Solving Angry Birds with Reinforcement Learning

Richard Kwasnicki & Julius Gonsior
February 23, 2017

Final presentation for INF-PM-FPA Profilmodul Forschungsprojekt Anwendung
Angry Birds - The Game
Angry Birds - The Game

- Artillery game
- Release date: December 2009 (iPhone)
- With > 2 billion downloads in total most popular mainstream game
- Revenue of Rovio Entertainment in 2015: 142 million euro
Principle

- Primary aim: elimination of all pigs
- Secondary aim: maximize points (3 stars)
- Destroyed/damaged objects (ice, wood, stone, pig) and remaining birds = score
Which types of birds do exist?

In the first episode “Poached Eggs’ exist:

<table>
<thead>
<tr>
<th>Picture</th>
<th>Type</th>
<th>Strength</th>
<th>On-Click effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎯</td>
<td>Red (Red)</td>
<td>Nothing</td>
<td>Nothing</td>
</tr>
<tr>
<td>🐦</td>
<td>Blue (Jim, Jake &amp; Jay)</td>
<td>Ice</td>
<td>Triplication</td>
</tr>
<tr>
<td>🍊</td>
<td>Yellow (Chuck)</td>
<td>Wood</td>
<td>Speed-up</td>
</tr>
<tr>
<td>🐦</td>
<td>Black (Bomb)</td>
<td>Stone</td>
<td>Explosion</td>
</tr>
<tr>
<td>🕳️</td>
<td>White (Matilda)</td>
<td>Explosive egg</td>
<td>Drops egg</td>
</tr>
</tbody>
</table>

More types in later episodes ...
Why does it matter for AI?

• Predict outcome of physical actions
• No complete knowledge of the world
• Select best action out of \((640 \times 480)^{birds \times taps}\)
• 3Birds → 1.68 × 10^{1646}
• Planning over multiple shots

→ Competition was born
The competition

- Yearly competition during IJCAI
- Main goal: AI better than Human
- Unknown, newly created levels
- 4 Rounds (Elimination, highest points)
- See more on http://aibirds.org/
Existing Solutions/Approaches
Existing Solutions/Approaches

- DataLab Birds (Data Science Lab Prague, Czech Republic, 1018230 Points)
  - Heuristics for different good actions
- AngryBER (Data Science, Ioannina, Greece, 935330 Points)
  - Machine Learning
  - Calculate expected reward
- Plan A+ (Computer Engineering, Sejong, Korea, 1002380 Points)
  - Two strategy’s depending on object breakability
  - Manually tuned parameters (heuristic)
- AngryHEX (974670 Points)
  - ASP Knowledge Base (heuristic)
  - Scene encoding into logic assertions
Our Approach
What do we want to do different?

- Existing solutions (mainly): manual identification of rules or creation of heuristics/scores for good shots
- Our aim: **automatic identification** of a heuristic which means:
  - Experimental shooting
  - Learn what makes a good combination (planning)
  - Application of learned knowledge in unknown levels

→ Reinforcement Learning
Reinforcement Learning
Origin

- Field of machine learning
- Behaviorist psychology: animal learning
- Math: optimal control
What is the main principle of Reinforcement Learning?

[Diagram showing the components of Reinforcement Learning: Agent State ($S^a_t$), Observation ($O_t$), Reward ($R_t$), Action ($A_t$), Environmental State ($S^e_t$).]
Q-Learning (1)

Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Old Value

Parameter:

- \( s \) ... state
- \( a \) ... action

<table>
<thead>
<tr>
<th>( s )</th>
<th>12 31</th>
<th>14 52</th>
<th>15 72</th>
<th>15 65</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0</td>
<td>-1</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Q-Learning (1)

Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Learning rate

Parameter:

- s ... state
- a ... action

<table>
<thead>
<tr>
<th></th>
<th>12 31</th>
<th>14 52</th>
<th>15 72</th>
<th>15 65</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>0</td>
<td>-1</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>s1</td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
</tr>
<tr>
<td>s2</td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td>s3</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>s4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Q-Learning (1)

Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Reward

Parameter:
- \( s \) ... state
- \( a \) ... action

<table>
<thead>
<tr>
<th>State</th>
<th>12 31</th>
<th>14 52</th>
<th>15 72</th>
<th>15 65</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0</td>
<td>-1</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Q-Learning (1)

Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Discount factor

Parameter:

- \( s \) ... state
- \( a \) ... action

<table>
<thead>
<tr>
<th></th>
<th>12 31</th>
<th>14 52</th>
<th>15 72</th>
<th>15 65</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0</td>
<td>-1</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Estimate of optimal future value

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>31</th>
<th>14</th>
<th>52</th>
<th>15</th>
<th>72</th>
<th>15</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td>27</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>( s_1 )</td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_2 )</td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_3 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s_4 )</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameter:

- \( s \) ... state
- \( a \) ... action
Q-Learning (1)

Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Parameter:

- s ... state
- a ... action

<table>
<thead>
<tr>
<th>( s_0 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( s_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>17</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>27</td>
<td>12</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-10</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>15 65</td>
<td>15 72</td>
<td>15 65</td>
<td>15 65</td>
<td>15 65</td>
</tr>
</tbody>
</table>

Solving Angry Birds with Reinforcement Learning Richard Kwasnicki & Julius Gonsior 10
Q-Learning (1)

Bellmann → Value Iteration:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right)
\]

Parameter:

- s ... state
- a ... action

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| s_0 | 0  | -1 | 27 | 3  |
| s_1 | 5  | -1 | 12 | -1 |
| s_2 | -1 | 17 | 2  | -10|
| s_3 | -1 | -1 | -1 | -1 |
| s_4 | -1 | -1 | -1 | -1 |
Bellmann → Value Iteration:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \times \left( r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Parameter:

- **s** ... state
- **a** ... action

<table>
<thead>
<tr>
<th></th>
<th>12 31</th>
<th>14 52</th>
<th>15 72</th>
<th>15 65</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>s_0</strong></td>
<td>0</td>
<td>15</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td><strong>s_1</strong></td>
<td>5</td>
<td>-1</td>
<td>12</td>
<td>-1</td>
</tr>
<tr>
<td><strong>s_2</strong></td>
<td>-1</td>
<td>17</td>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td><strong>s_3</strong></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td><strong>s_4</strong></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Law of Effect aka Trial & Error with parameter $\varepsilon$ (exploration rate)

```java
public Action getNextAction() {
    int randomValue = randomGenerator.nextInt(100);
    Action action;
    if (randomValue < explorationRate * 100) {
        action = qValuesDAO.getRandomAction();
    } else {
        action = qValuesDAO.getBestAction();
    }
    return action;
}
```
Q-Learning (3) - Delayed feedback

- Positive reward only on last shot
- First action affected after $n$ games for $n$ birds
Labyrinth Example

- 5 x 5 World
- Possible Actions: Up, Right, Down, Left
- Rewards per move:
  - Green tile: +1
  - Red tile: -1
  - Default tile: -0.05

Live
Our Model for Angry Birds

- State: Serialized screenshot with
  - rounded coordinates
  - object-type
  - object-shape
- Action: Shoot on center of an object (Limitation: object direct reachable)
- Reward: Score after successful finishing the level
State: RedBird 19 32 Rect ... Wood 59 35 Rect Pig 54 29 Rect
Action: Wood 49 35 Rect Wood 54 31 Rect ...
Observation of pure Reinforcement Learning Approach
Observation of pure Reinforcement Learning Approach

- Played levels after 1 week: 3.217 on a VM from the ZIH with 16 cores and 32GB ram
Observation of pure Reinforcement Learning Approach

- Played levels after 1 week: 3.217 on a VM from the ZIH with 16 cores and 32GB ram → much too less for proper Reinforcement Learning → delayed feedback!
Observation of pure Reinforcement Learning Approach

- Played levels after 1 week: 3.217 on a VM from the ZIH with 16 cores and 32GB ram → much too less for proper Reinforcement Learning → delayed feedback!
- Achieved scores similar to naïve agent → suggests that nothing was learned
Encountered practical Problems while implementing the Theory
Game is implemented as closed source Chrome plugin

- Chrome plugin and screenshot analysis needs lots of resources
  - couldn’t run more than 4 Chrome processes in parallel on a ZIH VM with 16 cores and 32GB ram
Encountered Problems

