รายงาน

รายวิชา CPSC 421 ปัญญาประดิษฐ์

จัดทำโดย

นางสาวนิชาภา กะจันทร์

รหัสนิสิต 6108111006

คณะบริหารธุรกิจและรัฐประศาสนศาสตร์

สาขาวิทยาการคอมพิวเตอร์

เสนอ

อาจารย์วิเชพ ใจบุญ

มหาวิทยาลัยเนชั่นลำปาง ภาคการศึกษาที่ 1/2563

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา CPSC 421 ปัญญาประดิษฐ์ โดยมีจุดประสงค์เพื่อศึกษาเรื่องของการใช้ AI โดยจะมีพูดถึง Minimax Algorithm โดยนำมาใช้กับการทำ AI สำหรับเกมส์ Tic-Tac-Toe เพื่อเป็นแนวทางในการศึกษา และต่อยอดความรู้ที่ได้จากการเรียนช่วยเสริมสร้างทักษะความรู้ให้กับตัวข้าพเจ้าเองและผู้ที่อ่านรายงานฉบับนี้

ข้าพเจ้าหวังเป็นอย่างยิ่งว่า รายงานฉบับนี้จะทำให้ทุกท่านที่ได้อ่านเห็นถึง การทำงานแบบ Recursive ของ Minimax Algorithm และข้าพเจ้าหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะประโยชน์แก่ตัวข้าพเจ้าและผู้ที่สนใจไม่มากก็น้อย

ข้าพเจ้าขอขอบคุณ อาจารย์วิเชพ ใจบุญ ที่มีส่วนร่วมในการให้ความช่วยเหลือ คำแนะนำ และข้อมูลที่เป็นประโยชน์อย่างยิ่งเพื่อจัดทำ PROJECT ในครั้งนี้ หากมีข้อผิดพลาดประการใดก็ขออภัยมา ณ ที่นี้ด้วย

นางสาวนิชาภา กะจันทร์

ผู้จัดทำ

สารบัญ

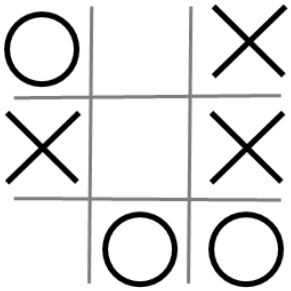
ความเป็นมา

ปัญญาประดิษฐ์ (Artificial Intelligence) หรือที่เรียกสั้น ๆ ว่า “AI” เป็นศาสตร์แขนงหนึ่งของวิทยาการคอมพิวเตอร์ที่ใช้ในการพัฒนาเครื่องจักรกล ให้สามารถเรียนรู้เพื่อแก้ปัญหา โดยหาเหตุผลจากองค์ความรู้ที่มีอยู่ แล้วนำไปวิเคราะห์จนได้ผลสรุปหรือผลลัพธ์ที่ต้องการปัญญาประดิษฐ์จึงเป็นศาสตร์ที่มีวัตถุประสงค์เพื่อใช้พัฒนาให้เครื่องจักรมีสมองและกระบวนการคิดเสมือนมนุษย์ มีประสิทธิภาพในการดำเนินงานต่าง ๆ ลดข้อผิดพลาดให้น้อยลงแก้ไขปัญหา และตัดสินใจด้วยความรวดเร็วเทียบเท่าหรือ มากกว่ามนุษย์ ระบบการกระทำที่เหมือนมนุษย์ ระบบความคิดอย่างมีเหตุผล และระบบการกระทำอย่างมีเหตุผล

การประยุกต์ใช้งานปัญญาประดิษฐ์สามารถทำได้หลายรูปแบบ ขึ้นอยู่กับจุดประสงค์ของระบบงาน ว่าต้องการให้ปัญญาประดิษฐ์ทำหน้าที่อะไร เช่น ทดแทนแรงงานมนุษย์ เพิ่มประสิทธิภาพให้ระบบงาน และลดข้อผิดพลาดในการคำนวณหรือวิเคราะห์ข้อมูล เป็นต้น ศาสตร์ที่นำปัญญาประดิษฐ์ไปประยุกต์ใช้ งานมีอยู่มากมาย เช่น ระบบผู้เชี่ยวชาญ ที่ใช้การวิเคราะห์หาวิธีการแก้ไขปัญหาต่างๆ ซึ่งมีความแน่นอน และรวดเร็วกว่ามนุษย์ สำหรับศาสตร์ทางด้านหุ่นยนต์จะนำปัญญาประดิษฐ์มาใช้ในการควบคุมเครื่องจักร และทำงานที่มีความเสี่ยงแทนแรงงานมนุษย์ เป็นต้น

