buffer, capacity=capacity,
                         batch_size=batch_size, lr=lr, hidden_size=hidden_size,
                         gamma=gamma, network=network, loss_fn=loss_fn, optim=optim,
                         eps_start=eps_start, eps_end=eps_end,
                         eps_last_episode=eps_last_episode, window_size=window_size,
                         samples_per_epoch=samples_per_epoch, sync_rate=sync_rate)
```

แผนภาพที่ 3.30 การสร้าง Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนถังเก็บข้อมูลเป็นประเภทถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ

เมื่อมีการเปลี่ยนถังเก็บข้อมูลประสบการณ์ ดังนั้นกระบวนการฝึกสอนก็ต้องมีการเปลี่ยนแปลง โดยกระบวนการเรียนรู้จะมีการนำค่า Weight เข้ามาใช้ในการคำนวณค่าสูญเสีย และมีกระบวนการอัพเดทค่าความสำคัญ ดังแผนภาพที่ 3.31

```python
def training_step(self, batch, batch_idx):
    # sample จาก PER ที่ถูก unpack experience ออกมาด้วย
    indices, weights, states, actions, rewards, dones, next_states = batch
    weights = weights.unsqueeze(1)
    actions = actions.unsqueeze(1)
    rewards = rewards.unsqueeze(1)
    dones = dones.unsqueeze(1)
    state_action_values = self.online_q_net(states).gather(1, actions)
    with torch.no_grad():
        _, next_actions = self.online_q_net(next_states).max(dim=1, keepdim=True)
        next_action_values = self.target_q_net(next_states).gather(1, next_actions)
        next_action_values[dones] = 0.0
    expected_state_action_values = rewards + self.hparams.gamma * next_action_values

    # คำนวณ TD Error แล้ว Take Absolute ตามสมการเพื่อนำไปคำนวณ Priorities
    td_errors = (state_action_values - expected_state_action_values).abs().detach()

    # นำ TD Error ของแต่ละ sample กลับไปอัพเดท
    for idx, td_error in zip(indices, td_errors):
        self.buffer.update(idx, td_error.cpu().item())

    # คำนวณ weighted loss function โดยนำ weight มาคูณผลลัพธ์จาก loss function
    loss = weights * self.hparams.loss_fn(state_action_values, expected_state_action_values,
                                          reduction='none')
    loss = loss.mean()
    self.log('episode/Q-Error', loss)
    return loss
```

แผนภาพที่ 3.31 วิธีการอัพเดทค่าความสำคัญของประสบการณ์และการคำนวณค่าสูญเสียสำหรับการเรียนรู้

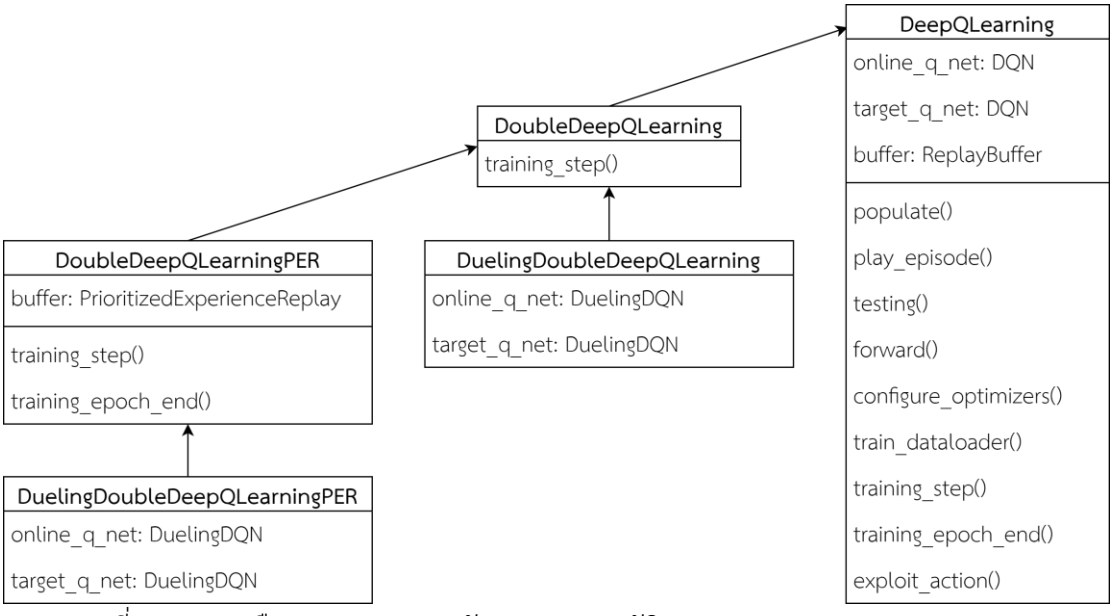
อีกทั้งต้องมีการปรับปรุงค่า Alpha ลงและปรับปรุงค่า Beta ขึ้นตามทฤษฎีหลังมีการเรียนรู้ตามรอบแบบเดียวกับการปรับค่า Epsilon ดังแผนภาพที่ 3.32

```python
def training_epoch_end(self, training_step_outputs):
    epsilon = max(
        self.hparams.eps_end,
        self.hparams.eps_start - self.current_epoch / self.hparams.eps_last_episode
    )
    self.log('episode/Epsilon', epsilon)

    # คำนวณค่า alpha
    alpha = max(
        self.hparams.alpha_end,
        self.hparams.alpha_start - self.current_epoch / self.hparams.alpha_last_episode
    )
    self.log('episode/Alpha', alpha)

    # คำนวณค่า beta
    beta = min(
        self.hparams.beta_end,
        self.hparams.beta_start + self.current_epoch / self.hparams.beta_last_episode
    )
    self.log('episode/Beta', beta)

    # อัพเดทค่า alpha และ beta ของ buffer
    self.buffer.alpha = alpha
    self.buffer.beta = beta
```

แผนภาพที่ 3.32 วิธีการปรับปรุงค่า Alpha และ Beta ตามรอบการเรียนรู้

3.6.6 การพัฒนา Agent ประเภท Dueling Double Deep Q-Learning (Dueling DDQN) โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay)

ในการพัฒนา Agent ประเภท Dueling Double Deep Q-Learning โดยใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญนั้นเราสามารถนำ Double Deep Q-Learning ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญเป็นตัวตั้งต้นแล้วสืบทอดคุณสมบัติมาได้เลย ซึ่งส่วนที่แตกต่างคือส่วนของโครงสร้างของโครงข่ายประสาทเทียมที่ใช้เป็นประเภท Dueling Deep Q-Network ดังแผนภาพที่ 3.33 และสามารถนำมาเขียนเป็นโปรแกรมได้ดังแผนภาพที่ 3.34



แผนภาพที่ 3.33 การสืบทอดและการปรับปรุงคุณสมบัติของ Agent ประเภท Dueling Double Deep Q-Learning ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ

```
class DuelingDoubleDeepQLearningPER(DoubleDeepQLearningPER):

    def __init__(self, policy=epsilon_greedy, buffer=PrioritizedExperienceReplay,
                 capacity=100_000, batch_size=256, lr=1e-3, hidden_size=128,
                 gamma=0.99, network=DuelingDQN, loss_fn=F.smooth_l1_loss,
                 optim=AdamW, eps_start=1.0, eps_end=0.15, eps_last_episode=150,
                 samples_per_epoch=1_000, sync_rate=10, window_size=8,
                 alpha_start=1.0, alpha_end=0.0, alpha_last_episode=150,
                 beta_start=0.5, beta_end=1.0, beta_last_episode=150):

        super().__init__(policy=policy, buffer=buffer, capacity=capacity,
                         batch_size=batch_size, lr=lr, hidden_size=hidden_size,
                         gamma=gamma, network=network, loss_fn=loss_fn, optim=optim,
                         eps_start=eps_start, eps_end=eps_end,
                         eps_last_episode=eps_last_episode, window_size=window_size,
                         samples_per_epoch=samples_per_epoch, sync_rate=sync_rate)
```

แผนภาพที่ 3.34 การสร้าง Dueling Double Deep Q-Learning จากการสืบทอดคุณสมบัติและเปลี่ยนถังเก็บข้อมูลเป็นประเภทถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ

บทที่ 4
ผลการวิจัยและการอภิปรายผล

การเปรียบเทียบประสิทธิภาพของการซื้อขายคู่สกุลเงินระหว่าง Agent ที่ใช้อัลกอริทึมที่แตกต่างกันและมีค่าพารามิเตอร์ที่ถูกควบคุมเหมือนกัน โดยใช้ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ที่มีการจัดเตรียมและประมวลผลค่าชี้วัดเชิงเทคนิคเรียบร้อยแล้วมาฝึกสอน ทดสอบและเปรียบเทียบประสิทธิภาพการทำงานระหว่าง Agent โดยทุกการฝึกสอน การทดสอบและการเปรียบเทียบจะมีการตั้งค่า Seed เพื่อให้การทดลองสามารถทำซ้ำได้และได้ผลลัพธ์การทดลองที่คงเดิม

ในบทนี้จะพูดถึงการทดลองและผลลัพธ์ที่ได้จากการทดลองซึ่งประกอบไปด้วย 3 การทดลอง คือ 1. การทดลองเพื่อเปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับปริมาณข้อมูลในการสุ่มมาเรียนรู้ (Batch Size) 2. การทดลองเพื่อเปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับอัลกอริทึมในการเรียนรู้ (Optimizer) และอัตราการเรียนรู้ (Learning Rate) และ 3. การทดลองเพื่อเปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับการให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคต (Gamma)