Game is implemented as closed source Chrome plugin

- Chrome plugin and screenshot analysis needs lots of resources
  → couldn’t run more than 4 Chrome processes in parallel on a ZIH VM with 16 cores and 32GB ram
- Game is slow: one shot takes ~ 20 seconds
  → 1,000,000 shots will take ~ 232 days
Game is implemented as closed source Chrome plugin

- Chrome plugin and screenshot analysis needs lots of resources
  → couldn’t run more than 4 Chrome processes in parallel on a ZIH VM with 16 cores and 32GB ram
- Game is slow: one shot takes ~ 20 seconds
  → 1,000,000 shots will take ~ 232 days
- Not possible to create new levels → overfitting!
Encountered Problems

Given Vision Module is not working exactly

• Object coordinates differ in each run
Encountered Problems

Given Vision Module is not working exactly

- Object coordinates differ in each run
- Wrong object recognition results in:
Encountered Problems

Given Vision Module is not working exactly

- Object coordinates differ in each run
- Wrong object recognition results in:
  - same action can lead to multiple states
Encountered Problems

Given Vision Module is not working exactly

- Object coordinates differ in each run
- Wrong object recognition results in:
  - same action can lead to multiple states
  - same target object didn’t result always in same states
Encountered Problems

Given Vision Module is not working exactly

- Object coordinates differ in each run
- Wrong object recognition results in:
  - same action can lead to multiple states
  - same target object didn’t result always in same states

$\rightarrow$ increases exponentially search space
Possible Solutions:

• Reimplement game without GUI and actual API
  → takes a lot of time
Encountered Problems

Possible Solutions:

• Reimplement game without GUI and actual API
  → takes a lot of time

• Try to optimize/speed up existing game implementation
  → will never be as performant as reimplementation
Encountered Problems

Possible Solutions:

- Reimplement game without GUI and actual API
  → takes a lot of time
- Try to optimize/speed up existing game implementation
  → will never be as performant as reimplementation
- Limit search space
  → need to manually discard possible solutions
Encountered Problems

Possible Solutions:

- Reimplement game without GUI and actual API
  → takes a lot of time
- Try to optimize/speed up existing game implementation
  → will never be as performant as reimplementation
- Limit search space
  → need to manually discard possible solutions
- Use much more computing power
Encountered Problems

Possible Solutions:

• Reimplement game without GUI and actual API
  → takes a lot of time

• Try to optimize/speed up existing game implementation
  → will never be as performant as reimplementation

• Limit search space
  → need to manually discard possible solutions

• Use much more computing power
New approach
• Difference to previous solution: drastically reduced search space by limiting possible target actions
Hybrid approach: Reinforcement Learning and Heuristics

• Difference to previous solution: drastically reduced search space by limiting possible target actions
• Instead of proposing all possible target objects, low/high trajectories and tap time a preselection of most promising targets

→ no totally new strategies

Solving Angry Birds with Reinforcement Learning Richard Kwasnicki & Julius Gonsior 20
Hybrid approach: Reinforcement Learning and Heuristics

- Difference to previous solution: drastically reduced search space by limiting possible target actions
- Instead of proposing all possible target objects, low/high trajectories and tap time a preselection of most promising targets
- Reinforcement learning shall again select, based on the current state, which preselected action to take
Hybrid approach: Reinforcement Learning and Heuristics

• Difference to previous solution: drastically reduced search space by limiting possible target actions
• Instead of proposing all possible target objects, low/high trajectories and tap time a preselection of most promising targets
• Reinforcement learning shall again select, based on the current state, which preselected action to take
• Drawback: learned strategy is limited by preselected actions
→ no totally new strategies
Heuristics used for preselection of possible targets:

- Big round objects
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🎈
- TNT 💣

Score depending on following factors:
- Number of objects above
- Number of objects in trajectory
- Number of objects to the right
- Number of objects below
- How many shots are left
- Material
- Orientation
- Distance to pigs
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

• Big round objects 🎈
• TNT 💣
• Multiple pig shot

Score depending on following factors:

• Number of objects above
• Number of objects in trajectory
• Number of objects to the right
• Number of objects below
• How many shots are left
• Material
• Orientation
• Distance to pigs
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