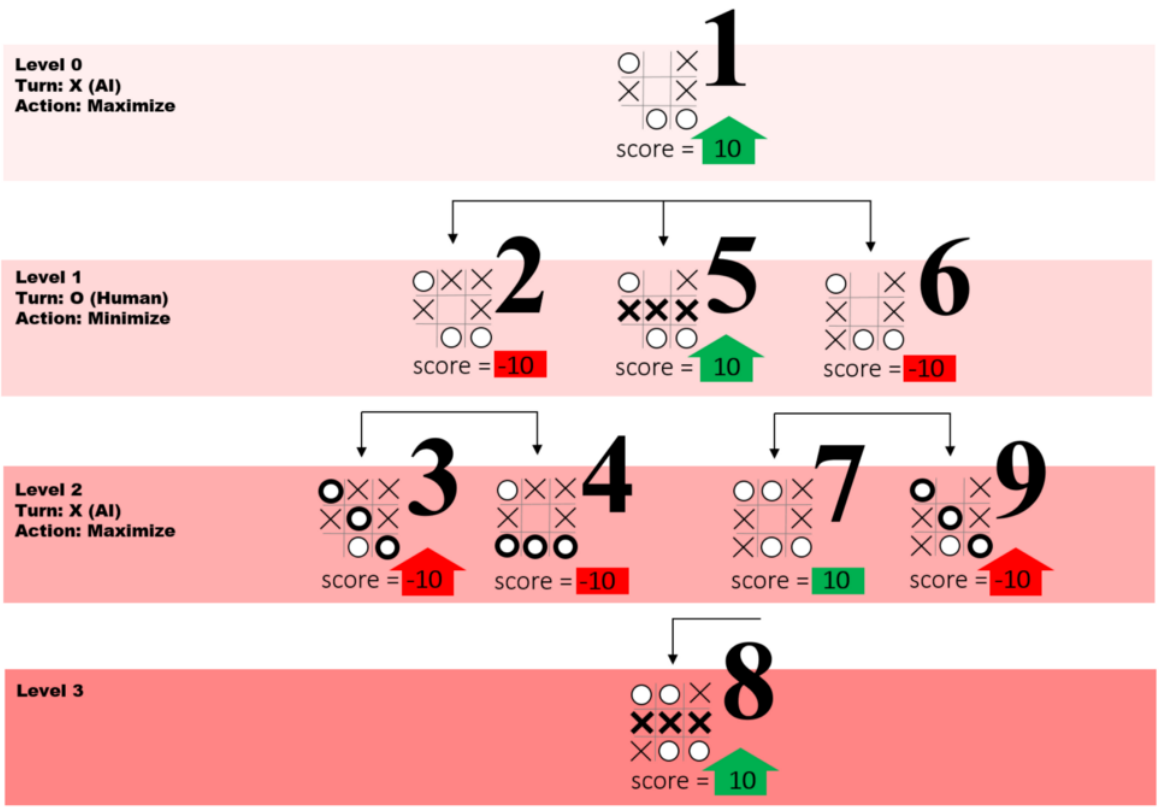
ตัวอย่างการใช้ Minimax แก้ปัญหาเกม tic-tac-toe

สมมติว่า ณ ตอนนี้บนกระดานเป็นแบบนี้ โดย AI เราคือ X, ส่วนคนเล่นคือ O



มันจะทำการคิดล่วงหน้าโดยจะมองไปให้ถึงจุดจบของเกมส์ (Terminal State) พร้อมกับคำนวณคะแนนของแต่ละแบบ โดยเราจะให้คะแนนดังนี้ละกันครับ