4.1 การทดลองที่ 1 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับปริมาณข้อมูลในการสุ่มมาเรียนรู้ (Batch Size)

ในการทดลองนี้จะมีการแบ่งเป็น 3 การทดลองย่อย ซึ่งมีการกำหนดขนาดของ Batch Size ของแต่ละการทดลองคือ 64 128 และ 256 ตามลำดับ และมีการกำหนดค่าให้กับพารามิเตอร์ดังนี้ Gamma มีค่าเป็น 0.7 และ Agent ใช้อัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001

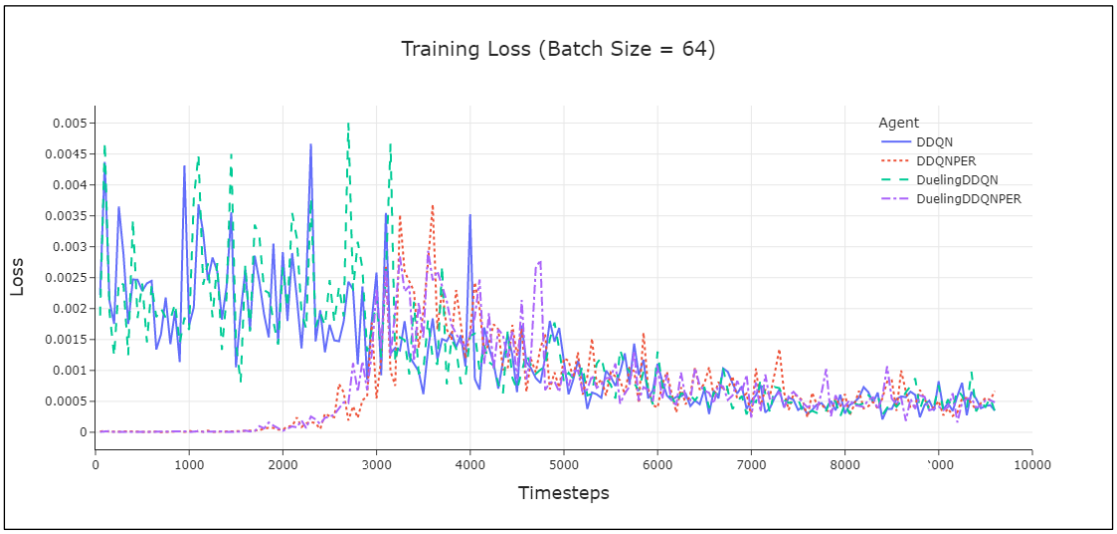
4.1.1 วัตถุประสงค์

เพื่อศึกษาพฤติกรรมและผลกระทบของปริมาณข้อมูลในการสุ่มมาเรียนรู้ในแต่ละครั้ง ว่าปริมาณข้อมูลที่นำมาเรียนรู้ที่แตกต่างกันจะส่งผลให้แต่ละอัลกอริทึมมีเสถียรภาพในการเรียนรู้เพิ่มมากขึ้นหรือลดน้อยลงหรือไม่ โดยดูจากกราฟแสดงค่าผิดพลาด (Error) และปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนและทดสอบ

4.1.2 การทดลองที่ 1.1 สุ่มข้อมูลทีละ 64 samples ในการเรียนรู้

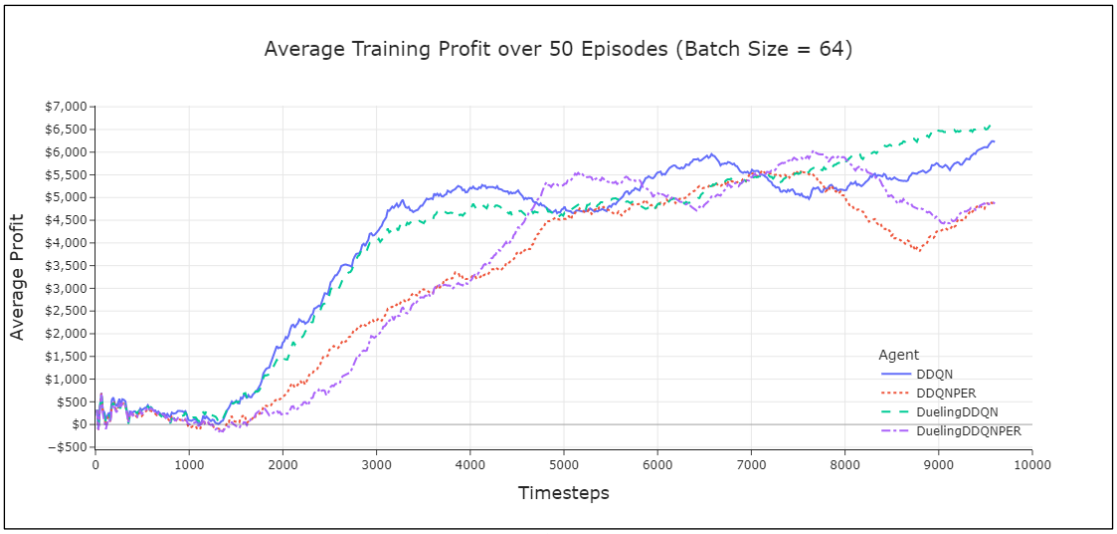
จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง ซึ่งในช่วงแรกจะพบว่า Agent ที่ใช้ถังเก็บข้อมูลประสบการณ์แบบจัดลำดับความสำคัญ (Prioritized Experience Replay) มีปริมาณค่าผิดพลาดที่

ต่ำมากเนื่องจาก Agent ที่ใช้ Prioritized Experience Replay ในการเก็บประสบการณ์ นั้นจะมีการนำค่าน้ำหนักเข้ามาคูณกับค่าผิดพลาดเพื่อแก้ปัญหา Bias จากที่กล่าวไปในหัวข้อที่ 2.7.6.2 ทำให้ในช่วงแรกค่าน้ำหนักนี้จะเป็นตัวถ่วงค่าผิดพลาดให้ต่ำลง แต่เมื่อผ่านการฝึกสอนไประดับหนึ่ง ณ จำนวน timesteps ที่ 3000 (แกน x) จะพบว่าทั้ง 4 Agent มีค่าผิดพลาดที่พอ ๆ กันตามแผนภาพที่ 4.1



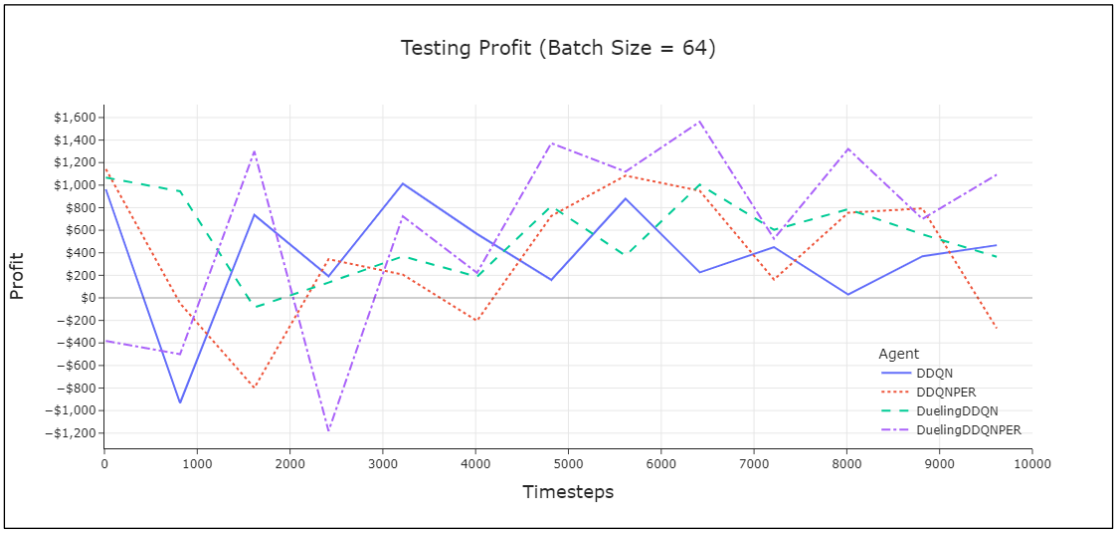
แผนภาพที่ 4.1 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่า Agent ที่ใช้ถังเก็บข้อมูลประสบการณ์ทั่วไป (Experience Replay) สามารถทำกำไรเฉลี่ยจาก 50 Episode ได้ดีกว่า Agent ที่ใช้ Prioritized Experience Replay ในการเก็บประสบการณ์ เนื่องจาก Agent ที่ใช้ Experience Replay นั้น จะให้ความสำคัญกับสถานะที่มักจะเจอบ่อยครั้งมากกว่าสถานะที่นาน ๆ จะเจอที ทำให้ในขั้นตอนฝึกสอนสามารถทำกำไรได้ดีกว่า Prioritized Experience Replay เพราะ Prioritized Experience Replay นั้นในช่วงแรกจะมีการเรียนรู้กับสถานะที่นาน ๆ จะเจอทีมากกว่าสถานะที่มักจะเจอบ่อยครั้งดังแผนภาพที่ 4.2



แผนภาพที่ 4.2 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples

เมื่อนำมาเปรียบเทียบกับกำไรในขั้นตอนทดสอบจะพบว่า Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay ในการเก็บประสบการณ์ สามารถทำกำไรได้โดนเด่นกว่า Agent ที่ใช้อัลกอริทึมในรูปแบบอื่นอย่างเห็นได้ชัด ซึ่งตรงข้ามกับปริมาณกำไรที่ Agent ทำได้ในขั้นตอนการฝึกสอน ดังแผนภาพที่ 4.3

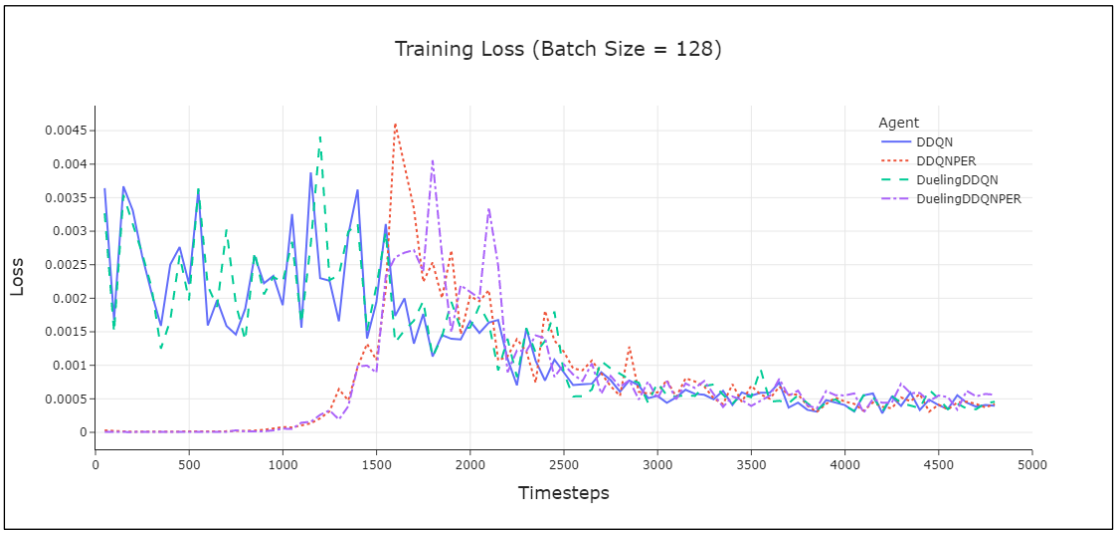


แผนภาพที่ 4.3 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 64 samples

4.1.3 การทดลองที่ 1.2 สุ่มข้อมูลทีละ 128 samples ในการเรียนรู้

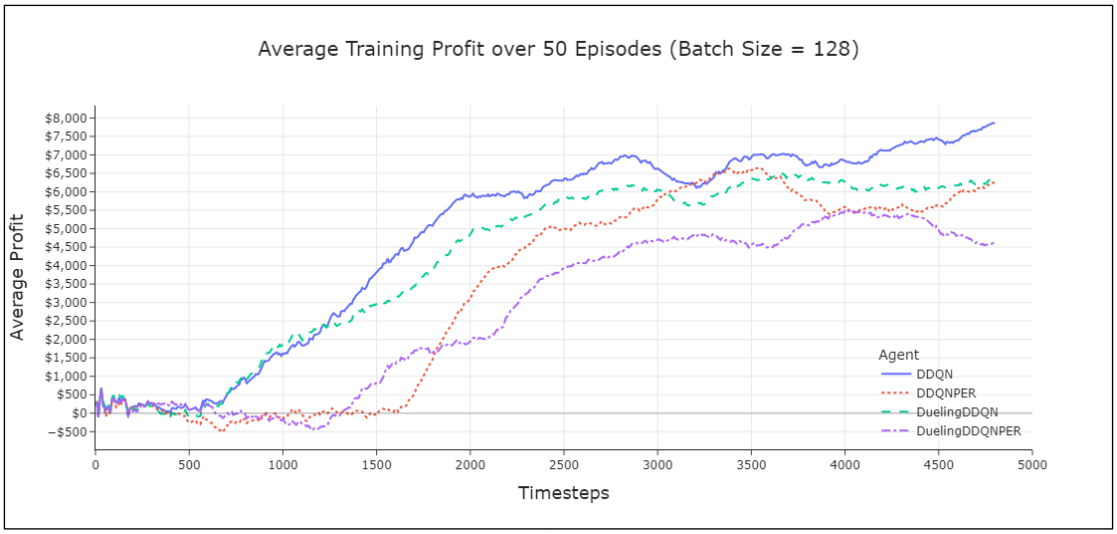
จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง ซึ่งในช่วงแรกจะพบว่า Agent ที่ใช้ Prioritized

Experience Replay มีปริมาณค่าผิดพลาดที่ต่ำมากเหมือนกับการทดลองที่ผ่านมา แต่เมื่อผ่านการฝึกสอนไประดับหนึ่ง ณ จำนวน timesteps ที่ 1500 จะพบว่าเมื่อมีการสุ่มข้อมูลเพิ่มขึ้นเป็น 2 เท่า เมื่อเทียบกับการทดลองที่ 1.1 จำนวน timesteps ในการเรียนรู้ก็จะลดลงไป 2 เท่าตัวจาก 9500 timesteps เหลือเพียง 4750 timesteps แต่ถึงแม้จะมีการเพิ่มปริมาณข้อมูลในการสุ่มมาเรียนรู้เพิ่มขึ้นจากเดิมเป็น 2 เท่าตัวแล้วก็ตาม แต่ทั้ง 4 Agent ก็ยังมีปริมาณค่าผิดพลาดใกล้เคียงกับการทดลองที่ 1.1 ดังแผนภาพที่ 4.4



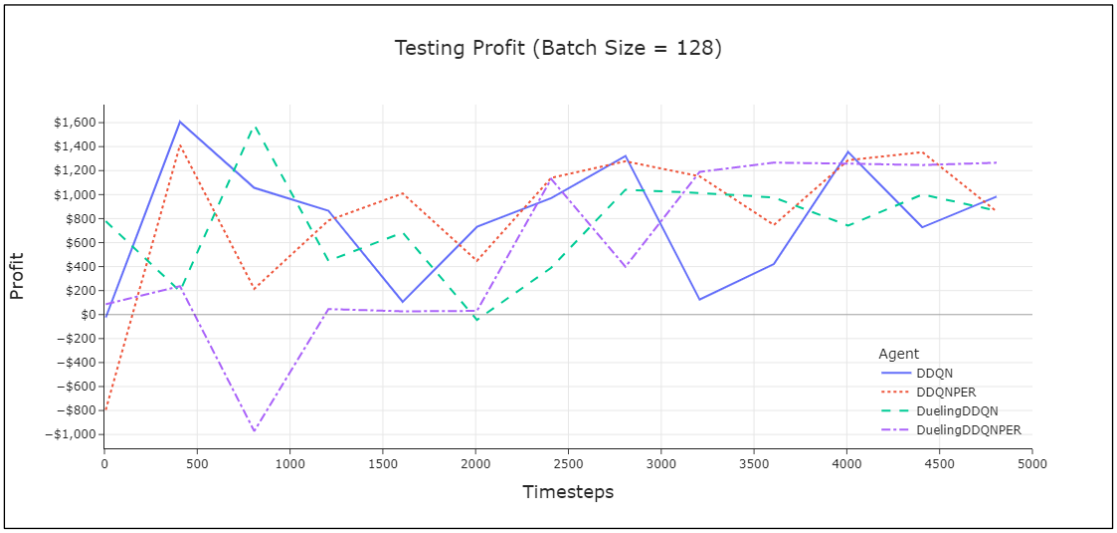
แผนภาพที่ 4.4 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่า Agent ที่ใช้ Experience Replay สามารถทำกำไรเฉลี่ย 50 Episode ได้ดีกว่า Agent ที่ใช้ Prioritized Experience Replay เหมือนกับการทดลองที่ 1.1 ดังแผนภาพที่ 4.5



แผนภาพที่ 4.5 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples

แต่ในทางกลับกันเมื่อดูเทียบกับกำไรในขั้นตอนทดสอบจะพบว่า Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay สามารถทำกำไรได้โดนเด่นกว่า Agent ที่ใช้อัลกอริทึมในรูปแบบอื่น ๆ อย่างเห็นได้ชัด ดังแผนภาพที่ 4.6



แผนภาพที่ 4.6 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 128 samples

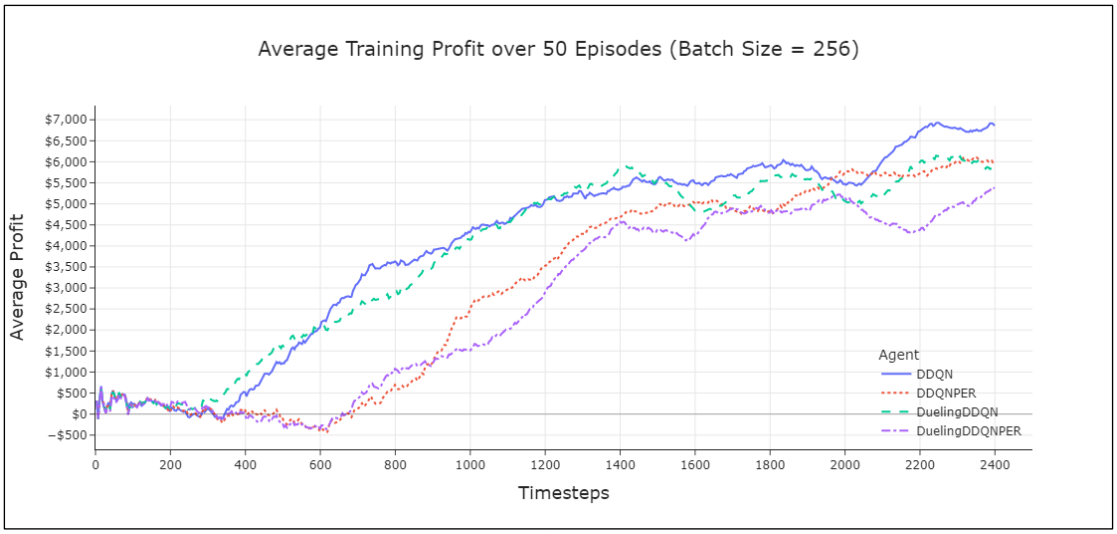
4.1.4 การทดลองที่ 1.3 สุ่มข้อมูลทีละ 256 samples ในการเรียนรู้

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง ซึ่งในช่วงแรกจะพบว่า Agent ที่ใช้ Prioritized Experience Replay มีปริมาณค่าผิดพลาดที่ต่ำมากเหมือนกับการทดลองที่ผ่านมาทั้ง 2 การทดลอง

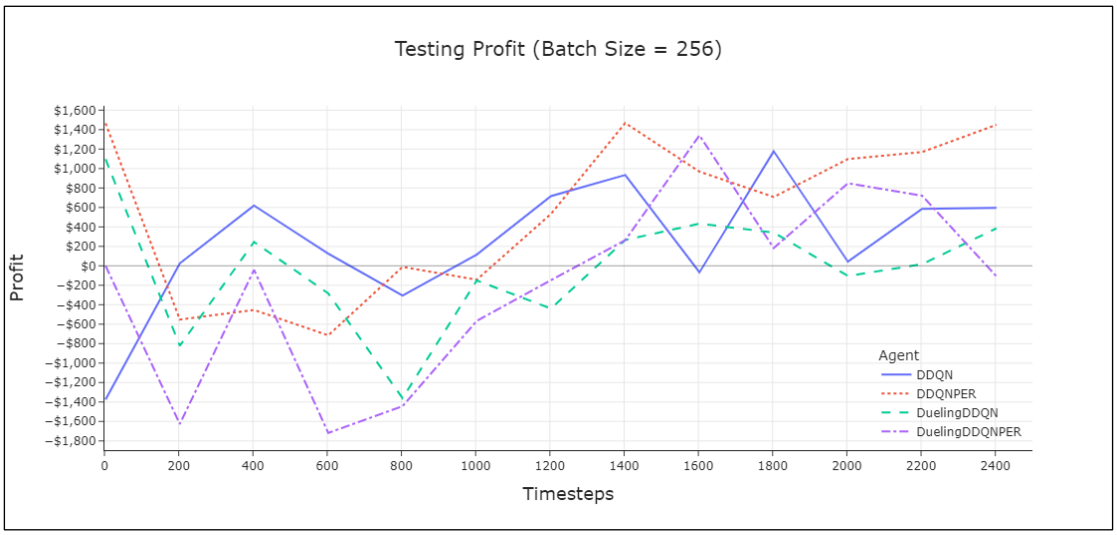
แต่เมื่อผ่านการฝึกสอนไประดับหนึ่ง ณ จำนวน timesteps ที่ 750 จะพบว่าเมื่อมีการสุ่มข้อมูลเพิ่มขึ้นเป็น 2 เท่าเมื่อเทียบกับการทดลองที่ 1.2 จำนวน timesteps ในการเรียนรู้ก็จะลดลงไป 2 เท่าตัวเช่นกันจาก 4750 timesteps เหลือเพียง 2375 timesteps ถึงแม้ว่าจะมีการเพิ่มปริมาณข้อมูลสุ่มเพิ่มขึ้นจากเดิมเป็น 2 เท่าตัวแล้วก็ตาม แต่ทั้ง 4 Agent ก็ยังคงมีปริมาณค่าผิดพลาดใกล้เคียงกับการทดลองที่ 1.1 และการทดลองที่ 1.2 ดังแผนภาพที่ 4.7

แผนภาพที่ 4.7 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่า Agent ที่ใช้ Experience Replay สามารถทำกำไรเฉลี่ย 50 Episode ได้ดีกว่า Agent ที่ใช้ Prioritized Experience Replay เหมือนกับการทดลองที่ 1.1 และการทดลองที่ 1.2 ดังแผนภาพที่ 4.8 แต่ในทางกลับกันเมื่อดูเทียบกับกำไรในขั้นตอนทดสอบจะพบว่าในการทดลองนี้ Agent ที่สามารถทำกำไรได้ดีที่สุดกลับไม่ใช่ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay ในการเก็บประสบการณ์ เหมือนกับการทดลองที่ 1.1 และการทดลองที่ 1.2 แต่เป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Prioritized Experience Replay แทนที่สามารถทำกำไรได้โดนเด่นกว่าอัลกอริทึมอื่น ๆ อย่างเห็นได้ชัด ดังแผนภาพที่ 4.9



แผนภาพที่ 4.8 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples



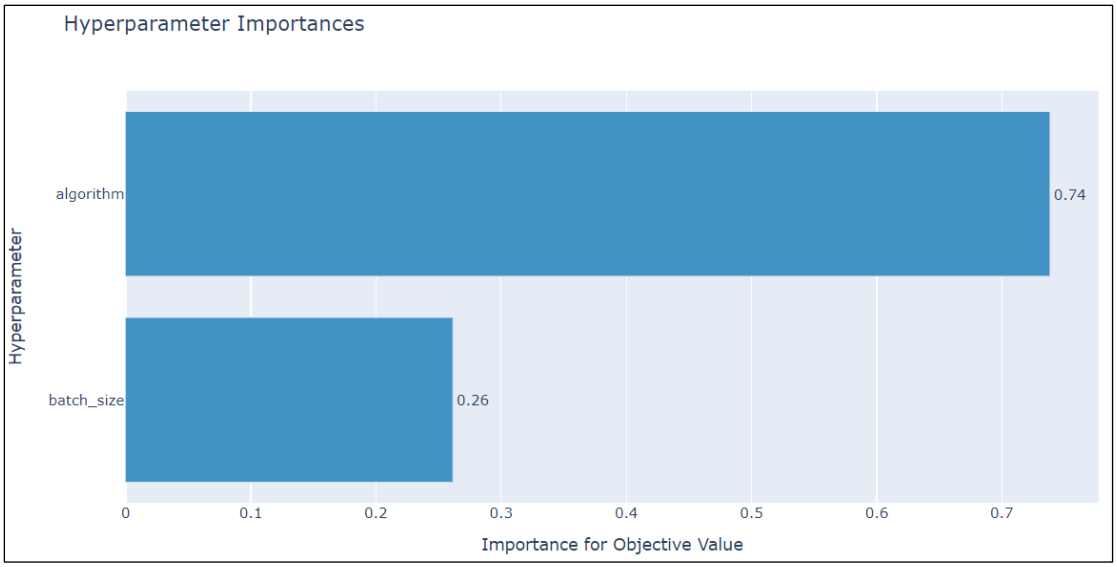
แผนภาพที่ 4.9 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดปริมาณข้อมูลในการเรียนรู้ทีละ 256 samples

4.1.5 สรุปผลการทดลอง