• Big round objects 🎮
• TNT 🔥
• Multiple pig shot

• Score depending on following factors:
  • Number of objects above
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🎱
- TNT 📦
- Multiple pig shot

Score depending on following factors:
- Number of objects above
- Number of objects in trajectory
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

• Big round objects
• TNT
• Multiple pig shot

Score depending on following factors:

• Number of objects above
• Number of objects in trajectory
• Number of objects to the right
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🎈
- TNT 🧵
- Multiple pig shot

- Score depending on following factors:
  - Number of objects above
  - Number of objects in trajectory
  - Number of objects to the right
  - Number of objects below
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🥤
- TNT 🧵
- Multiple pig shot

Score depending on following factors:
- Number of objects above
- Number of objects in trajectory
- Number of objects to the right
- Number of objects below
- How many shots are left 🎯
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🏸
- TNT 🔴
- Multiple pig shot

Score depending on following factors:

- Number of objects above
- Number of objects in trajectory
- Number of objects to the right
- Number of objects below
- How many shots are left 🎯
- Material 🏱
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🎉
- TNT 🔥
- Multiple pig shot

Score depending on following factors:
- Number of objects above
- Number of objects in trajectory
- Number of objects to the right
- Number of objects below
- How many shots are left 💦
- Material 🔶
- Orientation
Hybrid approach: Reinforcement Learning and Heuristics

Heuristics used for preselection of possible targets:

- Big round objects 🗸
- TNT 🧵
- Multiple pig shot

Score depending on following factors:
- Number of objects above
- Number of objects in trajectory
- Number of objects to the right
- Number of objects below
- How many shots are left 🎯
- Material 🔴
- Orientation
- Distance to pigs 🐰
Observation again
Observation of Reinforcement Learning and Heuristic

- Played levels after 2 weeks: 5172
Observation of Reinforcement Learning and Heuristic

- Played levels after 2 weeks: 5172
- 1–2 tries needed to solve first level, later level needed comparably much more tries
• Max values suggest that reinforcement learning is a suitable promising method for solving of Angry Birds
• Max values suggest that reinforcement learning is a suitable promising method for solving of Angry Birds
• But: good results aren’t reproducible yet because the really good shots haven’t been remembered → delayed feedback in Q-learning
Ready for competition yet?

- Max values suggest that reinforcement learning is a suitable promising method for solving of Angry Birds
- **But:** good results aren’t reproducible yet because the really good shots haven’t been remembered → delayed feedback in Q-learning
- for competition the overfitting still needs to be checked
Ready for competition yet?

- Max values suggest that reinforcement learning is a suitable promising method for solving of Angry Birds
- **But**: good results aren’t reproducible yet because the really good shots haven’t been remembered → delayed feedback in Q-learning
- for competition the overfitting still needs to be checked
- not yet implemented strategy for solving unknown level in competition modus
Outlook
What would we do different if we were starting all over?

- Reimplement the game in a more machine learning friendly way
What would we do different if we were starting all over?

- Reimplement the game in a more machine learning friendly way
  - API to get current objects
- Can't discourage to not use Reinforcement Learning, result of second approach looked promising → leads to speculation that first approach probably would have worked with more iterations
What would we do different if we were starting all over?

• Reimplement the game in a more machine learning friendly way
  • API to get current objects
  • Direct output of computed resulted state after shooting an object (no useless wait for pretty animation to finish)
What would we do different if we were starting all over?

- Reimplement the game in a more machine learning friendly way
  - API to get current objects
  - Direct output of computed resulted state after shooting an object (no useless wait for pretty animation to finish)
- Can’t discourage to not use Reinforcement Learning, result of second approach looked promising → leads to speculation that first approach probably would have worked with more iterations
Questions?
Level 14
Level 14

LEVEL CLEARED!

29140

LOGIN
Level 14

Level Cleared!

New Highscore!

86470
Level 21
Level 21
Level 21
Level 21
Level 21
Level 21

LEVEL CLEARED!

94770

NEW HIGHSCORE!

LOGIN →
Level 21

LEVEL CLEARED!

53768

LOGIN
Level 21
Level Cleared!

New Highscore!

94770