- AI ชนะ +10
- AI แพ้ -10
- เสมอกัน 0



1. จากรูปจะเห็นว่า บอร์ดของเราตอนนี้ (origBoard) มีช่องว่างที่ลงได้อยู่ 3 ช่อง ตัว Algorithm ก็จะทำการ สร้าง tree list เปล่า ๆ ไว้ 3 อัน แล้วก็ทำการเช็คดูว่าถึง Terminal state แล้วยัง แล้ววน Loop แบบนี้ให้ครบทุกจุด ถ้ายังไม่ถึง Terminal state ก็จะทำการใส่ X (AI เล่น) ลงไปในแต่ละช่อง แล้วเอา board หลังจากที่ใส่ X ลงไปแล้วเข้าไปคิดต่อ โดย tree level ถัดมาคนที่ใส่จะเป็น คนเล่น ที่จะใส่ O ลงไป มันก็จะใส่ O ลงไปในจุดที่ยังเหลืออยู่ แล้วก็คำนวนคะแนน ทำแบบนี้ไปเรื่อย ๆ

2. จนกระทั่งตอนจบเกมส์ (Terminal state) มันก็จะทำการ return คะแนนออกมา (คนชนะ -10, AI ชนะ +10) แล้วก็ทำการลดช่วง max, min ตาม turn ของผู้เล่น

3. สุดท้ายแล้วเมื่อไปจนครบทุก tree แล้วก็หาเส้นทางที่มี score เยอะที่สุด แล้วเลือกที่จะเล่นตามแผนที่วางไว้

ดังนั้นจากข้างต้น สุดท้ายแล้ว AI เลือกที่จะลง X ไปตรงกลาง เพื่อให้ได้ผลลัพธ์ดีที่สุด สรุปคือ Minimax มันจะพยายามทำให้ฝั่งเราได้เปรียบเยอะที่สุด (maximize) และเสียเปรียบน้อยที่สุด (minimize)

sort code

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="lib/chessboardjs/css/chessboard-0.3.0.css">
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div class="main">
  <table>
    <tr>
      <td class="cell" id="0"></td>
      <td class="cell" id="1"></td>
      <td class="cell" id="2"></td>
    </tr>
    <tr>
      <td class="cell" id="3"></td>
      <td class="cell" id="4"></td>
      <td class="cell" id="5"></td>
    </tr>
    <tr>
      <td class="cell" id="6"></td>
      <td class="cell" id="7"></td>
      <td class="cell" id="8"></td>
    </tr>
  </table>
  <p>choose</p>
  <div class="options">
    <div class="dots">
    </div>
```

```
    <div class="dots2">
    </div>
  </div>
</div>
<script src="lib/jquery/jquery-3.2.1.min.js"></script>
<script src="lib/chessboardjs/js/chess.js"></script>
<script src="lib/chessboardjs/js/chessboard-0.3.0.js"></script>
<script src="script.js"></script>
<script>


</script>
</body>
</html>
```

script.js

```
$(document).ready(function() {
  $(".dots").click(function() {
    $(".options, p").css("visibility", "hidden");
    $("td").css("visibility", "visible");
    aiCo = "#333";
    huCo = "white";
    console.log("white");
  });
  $(".dots2").click(function() {
    $(".options, p").css("visibility", "hidden");
    $("td").css("visibility", "visible");
    console.log("black");
  });

  $("td").click(function() {
    move(this, huPlayer, huCo);
    console.log("clicked");
  });
});
var board = [0, 1, 2, 3, 4, 5, 6, 7, 8];
var huPlayer = "P";
var aiPlayer = "C";
var iter = 0;
var round = 0;
var aiCo = "white";
var huCo = "#333";

function move(element, player, color) {
  console.log("element"+ element.id);
  if (board[element.id] != "P" && board[element.id] != "C") {
    round++;
```

```
$(element).css("background-color", color);
board[element.id] = player;
console.log(board);

if (winning(board, player)) {
  setTimeout(function() {
    alert("YOU WIN");
    reset();
  }, 500);
  return;
} else if (round > 8) {
  setTimeout(function() {
    alert("TIE");
    reset();
  }, 500);
  return;
} else {
  round++;
  var index = minimax(board, aiPlayer).index;
  var selector = "#" + index;
  $(selector).css("background-color", aiCo);
  board[index] = aiPlayer;
  console.log(board);
  console.log(index);
  if (winning(board, aiPlayer)) {
    setTimeout(function() {
      alert("YOU LOSE");
      reset();
    }, 500);
    return;
  } else if (round === 0) {
    setTimeout(function() {
      alert("tie");
```