จากการทดลองทั้ง 3 การทดลองสามารถสรุปผลได้ว่า Agent ที่ใช้ Experience Replay สามารถทำกำไรได้ดีกว่า Agent ที่ใช้ Prioritized Experience Replay ในขั้นตอนการฝึกสอน เนื่องจาก Prioritized Experience Replay นั้นมีการสมดุลกันระหว่างสถานะที่มักเกิดขึ้นบ่อยหรือสถานะที่นาน ๆ มักจะเกิดขึ้นทีทำให้ในขั้นตอนการฝึกสอนนั้นไม่สามารถทำกำไรได้มากกว่า Agent ที่ใช้ Experience Replay ในทางตรงกันข้าม Agent ที่ใช้ Prioritized Experience Replay สามารถทำกำไรได้ดีกว่า Agent ที่ใช้ Experience Replay ในขั้นตอนทดสอบ จากการสมดุลกันระหว่างสถานะที่มักเกิดขึ้นบ่อยหรือสถานะที่นาน ๆ มักจะเกิดขึ้น

ในส่วนของการปรับปริมาณการสุ่มข้อมูลเพื่อนำมาเรียนรู้นั้น ไม่ค่อยมีความแตกต่างกันเท่าไรจะพบว่าทั้ง 3 การทดลองค่าผิดพลาดจะลู่เข้าที่ 0.0005 ทั้งหมด แต่จากภาพรวมแล้วจะพบว่า ปริมาณการสุ่มทีละ 128 samples เพื่อนำมาเรียนรู้สามารถทำกำไรได้ดีทั้งในขั้นตอนการฝึกสอนและขั้นตอนการทดสอบจากทั้ง 4 Agent ดังนั้นจึงขอสรุปว่าปริมาณการสุ่มทีละ 128 samples เพื่อนำมาเรียนรู้เหมาะสมที่สุดในงานวิจัยนี้

ดังนั้นเมื่อนำทั้ง 3 การทดลองมาคำนวณหาความสำคัญของ Hyperparameter ต่อรางวัลสะสมด้วยวิธีการ Functional ANOVA (fANOVA) จะให้ผลลัพธ์ดังแผนภาพที่ 4.10 จะพบว่าค่าความสำคัญของ Hyperparameter นั้นให้น้ำหนักกับตัวอัลกอริทึมสูงถึง 74% ซึ่งในกรณีนี้คือ Agent ทั้ง 4 ที่แตกต่างกัน เมื่อเทียบกับปริมาณข้อมูลที่สุ่มเพื่อนำมาเรียนรู้ที่คิดเป็น 26% จึงสามารถสรุปได้อีกครั้งว่าอัลกอริทึมมีผลต่อประสิทธิภาพมากกว่าปริมาณข้อมูลที่สุ่มเพื่อนำมาเรียนรู้ และปริมาณการสุ่มที่เหมาะสมที่สุดในงานวิจัยนี้คือทีละ 128 samples



แผนภาพที่ 4.10 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึมและขนาดของข้อมูลที่สุ่มมาเรียนรู้ที่ส่งผลต่อรางวัลสะสม

4.2 การทดลองที่ 2 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับอัลกอริทึมในการเรียนรู้ (Optimizer) และอัตราการเรียนรู้ (Learning Rate)

อัลกอริทึมในการเรียนรู้คือ กระบวนการในการเรียนรู้เพื่อหาแนวทางในการปรับค่าน้ำหนักของโครงข่ายประสาทเทียมที่จะทำให้ค่าสูญเสียลดลง กล่าวคือทำให้โครงข่ายประสาทเทียมทำงานได้ดียิ่งขึ้น ซึ่งในการทดลองนี้เลือกอัลกอริทึมในการเรียนรู้มา 2 วิธีการคือ AdamW และ RMSprop เนื่องจากอัลกอริทึมในการเรียนรู้อย่าง Gradient Descent นั้นจะมีความเสถียรก็ต่อเมื่อข้อมูลมีอิสระต่อกันและมีการแจกแจงที่เหมือนกันและเป้าหมายในการเรียนรู้ต้องคงที่ ซึ่งจากที่กล่าว

ไปการเรียนรู้แบบเสริมแรงเชิงลึกนั้น มีการนำถังเก็บข้อมูลประสบการณ์และโครงข่ายประสาทเทียมสำหรับการประเมินมาแก้ปัญหา 2 ข้อนี้ แต่ก็ไม่ได้มีการการันตีว่าจะเป็นไปตามข้อสมมุติฐานเสมอ ดังนั้นการใช้อัลกอริทึมอย่าง Gradient Descent จึงมีความไม่เสถียร โดยส่วนใหญ่จึงนิยมใช้อัลกอริทึม RMSprop เนื่องจากเป็นวิธีการที่ค่อนข้างมีความปลอดภัยจากการหาค่าเฉลี่ยของ Gradient แทนและอัลกอริทึม AdamW เนื่องจากกระบวนค่อนข้างเหมือนกับ RMSprop แต่มีการนำโมเมนตั้มเข้ามาช่วยเพิ่มน้ำหนักของ Gradient และมีกระบวนการลดน้ำหนักเหล่านั้นให้น้อยลงตามช่วงเวลาดังนั้นวิธีการ AdamW จะค่อนข้างมีความแข็งขันกว่าวิธีการ RMSprop ซึ่งทั้ง 2 วิธีการมักใช้อัตราการเรียนรู้ที่อยู่ในช่วง 0.001 ถึง 0.0001 แต่ที่นิยมมากที่สุดคืออัตราการเรียนรู้ที่ 0.001

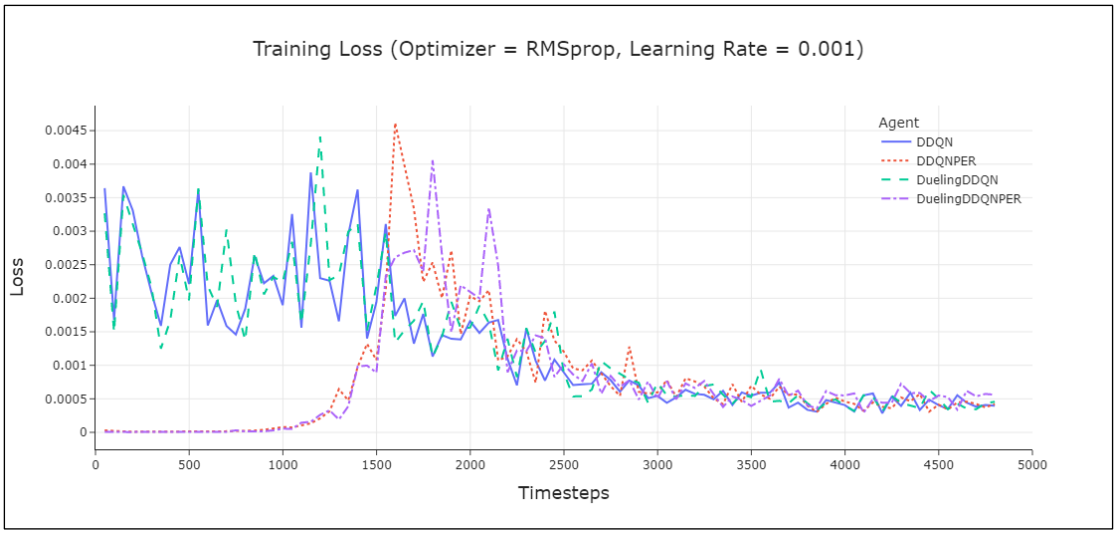
ในการทดลองนี้จะมีการแบ่งเป็น 4 การทดลองย่อย ซึ่งมีการกำหนดอัลกอริทึมในการเรียนรู้และอัตราการเรียนรู้ดังนี้ 1. อัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001 2. อัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.005 3. อัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001 และ 4. อัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005 ตามลำดับ และมีการกำหนดค่าให้กับพารามิเตอร์ดังนี้ Gamma มีค่าเป็น 0.7 และ Batch Size มีค่าเป็น 128

4.2.1 วัตถุประสงค์

เพื่อศึกษาพฤติกรรมและผลกระทบของแต่ละอัลกอริทึมที่ใช้ในการเรียนรู้และอัตราการเรียนรู้ว่าอัลกอริทึมไหนที่สามารถเรียนรู้ได้รวดเร็วกว่าและอัตราการเรียนรู้ช่วยเพิ่มเสถียรภาพในการเรียนรู้หรือไม่ โดยดูจากกราฟแสดงค่าผิดพลาด (Error) และปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนและทดสอบ

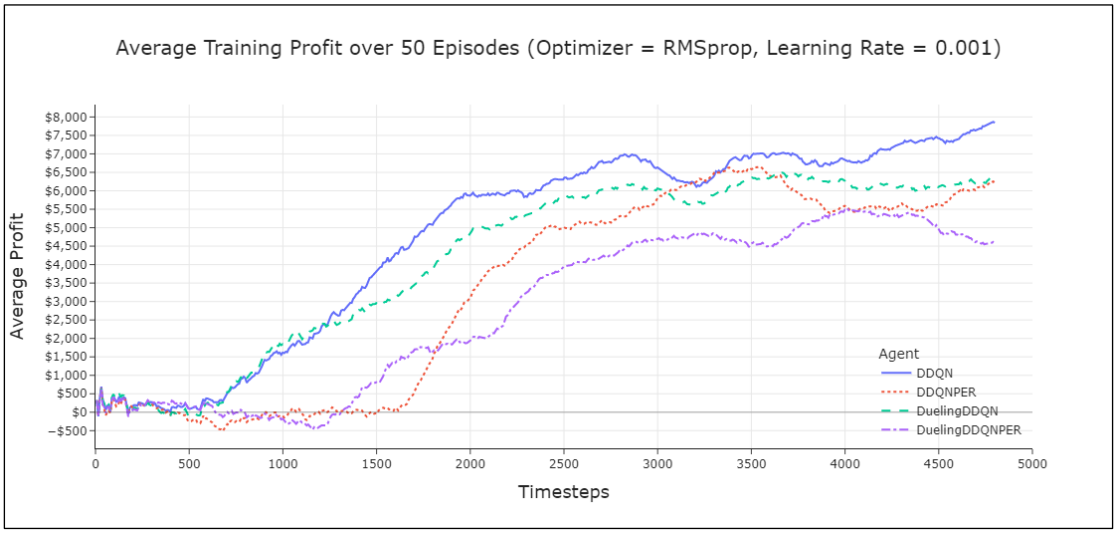
4.2.2 การทดลองที่ 2.1 ใช้อัลกอริทึม RMSprop ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.001

จากการทดลองพบว่า ทั้ง 4 Agent เมื่อมีการใช้อัลกอริทึม RMSprop ในการเรียนรู้ด้วยอัตราการเรียนรู้ที่ 0.001 ค่าผิดพลาดจะเริ่มมีการลู่เร็วมากขึ้นในช่วง timesteps ที่ 1500 – 2500 และจะเริ่มค่อย ๆ ลู่เข้าสู่ 0.0005 ในช่วง timesteps ที่ 2500 – 4500 เหมือนกันทั้ง 4 Agent ดังแผนภาพที่ 4.11



แผนภาพที่ 4.11 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001

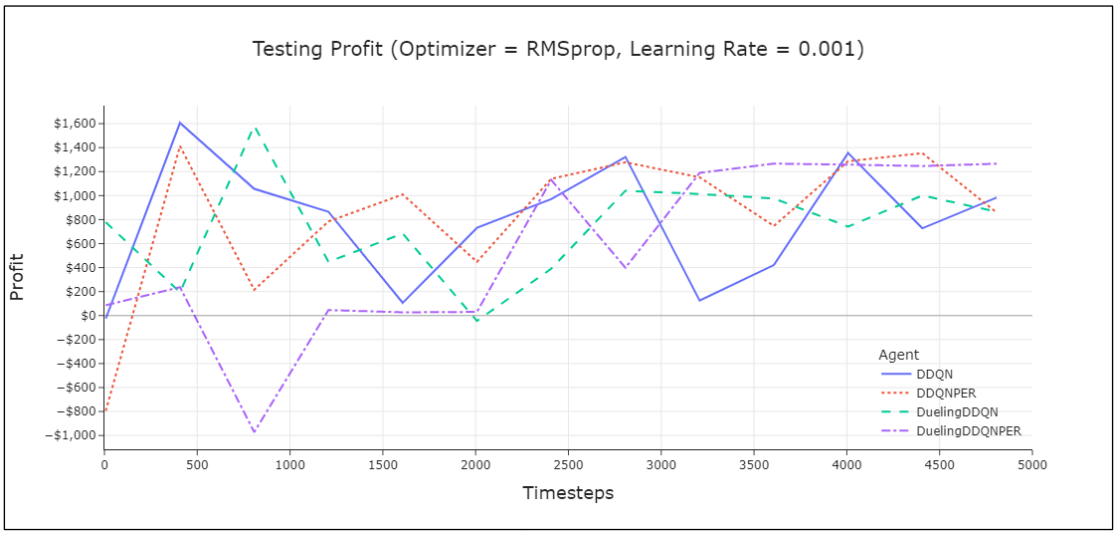
จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent สามารถทำกำไรได้อย่างก้าวกระโดดในช่วง timesteps ที่ 1500 – 2500 ดังนั้นกราฟที่แสดงค่าผิดพลาดในช่วงนั้นจึงลดลงอย่างรวดเร็ว ดังแผนภาพที่ 4.12



แผนภาพที่ 4.12 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001

แต่ในทางกลับกันเมื่อนำ Agent มาทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อน ในช่วง timesteps ที่ 1500 – 2500 นั้น Agent ยังทำกำไรได้ไม่ดีมากพอ เนื่องจากโครงข่ายประสาทเทียมอาจจะยังเรียนรู้ไม่มากพอแต่เมื่อไปถึงจุดหนึ่ง ณ ค่าผิดพลาดที่เข้าใกล้ 0.0005 จะพบว่าทั้ง 4 Agent สามารถทำกำไรได้ดีทั้งในขั้นตอนการฝึกสอนและขั้นตอนการทดสอบ และจะพบว่า Agent ที่สามารถ

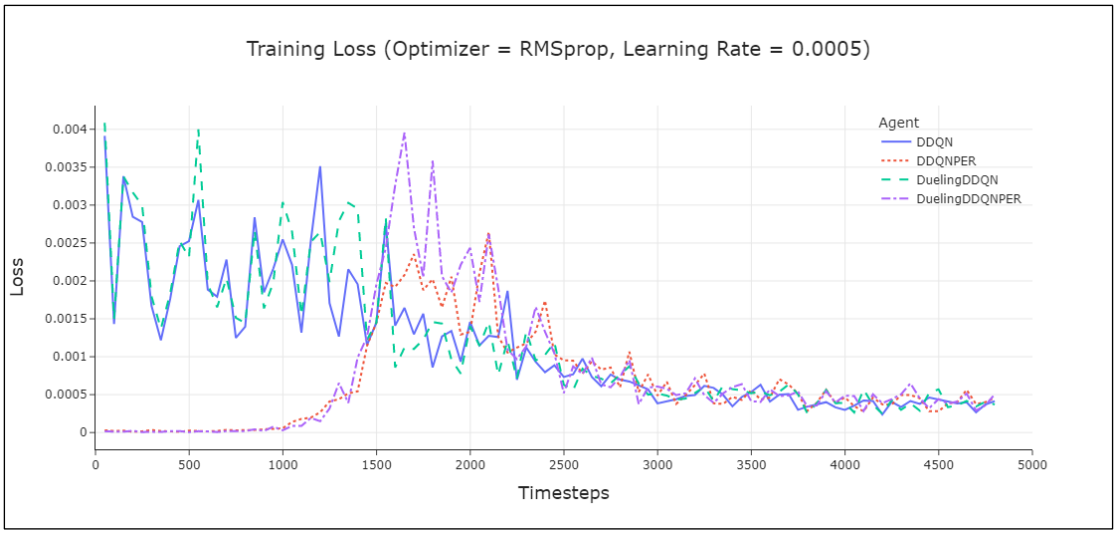
ทำกำไรได้ดีที่สุดในขั้นตอนการฝึกสอนคือ Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay แต่ในขั้นตอนการทดสอบ Agent ที่ทำกำไรได้ดีที่สุดกลับเป็น Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay ดังแผนภาพที่ 4.13



แผนภาพที่ 4.13 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001

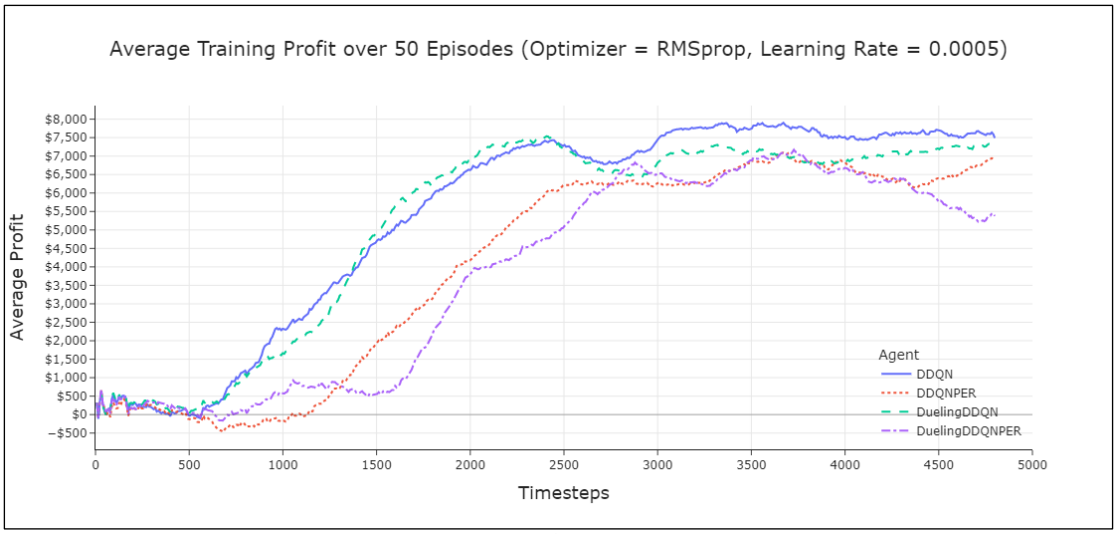
4.2.3 การทดลองที่ 2.2 ใช้อัลกอริทึม RMSprop ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.0005

จากการทดลองพบว่า ทั้ง 4 Agent เมื่อใช้อัลกอริทึม RMSprop ในการเรียนรู้ด้วยอัตราการเรียนรู้ที่ 0.0005 ค่าผิดพลาดจะเริ่มมีการลู่เร็วมากขึ้นในช่วง timesteps ที่ 1500 – 2500 และจะเริ่มค่อย ๆ ลู่เข้าสู่ 0.0005 ในช่วง timesteps ที่ 2500 – 4500 ทั้ง 4 Agent เช่นเดียวกันกับการทดลองที่ 2.1 แต่จะพบว่าเมื่อมีการใช้อัตราการเรียนรู้ที่ 0.0005 ตัวกราฟจะมีการสวิงที่น้อยกว่ามีเสถียรภาพในการเรียนรู้มากกว่า และพบว่า ณ timesteps ที่ 2500 จะมีค่าผิดพลาดต่ำกว่าการใช้ RMSprop ที่อัตราการเรียนรู้ที่ 0.001 อยู่เล็กน้อย และเมื่อการฝึกสอนเสร็จสิ้นทั้ง 4 Agent จะมีค่าผิดพลาดน้อยกว่า 0.0005 ซึ่งน้อยกว่าการทดลองที่ 2.1 ดังแผนภาพที่ 4.14

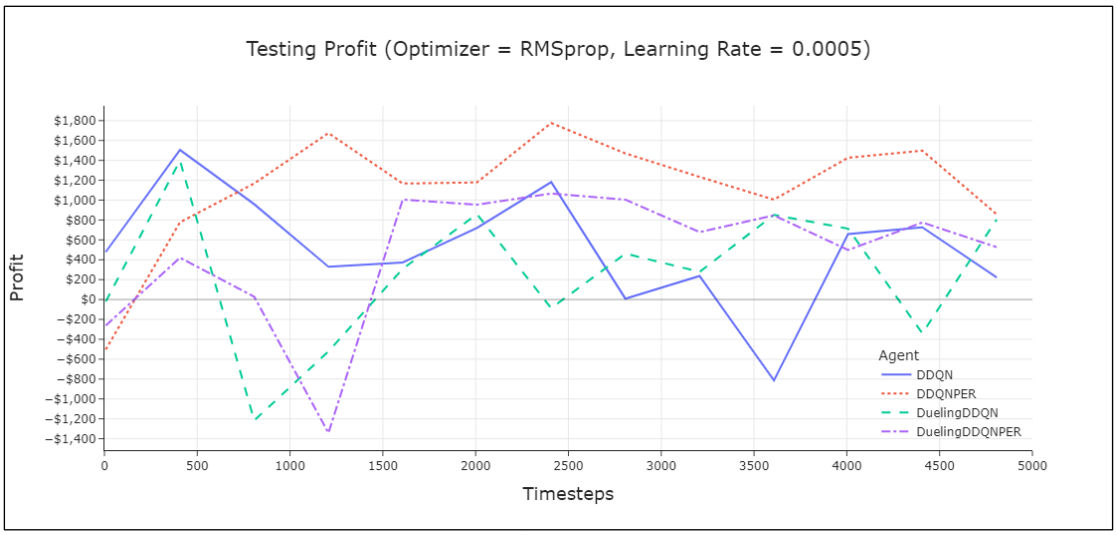


แผนภาพที่ 4.14 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent สามารถทำกำไรได้อย่างก้าวกระโดดในช่วง timesteps ที่ 1500 – 2500 ดังนั้นกราฟที่แสดงค่าผิดพลาดในช่วงนั้นจึงลดลงอย่างรวดเร็วนั้นเองดังแผนภาพที่ 4.14 แต่ในทางกลับกันเมื่อนำมาทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อน ในช่วง timesteps นี้ Agent ยังคงทำกำไรได้ไม่ดีมากพอ เนื่องจากโครงข่ายประสาทเทียมอาจจะยังเรียนรู้ไม่มากพอแต่เมื่อไปถึงจุดหนึ่ง ณ ค่าผิดพลาดที่เข้าใกล้ 0.0005 จะพบว่าทั้ง 4 Agent สามารถทำกำไรได้ทั้งขั้นตอนการฝึกสอนและขั้นตอนการทดสอบ และจะพบว่า Agent ที่สามารถทำกำไรได้ดีที่สุดในขั้นตอนการฝึกสอนคือ Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay แต่ในขั้นตอนการทดสอบ Agent ที่ทำกำไรได้ดีที่สุดกลับไม่ใช่ Agent ที่ใช้อัลกอริทึม DuelingDDQN แต่เป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Prioritized Experience Replay ดังแผนภาพที่ 4.15 ซึ่งเมื่อเทียบกับการทดลองที่ 2.1 จะพบว่าการใช้อัตราการเรียนรู้ที่ 0.0005 ทั้ง 4 Agent โดยเฉลี่ยจะสามารถทำกำไรในขั้นตอนฝึกสอนได้มากกว่าการใช้อัตราการเรียนรู้ที่ 0.001 แต่ในทางกลับกันในขั้นตอนทดสอบการใช้อัตราการเรียนรู้ที่ 0.001 กลับทำกำไรได้ดีกว่าการใช้อัตราการเรียนรู้ที่ 0.0005



แผนภาพที่ 4.15 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005

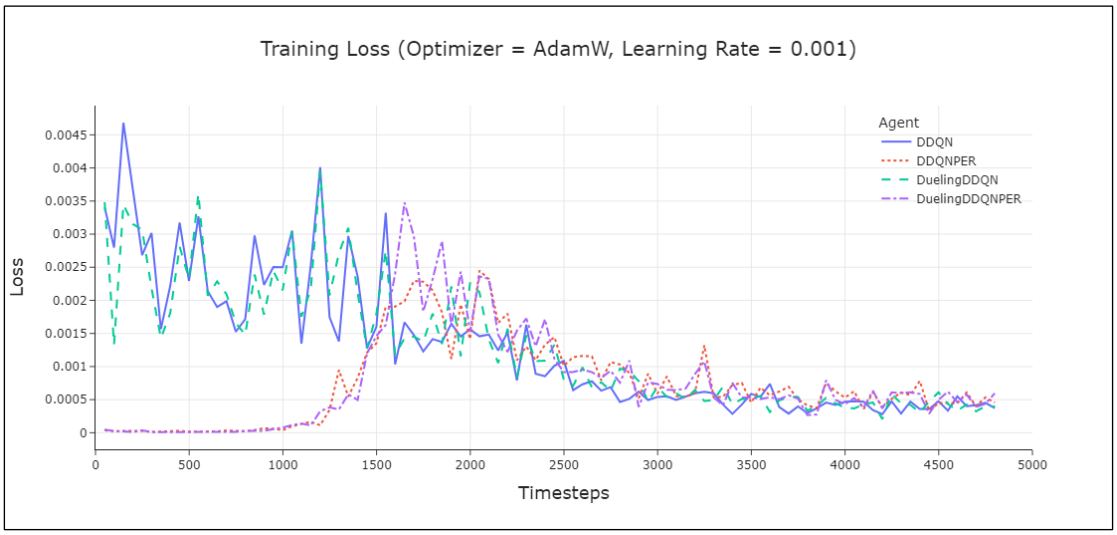


แผนภาพที่ 4.16 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.0005

4.2.4 การทดลองที่ 2.3 ใช้อัลกอริทึม AdamW ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.001

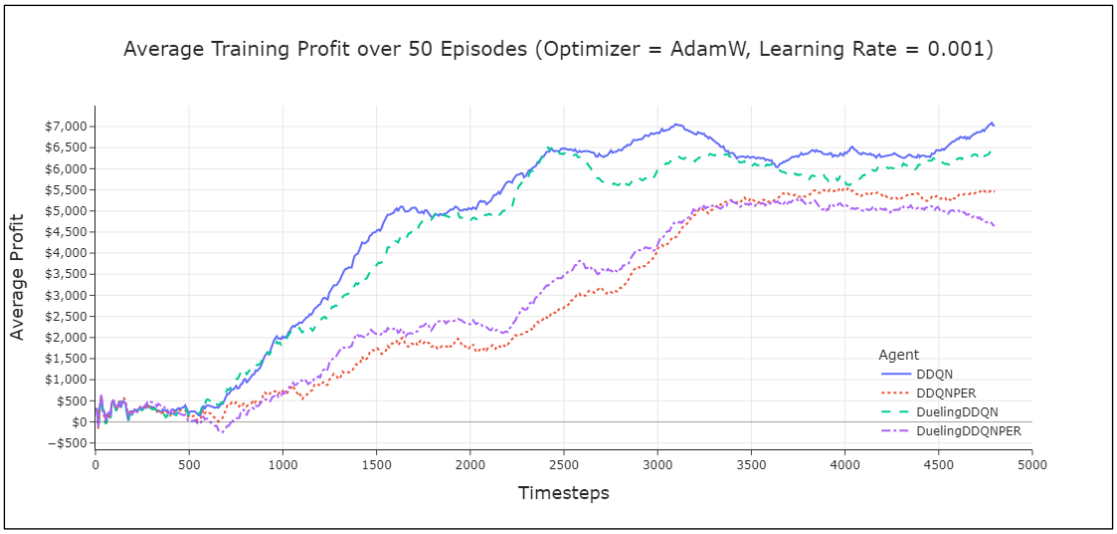
จากการทดลองพบว่า ทั้ง 4 Agent เมื่อใช้อัลกอริทึม AdamW ในการเรียนรู้ด้วยอัตราการเรียนรู้ที่ 0.001 ค่าผิดพลาดจะเริ่มมีการลู่เร็วมากขึ้นในช่วง timesteps ที่ 1500 – 2500 และจะเริ่มค่อย ๆ ลู่เข้าสู่ 0.0005 ในช่วง timesteps ที่ 2500 – 4500 ทั้ง 4 Agent เช่นเดียวกันกับการใช้อัลกอริทึม RMSprop ในการเรียนรู้ ดังการทดลองที่ 2.1 และการทดลองที่ 2.2 ซึ่งเมื่อนำมาเปรียบเทียบกับอัลกอริทึม RMSprop ที่อัตราการเรียนรู้ที่ 0.001 จะมีค่าผิดพลาดมีการลดลงใกล้เคียงกันแต่อัลกอริทึม RMSprop จะมีการสวิงของค่าผิดพลาดที่สูงกว่าในช่วง 1500 – 2500 จะ

เห็นได้อย่างชัดเจนว่ามีการขึ้นลงที่มากกว่าอัลกอริทึม AdamW แต่สุดท้ายค่าผิดพลาดก็ยังลู่เข้าที่ 0.0005 เช่นเดียวกัน ดังแผนภาพที่ 4.17



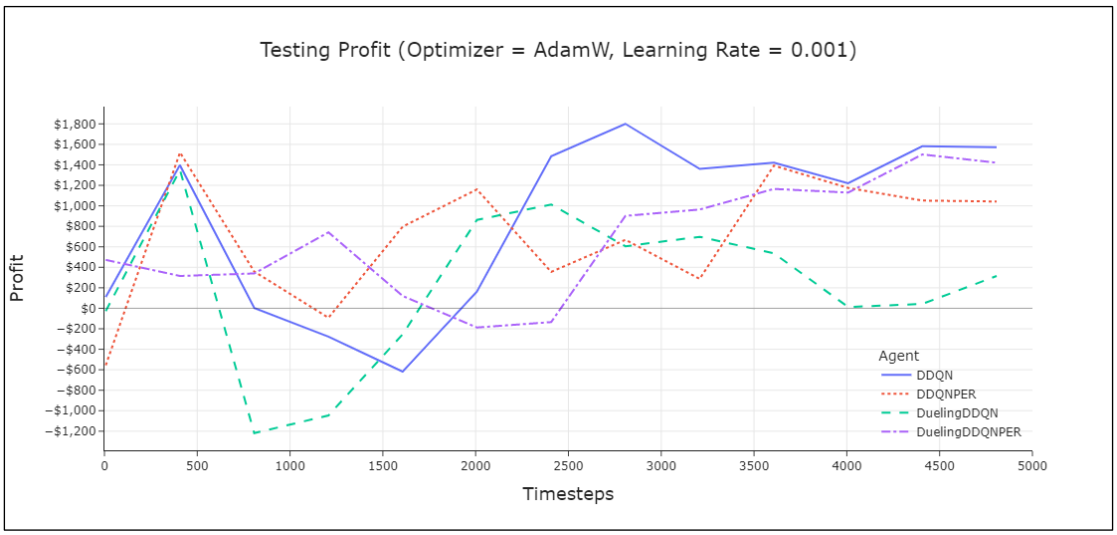
แผนภาพที่ 4.17 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่า Agent ที่ใช้อัลกอริทึม DDQN และ DuelingDDQN ที่ใช้ Experience Replay สามารถทำกำไรได้อย่างก้าวกระโดดในช่วง timesteps ที่ 1500 – 2500 ดังนั้นกราฟที่แสดงค่าผิดพลาดในช่วงนั้นจะลดลงอย่างรวดเร็ว แต่ Agent ที่ใช้อัลกอริทึม DDQN และ DuelingDDQN ที่ใช้ Prioritized Experience Replay จะสามารถทำกำไรได้เพิ่มมากขึ้นในช่วง timesteps ที่ 1000 - 3000 ดังแผนภาพที่ 4.18 ซึ่งเมื่อเทียบกับการทดลองที่ 2.1 Agent ที่ใช้ Prioritized Experience Replay และใช้อัลกอริทึม RMSprop ในการเรียนรู้สามารถทำกำไรได้รวดเร็วกว่า Agent ที่ใช้อัลกอริทึม AdamW ในการเรียนรู้



แผนภาพที่ 4.18 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001

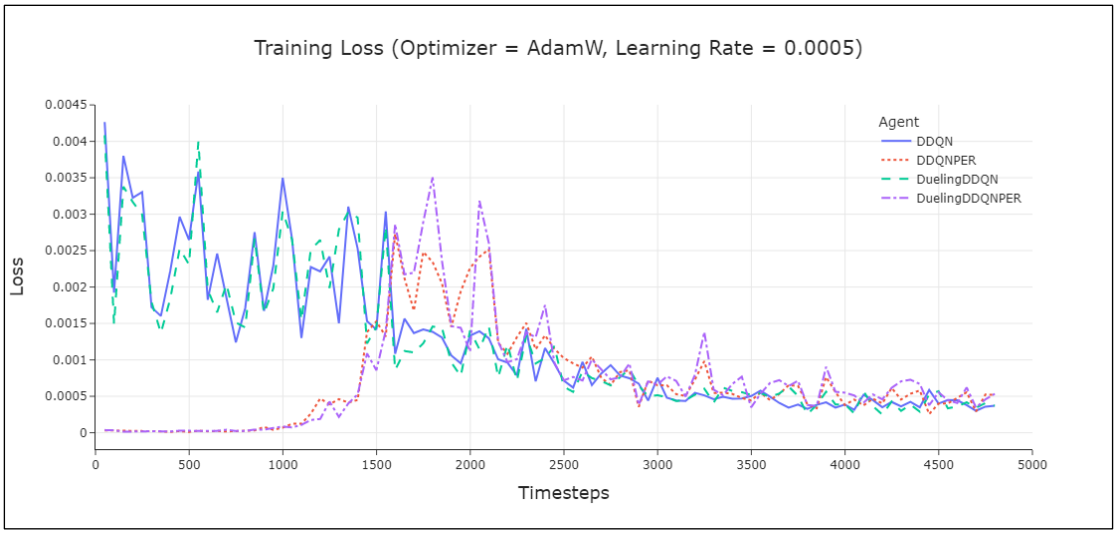
แต่ในทางกลับกันเมื่อนำมาทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อนจะพบว่าทั้ง 4 Agent สามารถทำกำไรได้ดีทั้งขั้นตอนการฝึกสอนและขั้นตอนการทดสอบ และจะพบว่า Agent ที่สามารถทำกำไรได้ดีที่สุดในขั้นตอนการฝึกสอนคือ Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay เช่นเดียวกันกับการใช้อัลกอริทึม RMSprop ในการเรียนรู้ แต่ในขั้นตอนการทดสอบ Agent ที่ทำกำไรได้ดีที่สุดกลับเป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay เท่านั้น ซึ่งแตกต่างจากการทดลองที่ผ่านมาทั้ง 2 การทดลอง ในขั้นตอนการทดสอบที่พบว่า Agent ที่ใช้ Prioritized Experience Replay จะทำกำไรได้ดีที่สุด ดังแผนภาพที่ 4.19



แผนภาพที่ 4.19 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.001

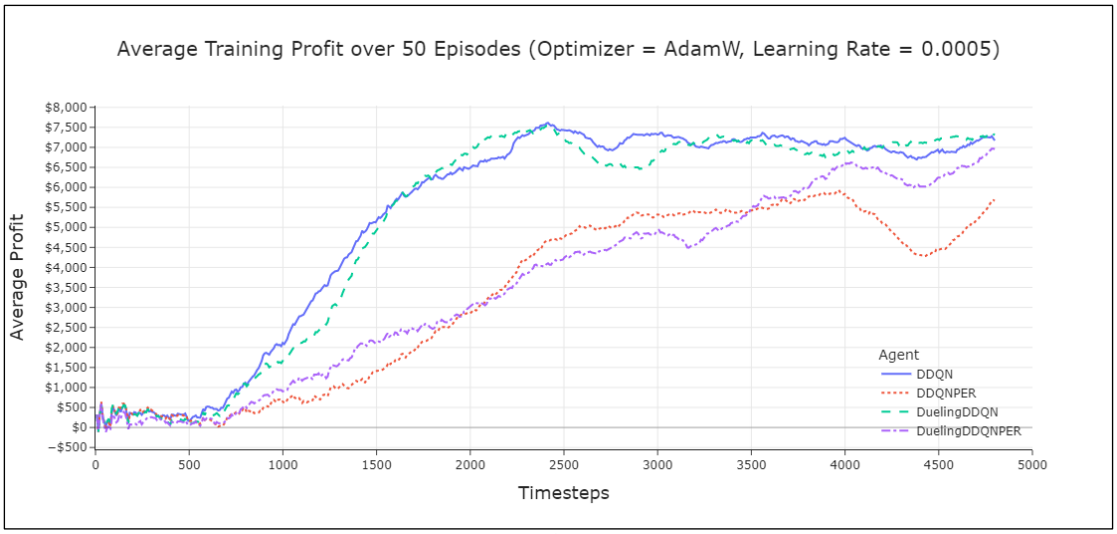
4.2.5 การทดลองที่ 2.4 ใช้อัลกอริทึม AdamW ในการเรียนรู้โดยมีอัตราการเรียนรู้ที่ 0.0005

จากการทดลองพบว่า ทั้ง 4 Agent เมื่อใช้อัลกอริทึม AdamW ในการเรียนรู้ด้วยอัตราการเรียนรู้ที่ 0.0005 อัตราการลดลงของค่าผิดพลาดไม่ค่อยมีความแตกต่างกับการทดลองที่ 2.3 ซึ่งความเร็วในการลดลงมีความเร็วที่พอ ๆ กันแต่เมื่อดูจากกราฟจะพบว่าเมื่อใช้อัตราการเรียนรู้ที่ 0.0005 ค่าผิดพลาดของ Agent ที่ใช้ Prioritized Experience Replay ค่อนข้างมีความผันผวนมากกว่าการใช้อัตราการเรียนรู้ที่ 0.001 ในช่วง timesteps ที่ 1500 – 2500 แต่เมื่อถึงจุด timesteps ที่ 2500 Agent ทั้ง 4 ของทั้ง 2 การทดลองก็ลดลงมาต่ำกว่า 0.001 ทั้งคู่ดังแผนภาพที่ 4.20 แต่โดยภาพรวมจะพบว่าเมื่อใช้ Agent ใช้อัลกอริทึม AdamW ในการเรียนรู้ด้วยอัตราการเรียนรู้ที่ 0.0005 กลับมีความผันผวนมากกว่าอัตราการเรียนรู้ที่ 0.001 ซึ่งตรงข้ามกับการใช้อัลกอริทึม RMSprop ที่อัตราการเรียนรู้ที่ 0.001 มีความผันผวนที่มากกว่าอัตราการเรียนรู้ที่ 0.0005



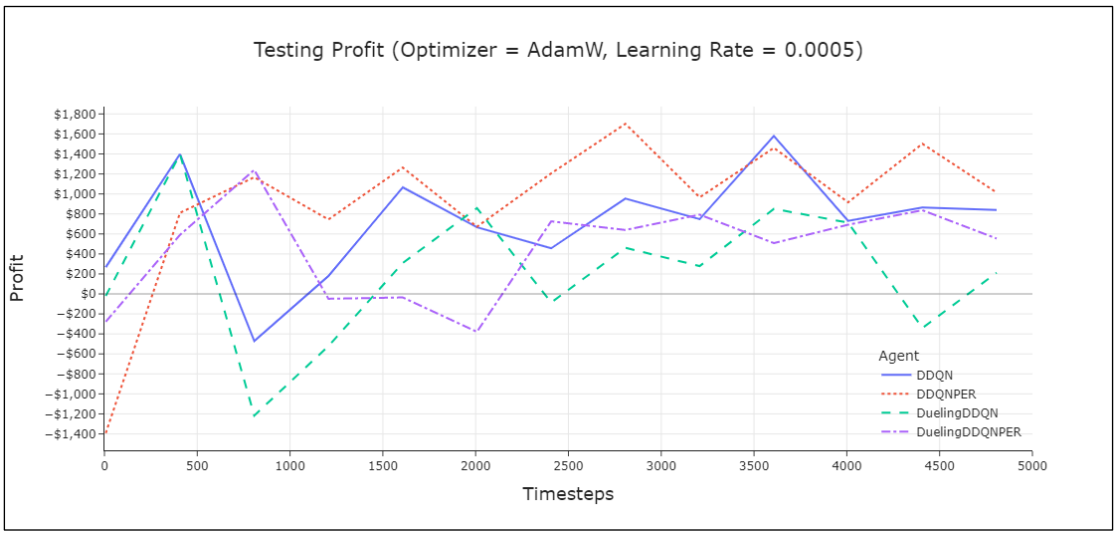
แผนภาพที่ 4.20 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่า Agent ที่ใช้อัลกอริทึม DDQN และ DuelingDDQN ที่ใช้ Experience Replay สามารถทำกำไรได้อย่างก้าวกระโดดในช่วง timesteps ที่ 1500 – 2500 ดังนั้นกราฟที่แสดงค่าผิดพลาดในช่วงนั้นจึงลดลงอย่างรวดเร็ว แต่ Agent ที่ใช้อัลกอริทึม DDQN และ DuelingDDQN ที่ใช้ Prioritized Experience Replay จะสามารถทำกำไรได้สูงขึ้นในช่วง timesteps ที่ 1000 - 3000 ดังแผนภาพที่ 4.21 เฉกเช่นเดียวกับการทดลองที่ 2.3



แผนภาพที่ 4.21 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005

แต่ในทางกลับกันเมื่อนำมาทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อนจะพบว่าทั้ง 4 Agent สามารถทำกำไรได้ดีทั้งขั้นตอนการฝึกสอนและขั้นตอนการทดสอบ และจะพบว่า Agent ที่สามารถทำกำไรได้ดีที่สุดในขั้นตอนการฝึกสอนคือ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Experience Replay ซึ่งแตกต่างจากการทดลองที่ผ่านมาทั้งหมด แต่ Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay ก็สามารถทำกำไรได้ดีไม่แพ้กัน ซึ่งในขั้นตอนการทดสอบ Agent ที่สามารถทำกำไรได้ดีที่สุดกลับเป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Prioritized Experience Replay ซึ่งแตกต่างกับการทดลองที่ 2.3 ที่ Agent ที่สามารถทำกำไรได้ดีกลับเป็น Agent ที่ใช้ Experience Replay ดังแผนภาพที่ 4.22



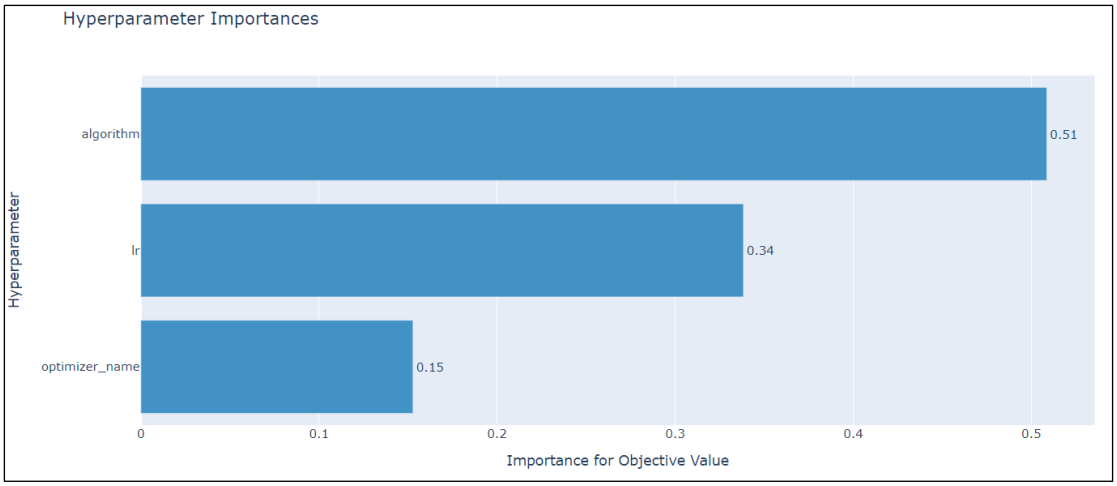
แผนภาพที่ 4.22 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดอัลกอริทึมในการเรียนรู้เป็น AdamW ด้วยอัตราการเรียนรู้ที่ 0.0005

4.2.6 สรุปผลการทดลอง