```
          reset();
        }, 500);
        return;
      }
    }
  }
}

function reset() {
  round = 0;
  board = [0, 1, 2, 3, 4, 5, 6, 7, 8];
  $("td").css("background-color", "transparent");
}

function minimax(reboard, player) {
  iter++;
  let array = avail(reboard);
  if (winning(reboard, huPlayer)) {
    return {
      score: -10
    };
  } else if (winning(reboard, aiPlayer)) {
    return {
      score: 10
    };
  } else if (array.length === 0) {
    return {
      score: 0
    };
  }

  var moves = [];
  for (var i = 0; i < array.length; i++) {
```

```
  var move = {};
  move.index = reboard[array[i]];
  reboard[array[i]] = player;

  if (player == aiPlayer) {
    var g = minimax(reboard, huPlayer);
    move.score = g.score;
  } else {
    var g = minimax(reboard, aiPlayer);
    move.score = g.score;
  }
  reboard[array[i]] = move.index;
  moves.push(move);
}

var bestMove;
if (player === aiPlayer) {
  var bestScore = -10000;
  for (var i = 0; i < moves.length; i++) {
    if (moves[i].score > bestScore) {
      bestScore = moves[i].score;
      bestMove = i;
    }
  }
} else {
  var bestScore = 10000;
  for (var i = 0; i < moves.length; i++) {
    if (moves[i].score < bestScore) {
      bestScore = moves[i].score;
      bestMove = i;
    }
  }
}
```

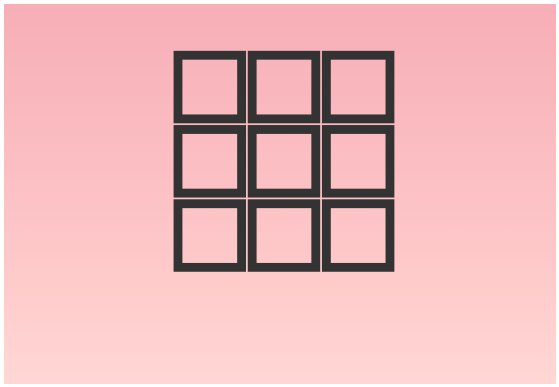
```
  return moves[bestMove];
}

//available spots
function avail(reboard) {
  return reboard.filter(s => s != "P" && s != "C");
}

// winning combinations
function winning(board, player) {
  if (
    (board[0] == player && board[1] == player && board[2] == player) ||
    (board[3] == player && board[4] == player && board[5] == player) ||
    (board[6] == player && board[7] == player && board[8] == player) ||
    (board[0] == player && board[3] == player && board[6] == player) ||
    (board[1] == player && board[4] == player && board[7] == player) ||
    (board[2] == player && board[5] == player && board[8] == player) ||
    (board[0] == player && board[4] == player && board[8] == player) ||
    (board[2] == player && board[4] == player && board[6] == player)
  ) {
    return true;
  } else {
    return false;
  }
}
```

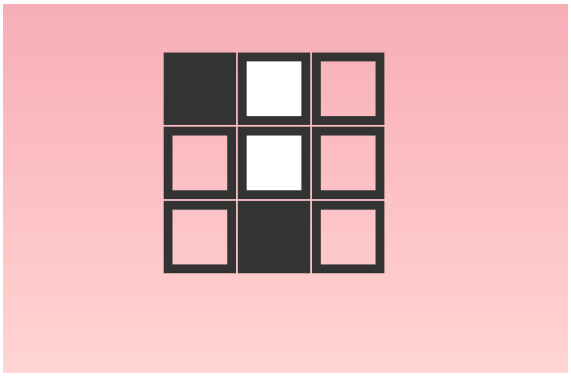
style.css

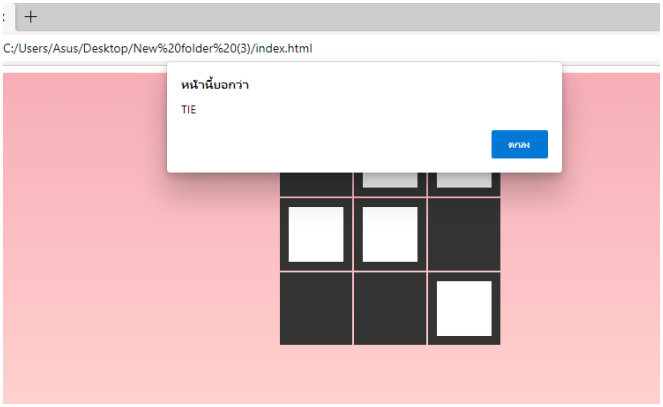
```
p {
  top:10%;
  font-size: 500%;
  background:transparent;
}
.dots{
display:inline-block;
  background:white;
  height:80px;
  width:80px;
  float: left;
  border: 10px solid #333;
}
.dots2{
  display:inline-block;
  background:#333;
  height:80px;
  width:80px;
  float: right;
}