จากการทดลองทั้ง 4 การทดลองสามารถสรุปผลได้ว่า Agent ที่ใช้ Experience Replay สามารถทำกำไรได้ดีกว่า Agent ที่ใช้ Prioritized Experience Replay ในขั้นตอนการฝึกสอน เนื่องจาก Prioritized Experience Replay นั้นมีการสมดุลกันระหว่างสถานะที่มักเกิดขึ้นบ่อยและสถานะที่นาน ๆ มักจะเกิดขึ้นทีทำให้ในขั้นตอนการฝึกสอนนั้นไม่สามารถทำกำไรได้มากกว่า Agent ที่ใช้ Experience Replay ถึงแม้จะมีการปรับอัลกอริทึมในการเรียนรู้ระหว่าง RMSprop และ AdamW แต่ผลที่ได้กลับพบว่ามันไม่ค่อยส่งผลต่อกำไรเฉลี่ยที่ทำได้ในขั้นตอนการฝึกสอนมาก แต่ถ้ามีการปรับอัตราการเรียนรู้ระหว่าง 0.001 และ 0.0005 จะพบว่าไม่ว่าจะใช้อัลกอริทึมในการเรียนรู้เป็น RMSprop หรือ AdamW การปรับอัตราการเรียนรู้เป็น 0.0005 อัลกอริทึมทั้ง 4 จะสามารถทำกำไรโดยเฉลี่ยในขั้นตอนการฝึกสอนได้ดีกว่าอัตราการเรียนรู้ที่ 0.001

ในขณะที่ขั้นตอนการทดสอบนั้น จะพบว่าอัลกอริทึมที่ใช้ในการเรียนรู้ไม่ว่าจะเป็น RMSprop หรือ AdamW ก็สามารถทำกำไรได้ทั้งคู่แต่สิ่งที่น่าสนใจคือ อัลกอริทึมที่ใช้อัตราการเรียนรู้ที่ 0.001 สามารถทำกำไรได้สูงกว่าอัลกอริทึมที่ใช้อัตราการเรียนรู้ที่ 0.0005 เมื่อใช้อัลกอริทึมในการเรียนรู้แบบเดียวกัน จะพบว่าผลการทดลองไปตรงข้ามกันระหว่างกำไรในขั้นตอนการฝึกสอนและกำไรในขั้นตอนการทดสอบดังนั้นอาจจะมีการทดลองเพิ่มเติมที่ใช้อัตราการเรียนรู้ที่มีค่าอยู่ระหว่าง 0.001 – 0.0005 เพื่อหาอัตราการเรียนรู้ที่เหมาะสมที่สุด

ดังนั้นเมื่อนำทั้ง 4 การทดลองมาคำนวณหาความสำคัญของ Hyperparameter ต่อรางวัลสะสมด้วยวิธีการ Functional ANOVA (fANOVA) จะให้ผลลัพธ์ดังแผนภาพที่ 4.23 จะพบว่าค่าความสำคัญของ Hyperparameter นั้นให้น้ำหนักกับตัวอัลกอริทึมสูงถึง 51% เมื่อเทียบกับอัตราการเรียนรู้จะอยู่ที่ 34% และอัลกอริทึมในการเรียนรู้อยู่ที่ 15% จะพบเลยว่าอัตราการเรียนรู้ในการศึกษานี้มีผลกระทบต่อค่าเป้าหมายของการเรียนรู้มากกว่าอัลกอริทึมในการเรียนรู้ และเมื่อเทียบกับการทดลองที่ 1 และจะพบว่าการปรับในส่วนของอัลกอริทึมในการเรียนรู้และอัตราการเรียนรู้จะส่งผลมากกว่าการปรับขนาดข้อมูลที่สุ่มมาเรียนรู้



แผนภาพที่ 4.23 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึม อัลกอริทึมในการเรียนรู้และอัตราการเรียนรู้ที่ส่งผลต่อรางวัลสะสม

4.3 การทดลองที่ 3 เปรียบเทียบประสิทธิภาพการซื้อขายจากการปรับการให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคต (Gamma)

การกำหนดค่าความสำคัญต่อรางวัลที่คาดหวังในอนาคต (Gamma) จะเป็นปัจจัยสำคัญอีกหนึ่งตัวในการกำหนดถึงพฤติกรรมของตัวแทนนอกเหนือจากฟังก์ชันการให้รางวัล เนื่องจากการวางแผนหาจุดเข้าซื้อและจุดปิดออร์เดอร์นั้น จำเป็นต้องมีการคาดการณ์หรือประมาณรางวัลที่คาดหวังในอนาคตที่จะได้รับ ดังนั้นการกำหนดค่าความสำคัญต่อรางวัลที่คาดหวังในอนาคตนั้นจะเป็นตัวบ่งบอกถึงระยะการคำนึงถึงรางวัลที่คาดหวังจะได้รับ โดยในการศึกษานี้เป็นการจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศของคู่สกุลเงิน EUR/USD ในระดับวันเท่านั้น ดังนั้นการกำหนดค่าความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่แตกต่างนั้น เพื่อหาว่าค่าความสำคัญต่อรางวัลที่คาดหวังในอนาคตใดเหมาะสมกับการซื้อขายคู่สกุลเงิน EUR/USD ที่มีความความผันผวนของราคาในระดับวันเท่านั้น

ในการทดลองนี้จะมีการแบ่งเป็น 5 การทดลองย่อย ซึ่งมีการกำหนดค่า Gamma ของแต่ละการทดลองคือ 0.1 0.3 0.5 0.7 และ 0.9 ตามลำดับ ซึ่งค่า Gamma แต่ละค่านี้จะส่งผลให้ Agent วางแผนไปในอนาคตประมาณ 2 วัน 3 วัน 6 วัน 12 วัน และ 43 วันตามลำดับ และมีการกำหนดค่าให้กับพารามิเตอร์ดังนี้ Batch Size มีค่าเป็น 128 และ Agent ใช้อัลกอริทึมในการเรียนรู้เป็น RMSprop ด้วยอัตราการเรียนรู้ที่ 0.001

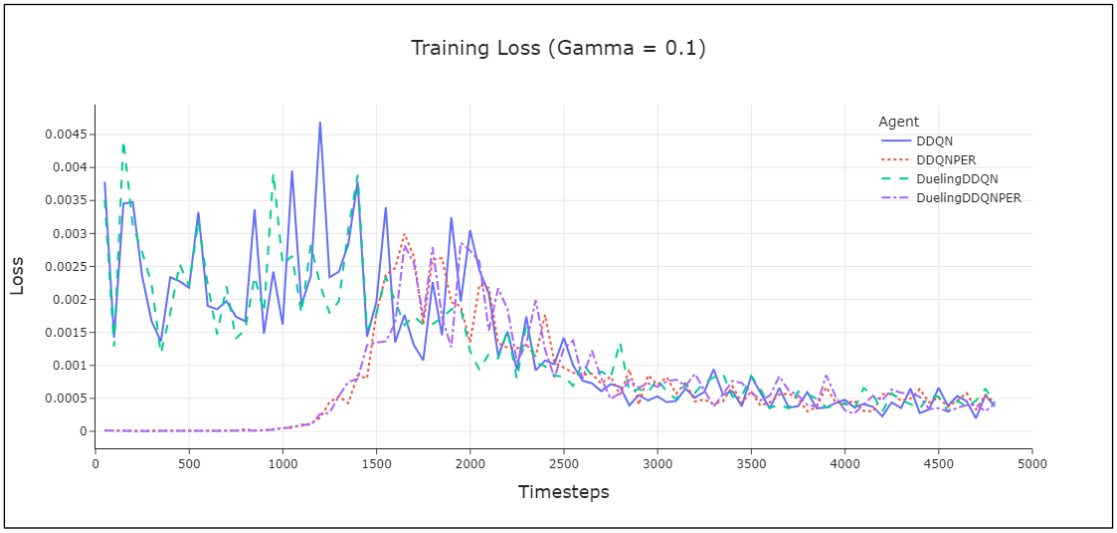
4.3.1 วัตถุประสงค์

เพื่อศึกษาพฤติกรรมและผลกระทบของความสำคัญของรางวัลที่คาดหวังในอนาคตว่าแต่ละ Agent สามารถทำงานได้ดีกับรางวัลที่คาดหวังในรูปแบบไหนและแต่ละ Agent สามารถทำกำไรในระยะเวลาที่แตกต่างกันหรือไม่

4.3.2 การทดลองที่ 3.1 กำหนด Gamma เท่ากับ 0.1

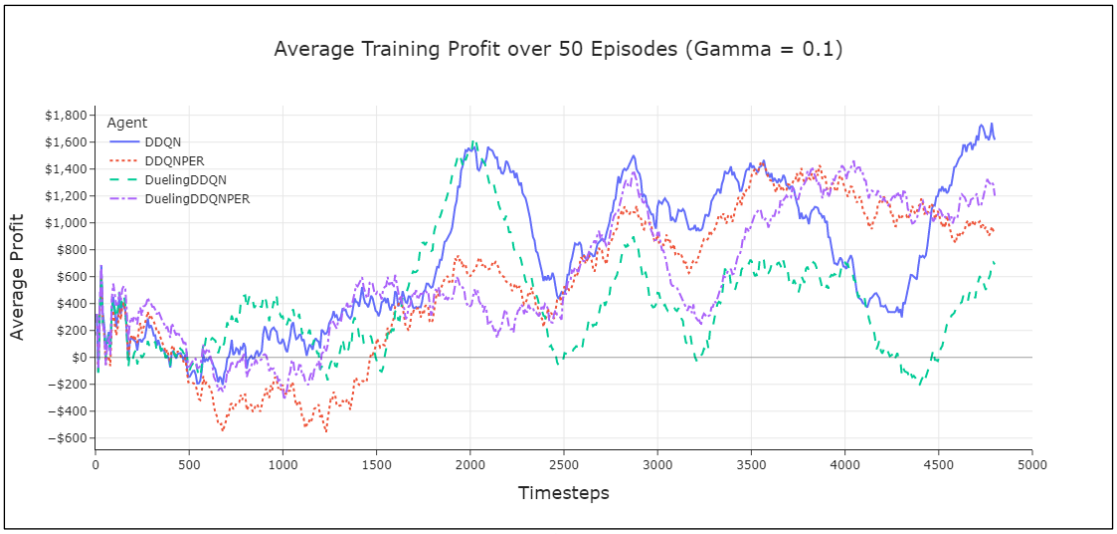
การทดลองนี้เป็นการทดลองที่ให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่น้อยที่สุดคือ 10% กล่าวคือในทุก ๆ timestep ค่ารางวัลในอนาคตจะถูกให้น้ำหนักเป็น 10% จากค่ารางวัลที่คาดหวังของสถานะถัดไป ดังนั้น Agent จะมีการคำนึงถึงการกระทำในอนาคตน้อยที่สุด ทำให้พฤติกรรมของ Agent อาจเกิดการซื้อขายที่รวดเร็วมากกว่าการทดลองอื่น ๆ

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เช่นเดียวกัน เมื่อผ่านการฝึกสอนไประยะหนึ่ง แต่จะพบว่าในช่วงแรกนั้นค่าผิดพลาดของ Agent ที่ใช้ Experience Replay ค่อนข้างมีความผันผวนสูง เพราะอาจจะเกิดการซื้อขายที่ถี่ จากการคาดการณ์รางวัลที่คาดหวังในระยะสั้นดังแผนภาพที่ 4.24



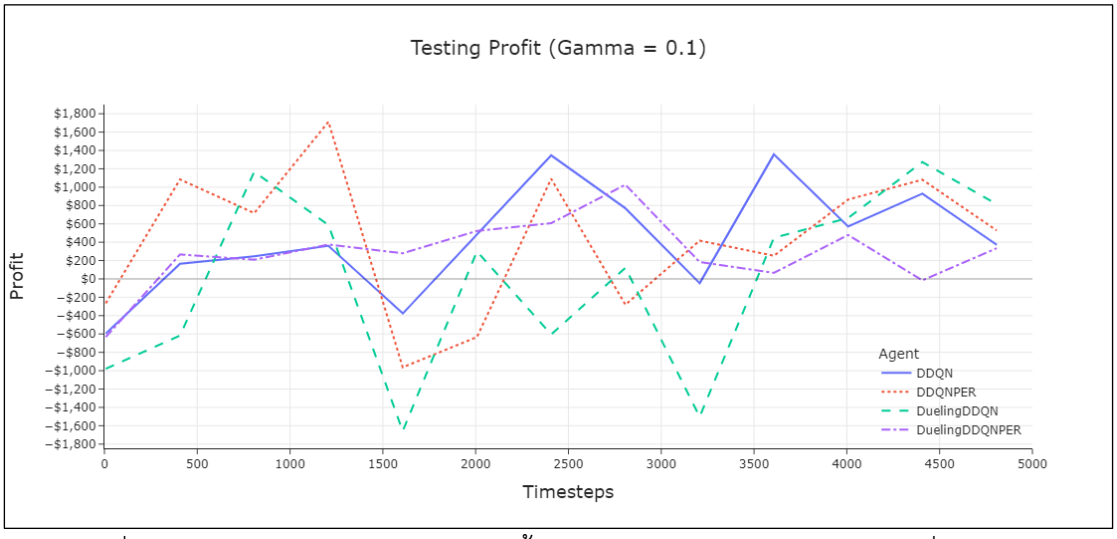
แผนภาพที่ 4.24 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10%

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent มีความผันผวนของกำไรที่สูงมาก ถึงแม้จะเป็นการเฉลี่ยมาจาก 50 Episode ก็ตาม เนื่องจากเกิดการซื้อขายที่บ่อยครั้ง และรอบการซื้อขายที่สั้นเพราะมีการคาดการณ์ในระยะสั้น ทำให้กำไรที่ได้จึงไม่มีความแน่นอน เพราะในชีวิตจริงหลังจากที่เราทำการซื้อขายในช่วงแรกที่ทำการเปิดออร์เดอร์ เรามักจะขาดทุนเป็นเรื่องปกติแต่เมื่อเวลาผ่านไปถ้าเราสามารถเข้าได้ถูกจุดเราก็จะสามารถทำกำไรได้ในอนาคต แต่เมื่อมีการกำหนดความสำคัญของรางวัลที่คาดหวังในอนาคตที่สั้นมาก ๆ จึงก่อให้เกิดความผันผวนเหล่านี้ อีกทั้งจะพบว่าทั้ง 4 Agent สามารถทำกำไรเฉลี่ยได้ไม่เกิน 1600 เหรียญเท่านั้น ซึ่ง Agent ที่สามารถทำกำไรได้ดีในขั้นตอนฝึกสอนก็ยังคงเป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay ดังแผนภาพที่ 4.25



แผนภาพที่ 4.25 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10%

ในทางกลับกัน Agent ทั้ง 4 ก็ยังคงทำกำไรได้ในขั้นตอนการทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อน แต่จะพบว่ากำไรสะสมที่ได้นั้นจะอยู่ที่ประมาณ 500 เหรียญเพราะเป็นการซื้อขายระยะสั้นและ Agent ที่สามารถทำกำไรได้ดีในขั้นตอนการทดสอบกลับเป็น Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Experience Replay ดังแผนภาพที่ 4.26



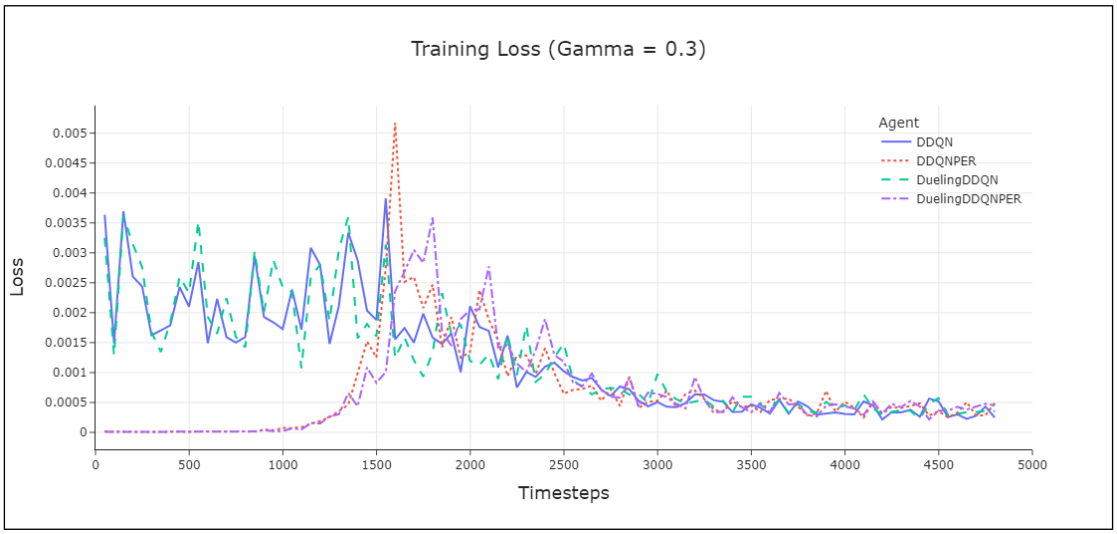
แผนภาพที่ 4.26 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 10%

4.3.3 การทดลองที่ 3.2 กำหนด Gamma เท่ากับ 0.3

การทดลองนี้เป็นการทดลองที่ให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่มากขึ้นมาเล็กน้อยคือ 30% กล่าวคือในทุก ๆ timestep ค่ารางวัลในอนาคตจะถูกให้น้ำหนักเป็น 30%

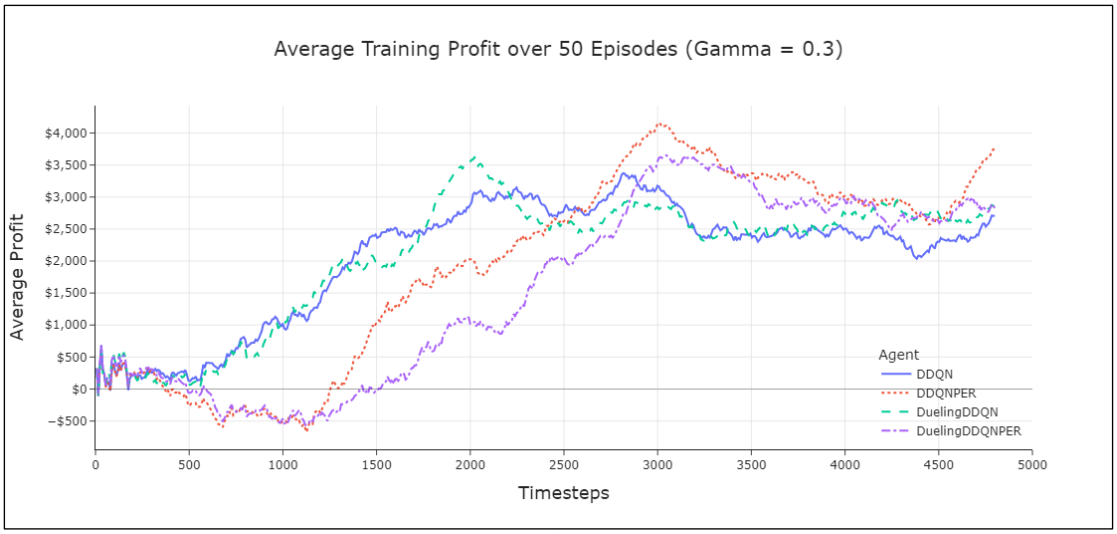
จากค่ารางวัลที่คาดหวังของสถานะถัดไป ดังนั้น Agent จะมีการคำนึงถึงการกระทำในอนาคตที่น้อย ทำให้พฤติกรรมของ Agent อาจเกิดการซื้อขายที่รวดเร็วมาก

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง แต่จะพบว่าในช่วงแรกนั้นค่าผิดพลาดของ Agent ที่ใช้ Experience Replay ค่อนข้างมีความผันผวนสูงเล็กน้อย แต่น้อยกว่าการทดลองที่ 3.1 เพราะอาจจะเกิดการซื้อขายที่ถี่ จากการคาดการณ์รางวัลที่คาดหวังในระยะสั้นดังแผนภาพที่ 4.27



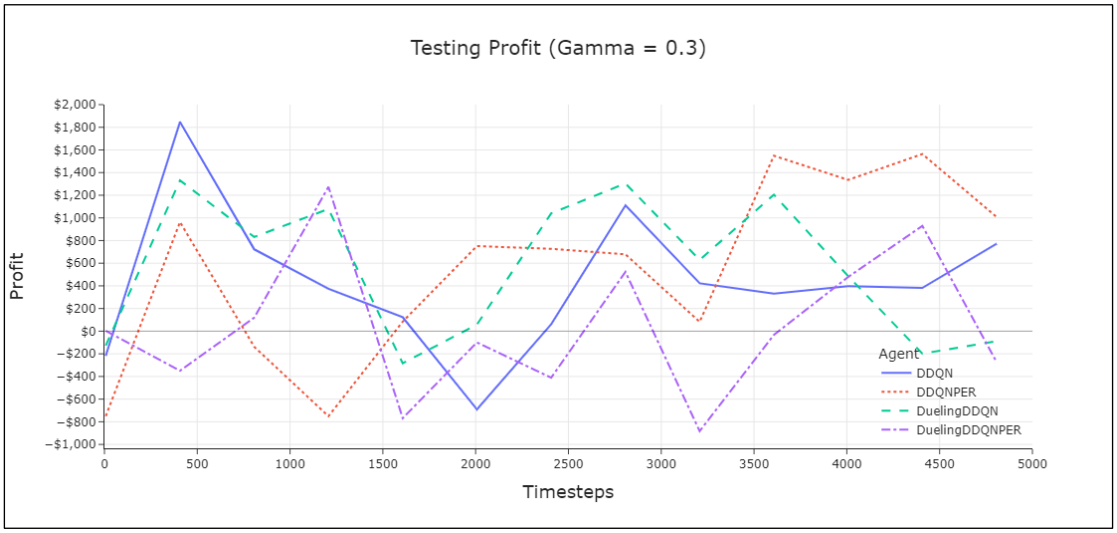
แผนภาพที่ 4.27 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30%

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent มีความผันผวนของกำไรที่นิ่งขึ้นอย่างมากเมื่อเทียบกับการทดลองที่ 3.1 หมายความว่าระยะการคำนึงถึงรางวัลที่คาดหวังในอนาคตที่ไกลขึ้นมาหน่อย ช่วยให้ Agent ตัดสินใจได้ดียิ่งขึ้น และอาจทำให้รอบการซื้อขายมีระยะที่ยาวมากขึ้น ทำให้กำไรที่ได้จึงมีความเสถียรมากยิ่งขึ้นและจะพบว่าทั้ง 4 Agent สามารถทำกำไรเฉลี่ยได้ในช่วง 2500 - 4000 เหรียญจากการซื้อขายที่ยาวมากขึ้น ซึ่ง Agent ที่ทำงานได้ดีกลับเป็น Agent ที่ใช้ Prioritized Experience Replay มากกว่า Agent ที่ใช้ Experience Replay ซึ่งแตกต่างจากการทดลองอื่น ๆ ที่ผ่านมาดังแผนภาพที่ 4.28



แผนภาพที่ 4.28 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30%

ในทางกลับกันในขั้นตอนการทดสอบกลับพบว่า Agent ที่ใช้โครงสร้างอัลกอริทึมเป็น DuelingDDQN ไม่สามารถทำกำไรในขั้นตอนการทดสอบได้ทั้ง 2 แบบถึงแม้จะมีการใช้ถังเก็บประสบการณ์ที่แตกต่างกัน แต่ Agent ที่ใช้โครงสร้างอัลกอริทึม DDQN นั้นสามารถทำกำไรได้ทั้ง 2 แบบ ดังนั้นถ้าหากเราต้องการซื้อขายในระยะที่ค่อนข้างสั้นเราควรจะใช้โครงสร้างอัลกอริทึม DDQN มากกว่า DuelingDDQN ดังแผนภาพที่ 4.29

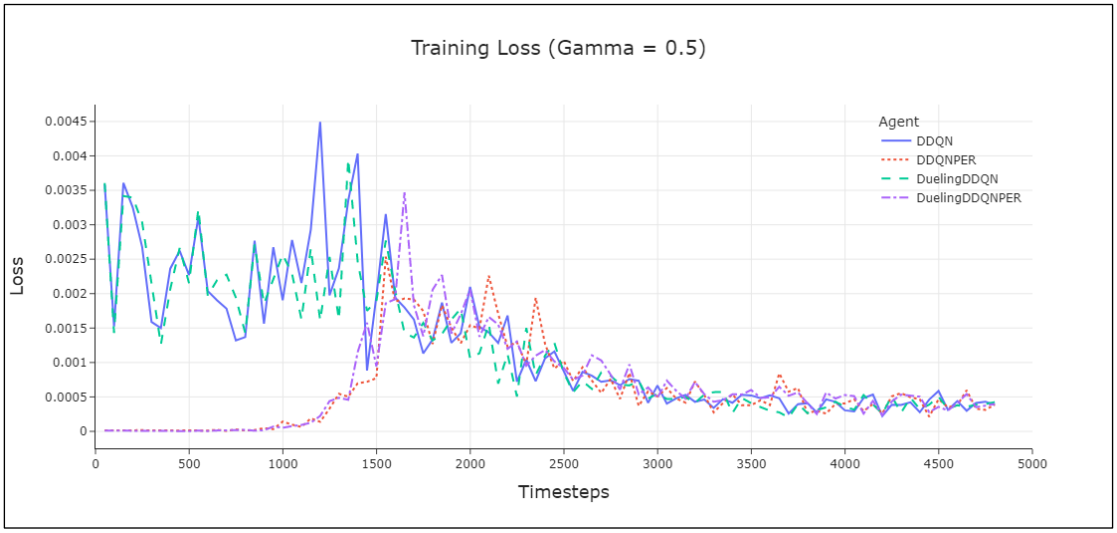


แผนภาพที่ 4.29 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 30%

4.3.4 การทดลองที่ 3.3 กำหนด Gamma เท่ากับ 0.5

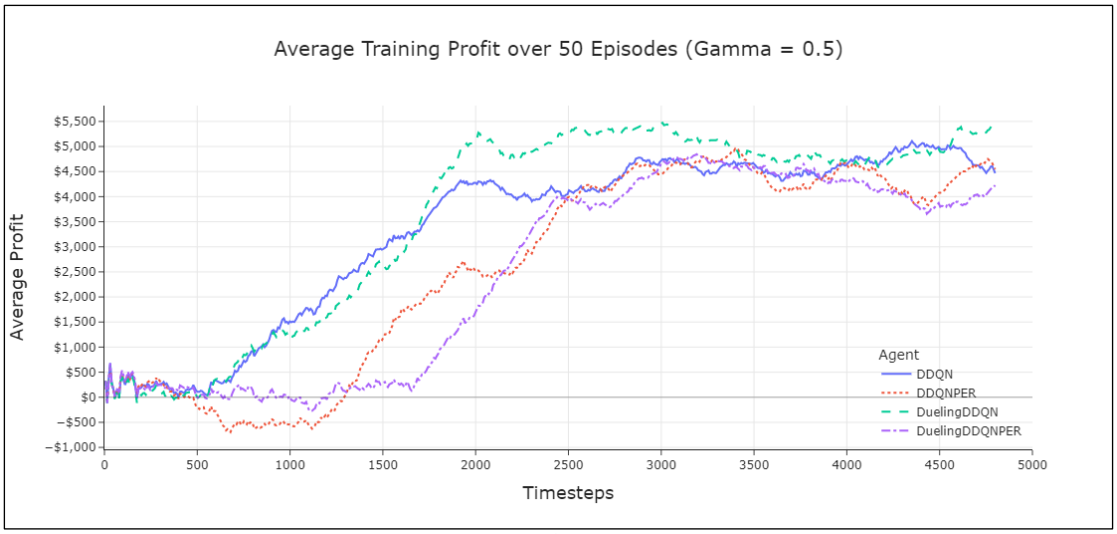
การทดลองนี้เป็นการทดลองที่ให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ระดับปานกลางคือ 50% กล่าวคือในทุก ๆ timestep ค่ารางวัลในอนาคตจะถูกให้น้ำหนักเป็น 50% จากค่ารางวัลที่คาดหวังของสถานะถัดไป ดังนั้น Agent จะมีการคำนึงถึงการกระทำในอนาคตในระดับปานกลาง ทำให้พฤติกรรมของ Agent อาจเกิดการซื้อขายที่ไม่สั้นไม่ยาวจนเกินไป

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง แต่จะพบว่าในช่วงแรกนั้นค่าผิดพลาดของ Agent ที่ใช้ Experience Replay ค่อนข้างมีความผันผวนสูงเล็กน้อยในช่วง timesteps ที่ 1000 ถึง 1500 แต่หลังจากนั้นก็มีการลู่เข้าปกติดังแผนภาพที่ 4.30



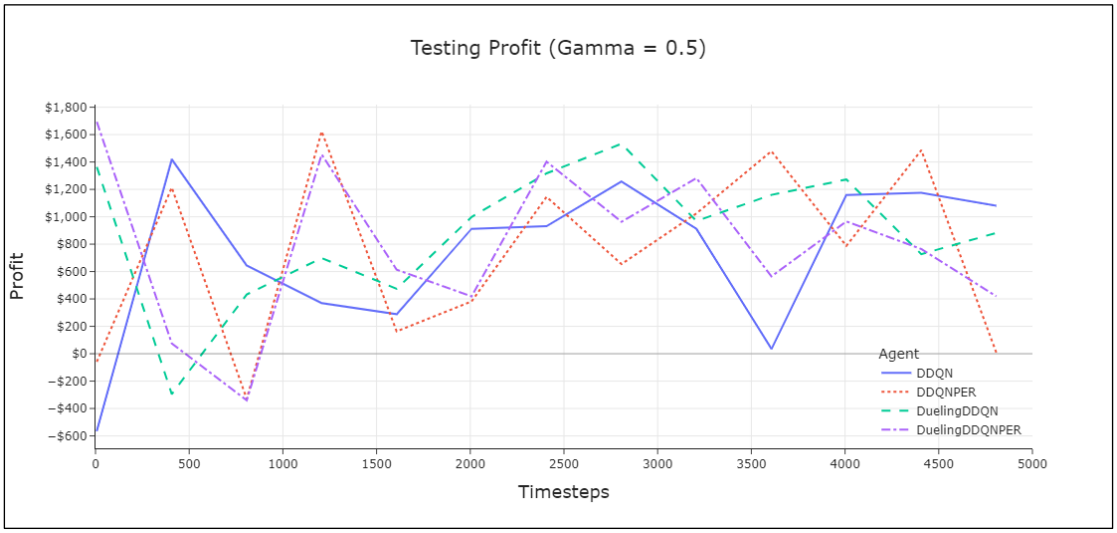
แผนภาพที่ 4.30 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50%

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent มีความผันผวนของกำไรที่นิ่งและมีแนวโน้มคล้ายกับการทดลองที่ 3.2 แต่สิ่งที่แตกต่างกันคือทั้ง 4 Agent สามารถทำกำไรเฉลี่ยได้มากกว่าการทดลองที่ 3.1 และ 3.2 ซึ่งสามารถทำกำไรเฉลี่ยได้สูงถึง 4000 - 5500 เหรียญจากการซื้อขายที่ยาวมากขึ้นในขั้นตอนการฝึกสอน ซึ่ง Agent ที่ทำงานได้ดีกลับเป็น Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Experience Replay ซึ่งแตกต่างจากการทดลองอื่น ๆ ที่ผ่านมาดังแผนภาพที่ 4.31



แผนภาพที่ 4.31 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50%

ในขั้นตอนการทดสอบกลับพบว่า Agent ที่ใช้ Prioritized Experience Replay สามารถทำกำไรได้น้อยกว่า Agent ที่ใช้ Experience Replay แต่ทั้ง 4 Agent ก็ยังสามารถทำกำไรได้โดยไม่ขาดทุน ดังนั้นถ้าต้องการซื้อขายในระยะกลาง Agent ควรใช้ Experience Replay มากกว่า Prioritized Experience Replay ดังแผนภาพที่ 4.32



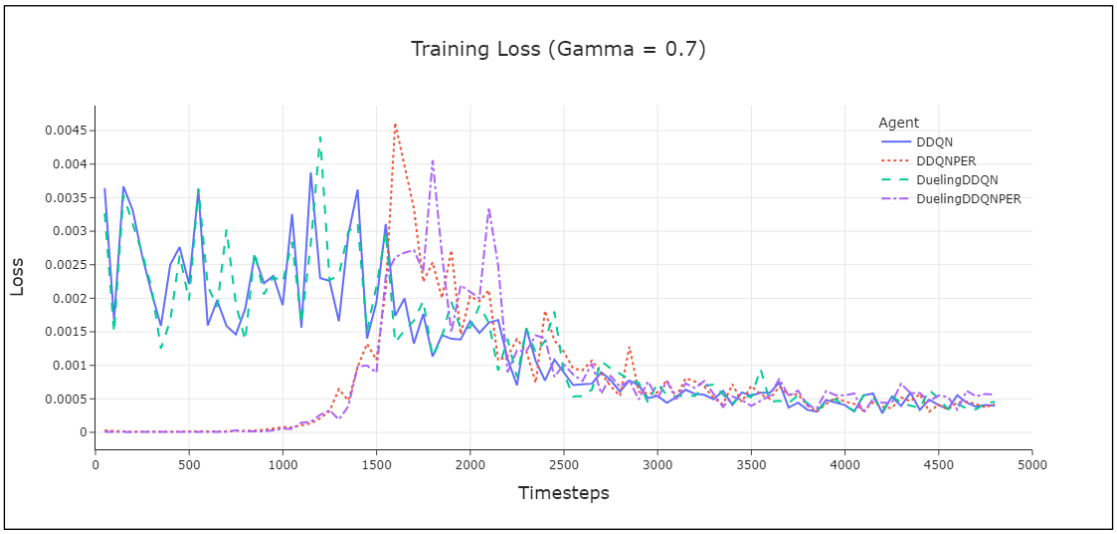
แผนภาพที่ 4.32 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 50%

4.3.5 การทดลองที่ 3.4 กำหนด Gamma เท่ากับ 0.7

การทดลองนี้เป็นการทดลองที่ให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ระดับค่อนข้างมากคือ 70% กล่าวคือในทุก ๆ timestep ค่ารางวัลในอนาคตจะถูกให้น้ำหนักเป็น 70%

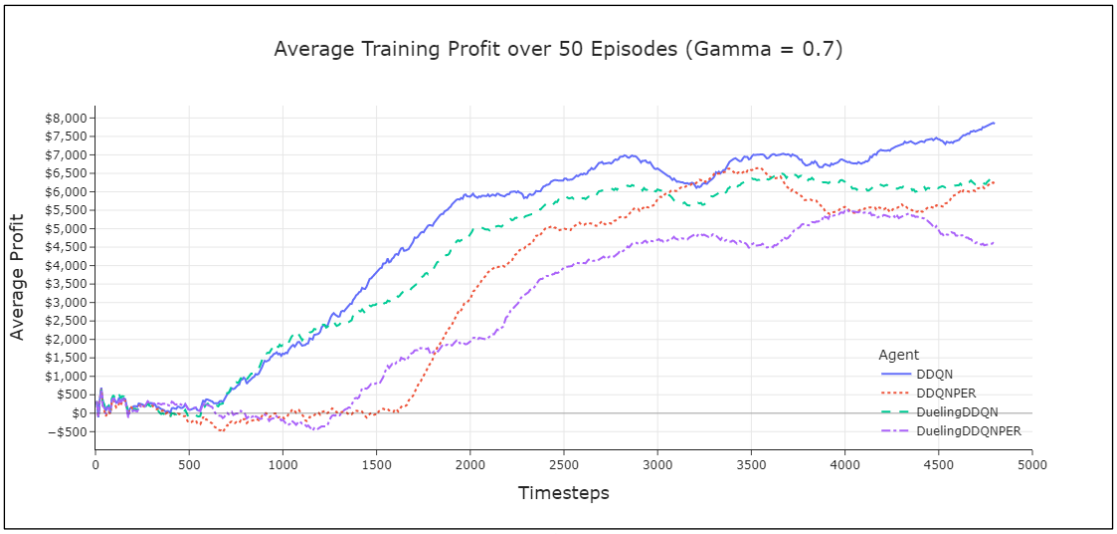
จากค่ารางวัลที่คาดหวังของสถานะถัดไป ดังนั้น Agent จะมีการคำนึงถึงการกระทำในอนาคตในระดับค่อนข้างยาว ทำให้พฤติกรรมของ Agent อาจเกิดการซื้อขายที่น้อยลงและมีการถือที่ยาวขึ้น

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง แต่จะพบว่าในช่วงแรกนั้นค่าผิดพลาดของ Agent ค่อนข้างมีความผันผวนสูงเล็กน้อยในช่วง timesteps ที่ 1000 ถึง 2000 แต่หลังจากนั้นก็มีการลู่เข้าปกติดังแผนภาพที่ 4.33



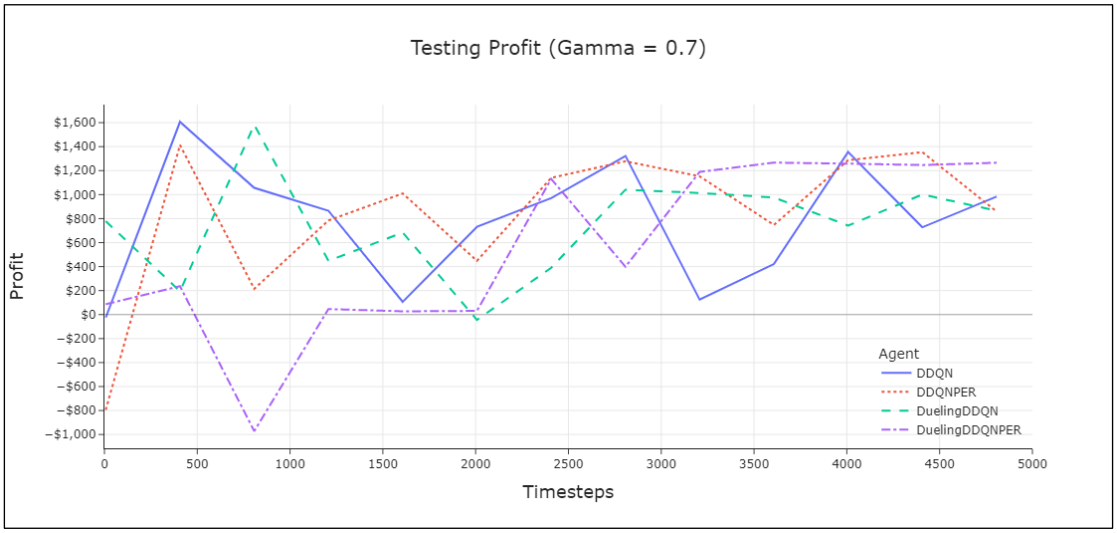
แผนภาพที่ 4.33 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70%

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent มีความผันผวนของกำไรที่นิ่งและมีแนวโน้มคล้ายกับการทดลองที่ 3.2 และการทดลองที่ 3.3 แต่สิ่งที่แตกต่างจะพบว่าทั้ง 4 Agent สามารถทำกำไรเฉลี่ยได้มากกว่าการทดลองที่ 3.1 - 3.3 ซึ่งสามารถทำกำไรเฉลี่ยได้สูงถึง 4500 - 8000 เหรียญจากการซื้อขายที่ยาวมากขึ้นในขั้นตอนการฝึกสอน ซึ่ง Agent ที่ทำงานได้ดีเป็น Agent ที่ใช้อัลกอริทึม DDQN และใช้ Experience Replay ดังแผนภาพที่ 4.34



แผนภาพที่ 4.34 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70%

ในทางกลับกันในขั้นตอนการทดสอบกลับพบว่า Agent ที่ใช้ Prioritized Experience Replay สามารถทำกำไรได้ดีกว่า Agent ที่ใช้ Experience Replay ซึ่ง Agent ที่ทำงานได้ดีที่สุดในขั้นตอนการทดสอบคือ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay ที่สามารถทำกำไรสะสมในขั้นตอนทดสอบได้สูงที่สุดถึง 1200 เหรียญเมื่อเทียบกับการทดลองที่ผ่านมาทั้งหมด ดังแผนภาพที่ 4.35

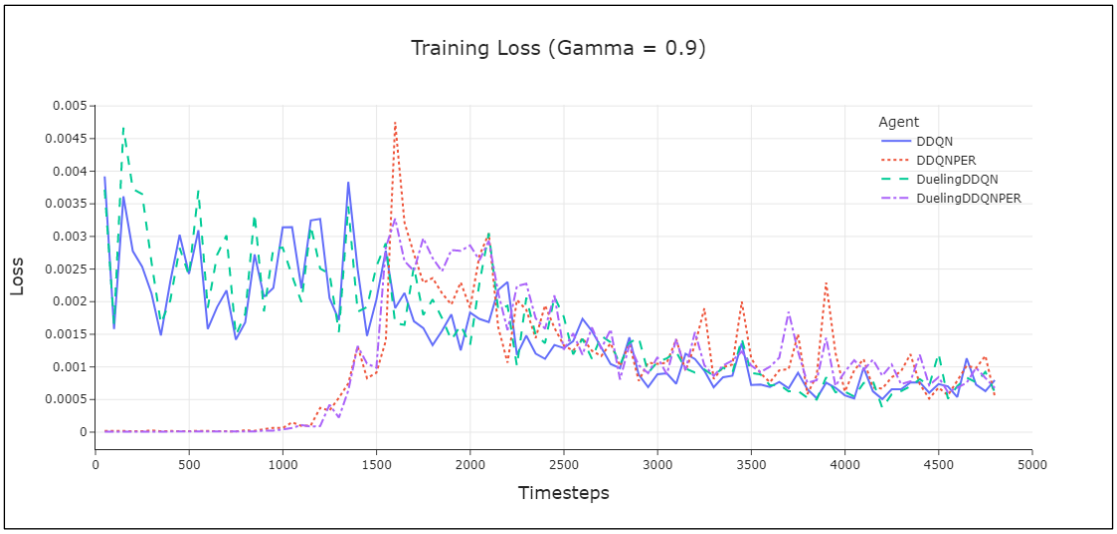


แผนภาพที่ 4.35 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 70%

4.3.6 การทดลองที่ 3.5 กำหนด Gamma เท่ากับ 0.9

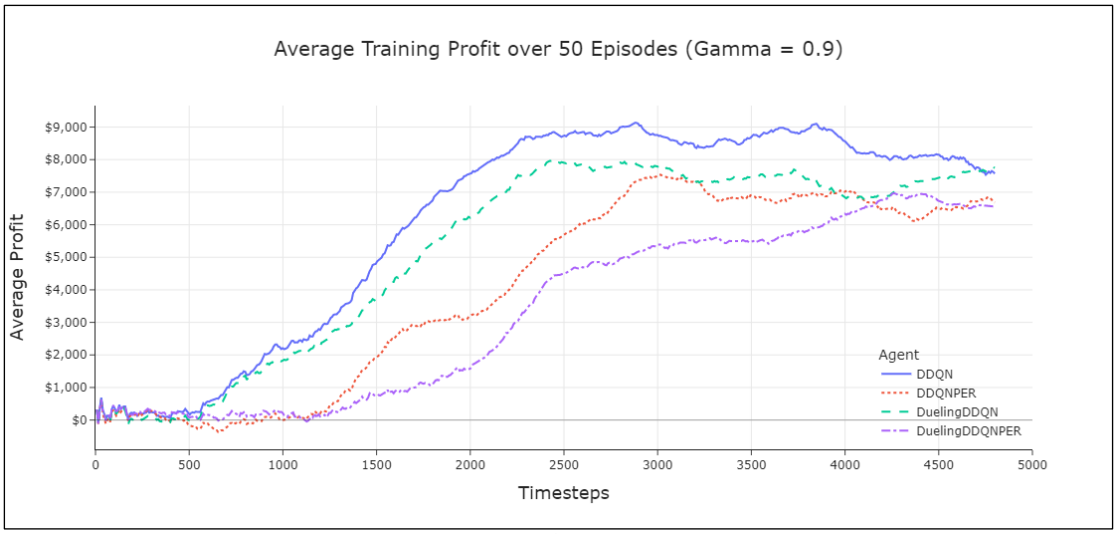
การทดลองนี้เป็นการทดลองที่ให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ระดับมากคือ 90% กล่าวคือในทุก ๆ timestep ค่ารางวัลในอนาคตจะถูกให้น้ำหนักเป็น 90% จากค่ารางวัลที่คาดหวังของสถานะถัดไป ดังนั้น Agent จะมีการคำนึงถึงการกระทำในอนาคตในระดับยาว ทำให้พฤติกรรมของ Agent อาจเกิดการซื้อขายที่น้อยและมีการถือที่ยาวมาก

จากการทดลองพบว่า ทั้ง 4 Agent มีค่าผิดพลาดลู่เข้าที่จุดเดียวกันคือประมาณ 0.0005 เมื่อผ่านการฝึกสอนไประยะหนึ่ง แต่จะพบว่าในช่วงแรกนั้นค่าผิดพลาดของ Agent ค่อนข้างมีความผันผวนสูงในช่วง timesteps ที่ 0 ถึง 2000 อาจมาจากการที่เราคำนึงถึงอนาคตที่ไกลเกินไป เพราะพฤติกรรมของตลาดเหล่านี้มักมีความผันผวนสูงอยู่แล้ว ทำให้บางครั้งการคาดการณ์ที่ไกลเกินไปอาจไม่ส่งผลดีแต่กลับกลายเป็นผลร้ายแทนได้ แต่หลังจากนั้นก็มีการลู่เข้าปกติดังแผนภาพที่ 4.36



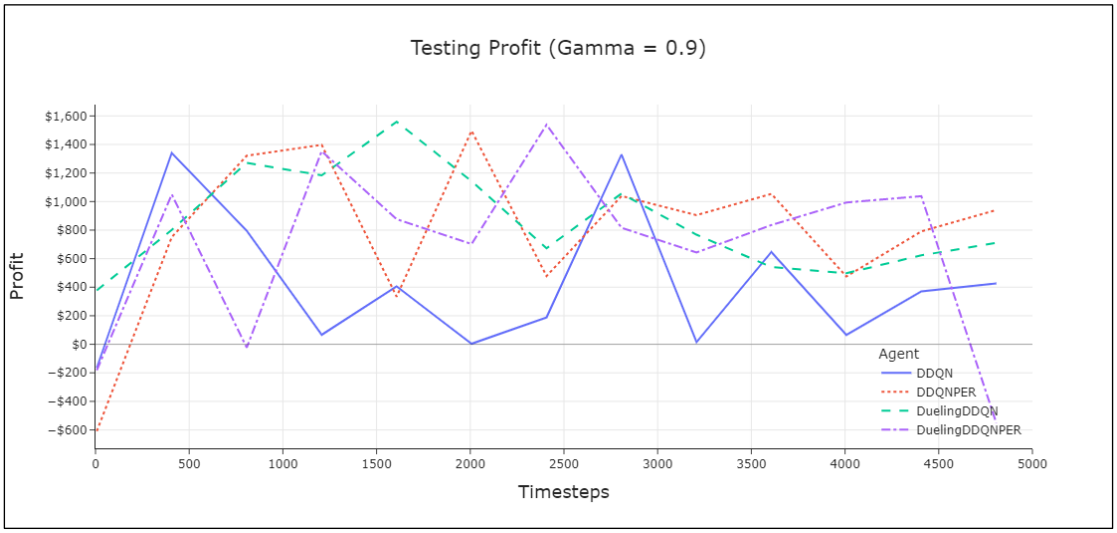
แผนภาพที่ 4.36 กราฟแสดงค่าผิดพลาดระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90%