.options{
  height:100px;
  width: 220px;
  z-index:2;
}

td {
border: 10px solid #333;
height: 60px;
width: 60px;
```

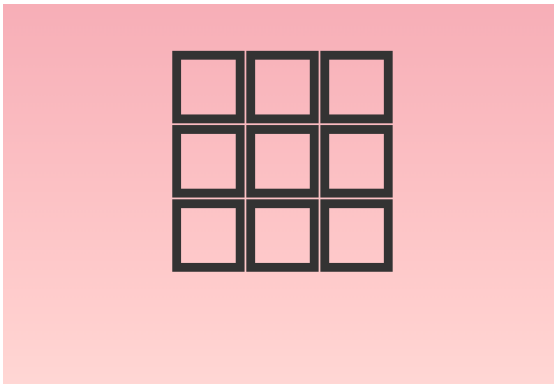
```
visibility: hidden;
}

table{
  position:absolute;
  left:50%;
  margin-left:-90px;
  top:10%;
  z-index:1;
}
.main{
  width:100vw;
  height:100vh;
  background:linear-gradient(#f7aeb7, #ffcccc, #ffeee6);
  display:flex;
  flex-direction:column;
  align-items:center;
}
```

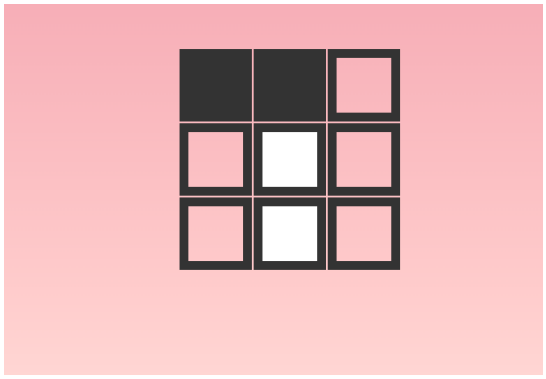
ผลลัพธ์

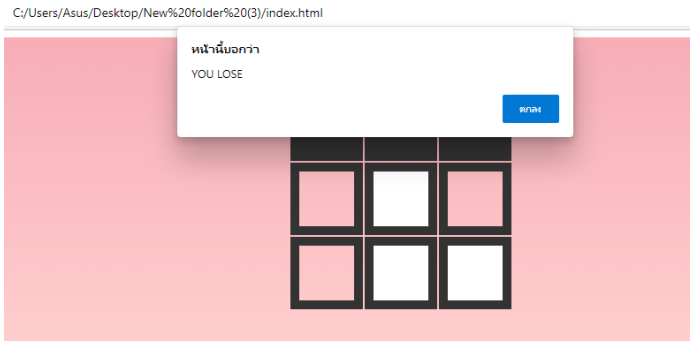





C:/Users/Asus/Desktop/New%20folder%20(3)/index.html
หน้านี้บอกว่า
TIE
ตกลง






C:/Users/Asus/Desktop/New%20folder%20(3)/index.html
หน้านี้บอกว่า
YOU LOSE
ตกลง