จากนั้นเมื่อดูกราฟเปรียบเทียบระหว่างปริมาณกำไรที่ Agent ทำได้ในขั้นตอนฝึกสอนจะพบว่าทั้ง 4 Agent มีความผันผวนของกำไรที่นิ่งและมีแนวโน้มค่อนข้างเพิ่มขึ้นจนกระทั่ง timesteps ที่ 3000 จะพบว่าแนวโน้มของกำไรที่ทำได้ค่อยข้างลดลงจนเฉลี่ยอยู่ที่ประมาณ 6000 – 8000 เหรียญ เมื่อเทียบกับการทดลองที่ 3.4 จะพบว่าสามารถทำกำไรเฉลี่ยได้ไม่แตกต่างกันเท่าไร ซึ่ง Agent ที่ทำงานได้ดีกลับเป็น Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Experience Replay ซึ่งแตกต่างจากการทดลองอื่น ๆ ที่ผ่านมาดังแผนภาพที่ 4.37



แผนภาพที่ 4.37 กราฟแสดงปริมาณกำไรเฉลี่ยในขั้นตอนฝึกสอนระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90%

ในส่วนของขั้นตอนการทดสอบนั้นกลับพบว่า Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ Prioritized Experience Replay สามารถทำกำไรสะสมได้ดีในการทดลองที่ 3.4 แต่กลับทำได้แย่มากในการทดลองนี้ จากการณ์คาดการณ์ที่ยาวเกินไป อีกทั้ง 3 Agent ที่เหลือยังทำกำไรได้ไม่เกิน 1000 เหรียญ ดังแผนภาพที่ 4.38



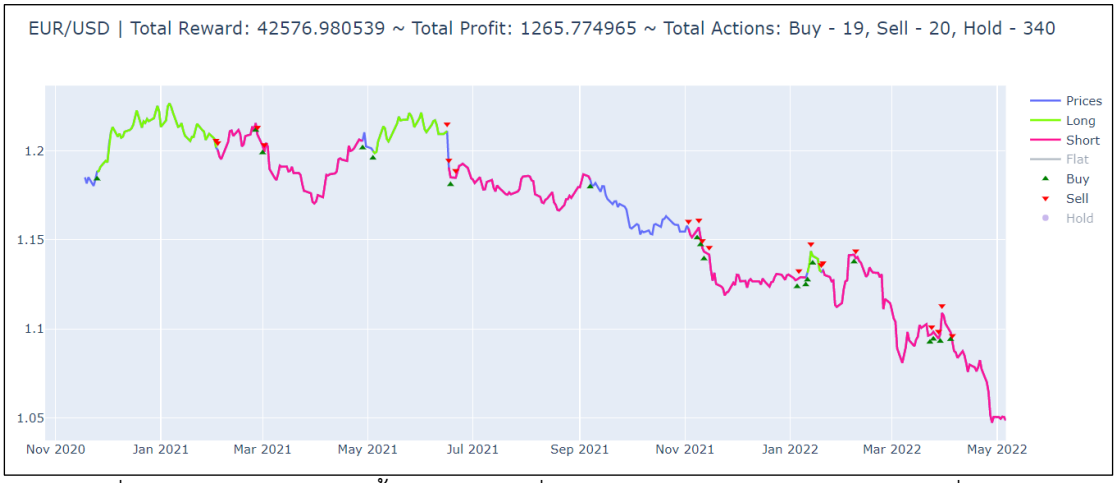
แผนภาพที่ 4.38 กราฟแสดงปริมาณกำไรในขั้นตอนทดสอบระหว่าง 4 Agent เมื่อมีการกำหนดความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ 90%

4.3.7 สรุปผลการทดลอง

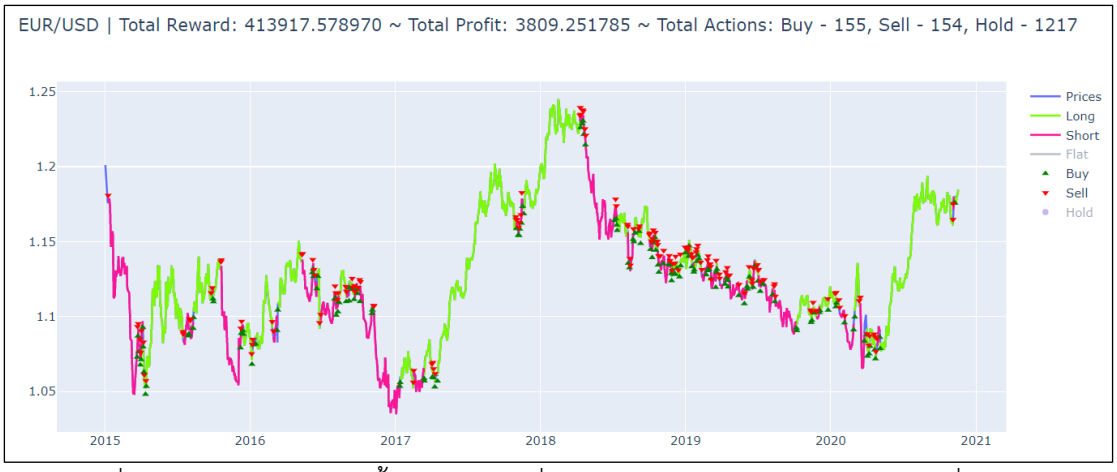
จากการทดลองทั้ง 5 การทดลองพบว่า Agent ทั้ง 4 ประเภทสามารถทำกำไรโดยเฉลี่ยได้พอ ๆ กันทั้ง 4 Agent ไม่ว่าจะเป็นขั้นตอนการฝึกสอนและขั้นตอนการทดสอบ ซึ่งปัจจัย

สำคัญที่ส่งผลต่อพฤติกรรมของ Agent คือการกำหนดความสำคัญของรางวัลที่คาดหวังในอนาคต ว่าควรคำนึงถึงในระยะเท่าใด ซึ่งจากการทดลองจะพบว่ายิ่งกำหนดความสำคัญมากเพียงใด Agent ก็จะสามารถทำกำไรในขั้นตอนการฝึกสอนได้มากขึ้นเท่านั้นเพราะมันสามารถคาดการณ์รางวัลที่คาดหวังในอนาคตที่จะได้รับได้ไกลมากขึ้น และเมื่อมันสามารถคาดการณ์ได้ไกลมากเท่าไรมันก็สามารถวางแผนเพื่อที่จะทำกำไรจากการคาดการณ์เหล่านั้นได้มากยิ่งขึ้น แต่ในทางกลับกันเมื่อนำมาทดสอบกับข้อมูลที่ Agent ไม่เคยเห็นมาก่อน ซึ่งเป็นข้อมูลราคา EUR/USD ในช่วงขาลงเนื่องจากสถานการณ์ COVID-19 การที่ Agent คาดการณ์ระยะที่ไกลเกินไป เช่น Gamma เป็น 0.9 ทำให้อัลกอริทึมที่สามารถทำงานได้ดีเมื่อ Gamma เป็น 0.7 กลับได้ผลลัพธ์ที่ตรงกันข้ามในทันที เนื่องจากสถานะที่ Agent ยังไม่เคยพบเจอและสถานการณ์ของตลาดที่มีความผันผวนสูง การคาดการณ์ที่ไกลเกินไปมักไม่มีความแน่นอนสูงมากยิ่งขึ้น

ดังนั้นในการทดลองนี้สามารถสรุปผลได้ว่า Gamma มีผลต่อพฤติกรรมของ Agent อย่างมาก เมื่อเทียบกับแต่ละอัลกอริทึมเพราะโดยส่วนใหญ่แต่ละอัลกอริทึมสามารถทำกำไรเฉลี่ยได้ใกล้เคียงกัน ฉะนั้นเราต้องทำการเลือกค่า Gamma ให้เหมาะสมกับแต่ละตลาดที่มีความผันผวนที่แตกต่างกัน ซึ่งในการทดลองนี้ Gamma ที่เหมาะสมที่สุดเมื่อทดสอบกับข้อมูลที่ไม่เคยเห็นมาก่อนคือ 0.7 เมื่อนำ Agent มาทดสอบการซื้อขายกับข้อมูลที่ไม่เคยเห็นมาก่อนจะได้ดังแผนภาพที่ 4.39 ซึ่งจะแสดงให้เห็นถึงการซื้อขายและระยะเวลาของการถือต่อของแต่ละการซื้อขาย และเมื่อทดสอบกับข้อมูลที่ใช้ฝึกสอนจะได้แสดงแผนภาพที่ 4.40 จะพบว่ามีระยะในการถือที่ค่อนข้างยาว เพราะเป็นสถานะที่ถูกเห็นบ่อยหรือถูกฝึกสอนมานั้นเอง จึงสามารถทำการซื้อขายได้ดี

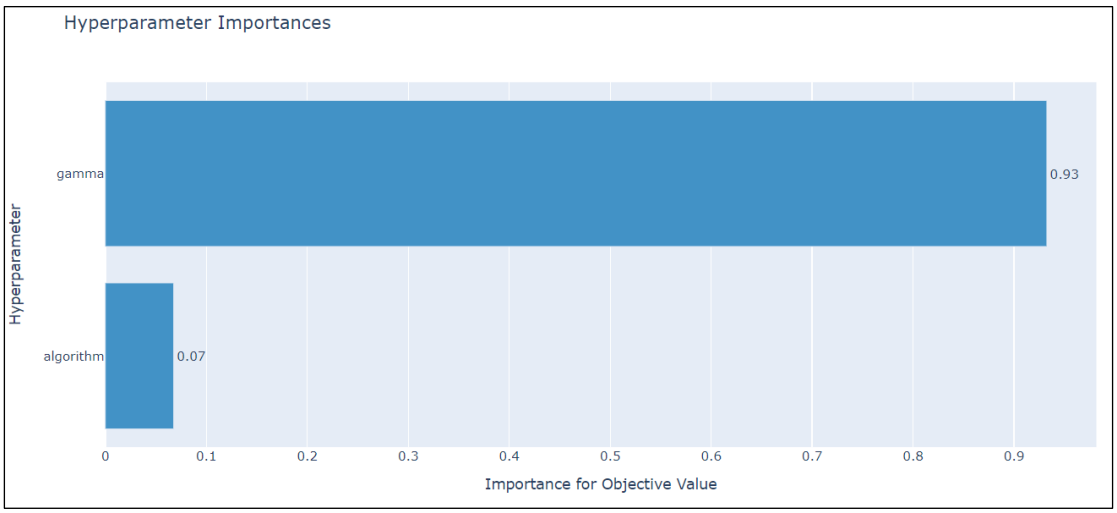


แผนภาพที่ 4.39 กราฟแสดงการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศของ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ถังเก็บข้อมูลที่มีการจัดลำดับความสำคัญและมีการกำหนดค่า Gamma ที่ 0.7 กับข้อมูลทดสอบ



แผนภาพที่ 4.40 กราฟแสดงการซื้อขายแลกเปลี่ยนเงินตราต่างประเทศของ Agent ที่ใช้อัลกอริทึม DuelingDDQN และใช้ถังเก็บข้อมูลที่มีการจัดลำดับความสำคัญและมีการกำหนดค่า Gamma ที่ 0.7 กับข้อมูลฝึกสอน

ดังนั้นเมื่อนำทั้ง 5 การทดลองมาคำนวณหาความสำคัญของ Hyperparameter ต่อรางวัลสะสมด้วยวิธีการ Functional ANOVA (fANOVA) จะให้ผลลัพธ์ดังแผนภาพที่ 4.41 จะพบว่าค่าความสำคัญของ Hyperparameter นั้นให้น้ำหนักกับค่า Gamma สูงถึง 93% เมื่อเทียบกับอัลกอริทึมที่คิดเป็นเพียง 7% เท่านั้น ซึ่งจะเห็นว่าการทดลองนี้เป็นการทดลองเดียวที่อัลกอริทึมมีความสำคัญน้อยกว่า Hyperparameter ตัวอื่น เมื่อเทียบกับการทดลองอื่น ๆ ที่ผ่านมา



แผนภาพที่ 4.41 กราฟแสดงค่าความสำคัญระหว่างอัลกอริทึมและความสำคัญต่อรางวัลที่คาดหวังในอนาคตที่ส่งผลต่อรางวัลสะสม

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

การศึกษานี้เป็นการศึกษาเพื่อเปรียบเทียบประสิทธิภาพของการซื้อขายคู่สกุลเงินระหว่างตัวแทนที่ใช้อัลกอริทึมที่แตกต่างกันและมีค่าพารามิเตอร์ที่ถูกควบคุมเหมือนกัน โดยใช้ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ที่มีการจัดเตรียมและประมวลผลค่าชี้วัดเชิงเทคนิคเรียบร้อยแล้วมาฝึกสอน ทดสอบการทำงานและเปรียบเทียบประสิทธิภาพการทำงานของตัวแทน ซึ่งในบทนี้จะเป็นข้อสรุปของผลการดำเนินงานและข้อเสนอแนะในการพัฒนาต่อไป

5.1 สรุปผลการวิจัย

งานวิจัยนี้เป็นการนำเสนอผลการดำเนินงานของการซื้อขายคู่สกุลเงินระหว่างตัวแทนที่ใช้อัลกอริทึมที่แตกต่างกันและมีค่าพารามิเตอร์ที่ถูกควบคุมเหมือนกัน โดยใช้ข้อมูลอัตราแลกเปลี่ยนรายวันของคู่สกุลเงิน EUR/USD ที่มีการจัดเตรียม ประมวลผลค่าชี้วัดเชิงเทคนิค และปรับช่วงของข้อมูลมาแล้ว โดยข้อมูลจะถูกนำมาแบ่งเป็น 2 ชุดคือ 1. ชุดข้อมูลสำหรับการฝึกสอนและ 2. ชุดข้อมูลสำหรับการทดสอบ เพื่อนำไปสร้างสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศสำหรับการฝึกสอนและสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศสำหรับการทดสอบ ในการพัฒนาสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศได้มีการออกแบบการกระทำและตำแหน่งที่ตัวแทนสามารถทำได้ อีกทั้งได้ทำการออกแบบฟังก์ชันการให้รางวัลขึ้นมาใหม่เพื่อให้ตัวแทนสามารถเปิดออร์เดอร์ได้ทั้งขาขึ้น (Long) และขาลง (Short) ของราคาคู่สกุลเงิน รวมทั้งยังมีการเพิ่มบทลงโทษเพียงเล็กน้อยที่เปรียบเสมือนเป็นตัวกระตุ้นให้ตัวแทนไม่ทำการรอเข้าซื้อขาย (Flat) นานเกินไป และบทลงโทษขั้นร้ายแรงที่สุดที่ไม่ต้องการให้ตัวแทนนั้นมีการเปิดออร์เดอร์ซ้ำกับตำแหน่งปัจจุบันของตัวแทนเพื่อให้ตัวแทนหาจุดเปิดออร์เดอร์และปิดออร์เดอร์ได้เหมาะสมที่สุดแทน ในส่วนของฟังก์ชันการให้รางวัลนี้ส่งผลให้เราสามารถฝึกสอนตัวแทนให้มีพฤติกรรมที่ตรงตามที่เราต้องการได้ แต่อีกสิ่งหนึ่งที่ขาดไม่ได้คือการออกแบบสถานะหรือข้อมูลที่ตัวแทนจะใช้ในการตัดสินใจในช่วงเวลานั้น ๆ จากการออกแบบที่มีการนำข้อมูลราคาและค่าชี้วัดเชิงเทคนิคย้อนหลังช่วยให้ตัวแทนสามารถตัดสินใจได้ดียิ่งขึ้น แต่ถ้าขาดตำแหน่งปัจจุบันของตัวแทนจะส่งผลให้ตัวแทนไม่สามารถรับรู้ได้เลยว่า ณ ปัจจุบันตัวแทนได้การเปิดออร์เดอร์อยู่หรือไม่ ดังนั้นการนำตำแหน่งปัจจุบันของตัวแทนเข้ามาเป็นส่วนหนึ่งของสถานะช่วยให้ตัวแทนสามารถประพฤติตัวได้ตรงตามเป้าหมายได้มากยิ่งขึ้น จากนั้นได้ทำการออกแบบและพัฒนาตัวแทนขึ้นมา 4 ตัวแทนที่มีอัลกอริทึมที่แตกต่างกันและมีถังเก็บข้อมูลประสบการณ์ที่แตกต่างกันในการฝึกสอนและทดสอบกับสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศ โดยทำการเปรียบเทียบประสิทธิภาพของ

ตัวแทนในการปรับค่าพารามิเตอร์ในรูปแบบต่าง ๆ เพื่อหาตัวแทนที่สามารถทำกำไรได้ดีที่สุดในสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศสำหรับการทดสอบ

จากการออกแบบฟังก์ชันการให้รางวัลและการทดลองทั้ง 3 การทดลองพบว่า ตัวแทนทั้ง 4 สามารถทำงานได้ดีในขั้นตอนการฝึกสอนหรือสามารถทำกำไรได้มากกับข้อมูลที่ทำการเรียนรู้ และเมื่อนำมาปรับใช้กับข้อมูลที่ไม่เคยเห็นมาก่อนหรือข้อมูลสำหรับการทดสอบ ถึงแม้ว่าสภาวะตลาดในการทดสอบจะค่อนข้างผันผวนจากเหตุการณ์ COVID-19 แต่ส่วนใหญ่ตัวแทนทั้ง 4 แบบก็ยังสามารถทำกำไรได้ดีกับสถานการณ์ที่ไม่เคยพบเห็นมาก่อน และเนื่องจากตัวแทนทั้ง 4 ประเภทนี้มีการให้ความสำคัญต่อรางวัลที่คาดหวังในอนาคต เพราะฉะนั้นการปรับจูนค่าความสำคัญนี้มีผลเป็นอย่างมากต่อพฤติกรรมของตัวแทน เพื่อให้ตัวแทนสามารถทำงานได้ตรงตามเป้าหมายที่เราทำการออกแบบไว้ในฟังก์ชันการให้รางวัล ฉะนั้นการหาค่าความสำคัญต่อรางวัลที่คาดหวังในอนาคตนั้นเป็นค่าที่ควรปรับเป็นอย่างยิ่ง ว่าในโจทย์ปัญหาที่เรากำลังจะแก้ไขนั้นเหมาะสมกับค่าความสำคัญที่ระดับใด

ข้อสรุปจากการทดลองในครั้งนี้ ตัวแทนที่ใช้อัลกอริทึม DuelingDDQN และใช้ถังประสบการณ์ที่มีการจัดลำดับความสำคัญ ที่มีการกำหนดค่าน้ำหนักต่อรางวัลคาดหวังในอนาคตที่ระดับ 70% สามารถทำกำไรได้สูงที่สุดกับสภาพแวดล้อมจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศสำหรับการทดสอบที่มีราคาและความผันผวนของคู่สกุลเงิน EUR/USD ในระดับวันเท่านั้น ซึ่งสามารถทำกำไรได้สูงเกือบถึง 1300 เหรียญเมื่อเทียบกับตัวแทนที่ใช้อัลกอริทึมแบบอื่นและการปรับจูนในรูปแบบอื่น ๆ ดังนั้นเราสามารถนำตัวแทนนี้ไปประยุกต์ใช้ในชีวิตประจำวัน เพื่อเป็นตัวเลือกหนึ่งที่ช่วยในการตัดสินใจหาจุดเข้าซื้อขายได้ดียิ่งขึ้น

5.2 ขอบเขตและข้อจำกัด

1. สภาพแวดล้อมยังไม่รองรับการเข้าซื้อหรือเข้าขายซ้ำ ควรมีการปรับปรุงเพิ่มเติมในกระบวนการเฉลี่ยของราคา
2. สภาพแวดล้อมยังไม่สามารถจำลองตลาดแลกเปลี่ยนเงินตราต่างประเทศที่แท้จริงได้ เพราะในความเป็นจริงตลาดแลกเปลี่ยนเงินตราต่างประเทศจะมีการคิดคำนวณค่าธรรมเนียมในรูปแบบต่าง ๆ การตั้งจุดขาดทุนหรือจุดปิดออร์เดอร์ หรือการกำหนดปริมาณในการซื้อขายเป็นต้น
3. สภาพแวดล้อมยังไม่รองรับการซื้อขายคู่สกุลเงินมากกว่า 1 คู่ในเวลาเดียวกันได้

5.3 ปัญหาและอุปสรรค

1. อัลกอริทึมของการเรียนรู้แบบเสริมแรงเชิงลึกนั้น มักใช้โครงข่ายประสาทเทียมมากกว่า 1 ตัวในการเรียนรู้หรือประเมินผล ดังนั้นการฝึกสอนตัวแทนในการซื้อขายคู่สกุลเงินนั้นจึงใช้เวลาในการฝึกสอน ทำให้ไม่สามารถใช้ข้อมูลระดับชั่วโมงหรือนาทีในการฝึกสอนได้หรือถ้าหากใช้ข้อมูลระดับชั่วโมงหรือนาทีในการฝึกสอนก็จะได้ปริมาณวันที่ไม่มากซึ่งอาจไม่ครอบคลุมพฤติกรรมที่เป็นไปได้ของตลาด

5.4 ข้อเสนอแนะ

1. ในการศึกษานี้ใช้ข้อมูลราคาและตัวชี้วัดเชิงเทคนิคเพียงอย่างเดียวให้กับตัวแทน ทำให้ตัวแทนไม่สามารถรับรู้ถึงอารมณ์ของตลาดได้ เพราะในหลาย ๆ ครั้งเมื่อเกิดเหตุการณ์ที่ไม่ปกติ นักลงทุนส่วนใหญ่จะพยายามทำการปิดออร์เดอร์ที่ทำการเปิดค้างไว้อยู่ ด้วยเหตุนี้จึงทำให้ราคาเกิดความผันผวนที่สูงขึ้นมากขึ้นในช่วงเวลาดังกล่าว ดังนั้นถ้าหากเราสามารถนำเทคนิคการวิเคราะห์เชิงพื้นฐานเข้ามาเป็นข้อมูลเพิ่มเติมให้กับตัวแทน อาจช่วยให้ตัวแทนทำงานได้ดียิ่งขึ้นกว่าเดิม

2. ออกแบบฟังก์ชันการให้รางวัลที่ดีกว่าการใช้ส่วนต่างของราคาเป็นเป้าหมายให้กับตัวแทน เพราะส่วนต่างของราคาที่แตกต่างกันไม่ได้วัดถึงความเสี่ยงเช่น การใช้อัตราส่วนชาร์ป (Sharpe Ratio) เป็นต้น

3. การประยุกต์ใช้อัลกอริทึมที่ต่อยอดมากกว่านี้เพื่อให้ตัวแทนตัดสินใจได้ดียิ่งขึ้น เช่นการใช้ตัวแทนมากกว่า 1 คนในการตัดสินใจ (Multi-agent) เป็นต้น

เอกสารอ้างอิง

[1] Krungsri Guru, **“ทำความรู้จักกับตลาดการเงินอย่าง Forex”.** [Online]. Available: https://www.krungsri.com/th/plearn-plearn/what-you-should-know-about-forex

[2] Investopedia, **“Forex Market: Who Trades Currencies and Why”.** [Online]. Available: https://www.investopedia.com/articles/forex/11/who-trades-forex-and-why.asp

[3] ADMIRAL MARKETS, **“พื้นฐาน : วิธีการเลือกคู่สกุลเงิน Forex ที่วิ่งแรง, นิยมเล่น”.** [Online]. Available: https://admiralmarkets.sc/th/education/articles/forex-basics/forex-currency-pairs

[4] MJV, **“Algo Trading: how algorithms are impacting the financial market”.** [Online]. Available: https://www.mjvinnovation.com/blog/algo-trading/

[5] FOREXBERISH, **“Long (Buy) / Short (Sell) คืออะไร”.** [Online]. Available: https://www.forexberich.com/home.php?topic=82.0

[6] Mnih, V. Kavukcuoglu, K. Silver, D. Graves, A. Antonoglou, I. Wierstra, D. and Riedmiller, M. 2013. **“Playing Atari with Deep Reinforcement Learning”.** DeepMind Technologies

[7] Hasselt, H.V. Guez, A. and Silver, D. 2015. **“Deep Reinforcement Learning with Double Q-learning”.** Google DeepMind

[8] Wang, Z. Schaul, T. Hessel, M. Hasselt, H.V. Lanctot, M. and Freitas, N.D. 2016. **“Dueling Network Architectures for Deep Reinforcement Learning”.** Google DeepMind

[9] Schaul, T. Quan, J. Antonoglou, I. and Silver, D. 2016. **“Prioritized Experience Replay”.** Google DeepMind

[10] Hongyang, Y. Xiao-Yang, L. Shan, Z. and Anwar, W. 2020. **“Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy”.** Columbia University, Nokia-Bell Labs

[11] Ishikawa, K. and Nakata, K. 2021. **“Online Trading Models with Deep Reinforcement Learning in the Forex Market Considering Transaction Costs”.** Department of Industrial Engineering and Economics Tokyo Institute of Technology, Tokyo, Japan

[12] Jo, U. Jo, T. Kim, W. Yoon, I. Lee, D. and Lee, S. 2019. **“Cooperative Multi-Agent Reinforcement Learning Framework for Scalping Trading”.** ModuLabs, Seoul, KR

[13] Paiva, F.C.L. Felizardo, L.K. Bianchi, R.A.D.C. and Costa, A.H.R. 2021. **“Intelligent Trading Systems: A Sentiment-Aware Reinforcement Learning Approach”.** Universidade de São Paulo, Centro Universitário FEI

[14] OpenAI, **“Faulty Reward Functions in the Wild”.** [Online]. Available: https://openai.com/blog/faulty-reward-functions/

ประวัติผู้เขียน

ชื่อ	นายธัชชัย สมุทรโสภากุล
วัน เดือน ปีเกิด	17 กันยายน 2539
ที่อยู่ปัจจุบัน	1248/3 ถ.ไชยพร ต.แม่กลอง อ.เมือง จ.สมุทรสงคราม 75000 โทร. 080-535-9355
ประวัติการศึกษา	(2561) วิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ เกรดเฉลี่ย 3.73 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี