Learning Artificial Intelligence in Large-Scale Video Games

— A First Case Study with Hearthstone: Heroes of WarCraft —

Master Thesis Submitted for the Degree of
MSc in Computer Science & Engineering

David Taralla
Author

Prof. Damien Ernst
Supervisor

Academic year 2014 – 2015

Video Games, Then and Now

- Then, the problems to solve were representable easily

  → Example: *Pac-Man*

  - Fully observable maze
  - Limited number of agents
  - Small, well-defined action space

- Now, the problems feature numerous variables

  → Example: *StarCraft*

  - Vast, partially observable map
  - Complex state representation
  - Prohibitively large action space, difficult to represent

Video Games, Then and Now

Games continue to feature richer environments...

Video Games, Then and Now

Games continue to feature richer environments...

... but designing robust AIs becomes increasingly difficult!

Video Games, Then and Now

Games continue to feature richer environments...

... but designing robust AIs becomes increasingly difficult!

$\Downarrow$

Making AI learn instead of being taught: a better solution?

Objectives of this Thesis

1. Design & study of a theory for creating autonomous agents in the case of large-scale video games

   → Study applied to the game *Hearthstone: Heroes of Warcraft*

2. Develop a modular and extensible clone of the game *Hearthstone: HoW*

   → Makes us able to test the theory practically

Problem Statement

1. State Vectors

- World vector $w \in \mathcal{W}$ contains all information available in a given state

  $\rightarrow$ Everything is not relevant

- If $\sigma(\cdot)$ is the projection operator such that

$$\forall w \in \mathcal{W}, s = \sigma(w)$$

is the relevant part of w for the targeted application, we define

$$\mathcal{S} := \{\sigma(w) \,|\, w \in \mathcal{W}\}$$

the set of all state vectors.

Problem Statement

2. Action Vectors

- Available actions have unknown consequences
- Let $\mathcal{A}$ be the set of available actions in the game
- Let $\mathcal{A}_s$ be the set of actions that can be taken in state $s \in \mathcal{S}$

Problem Statement

3. State Scoring Function

- There should exist a bounded function

$$\rho : \mathcal{S} \to \mathbb{R}$$

having the following properties:

$$\begin{cases} \rho(s) < 0 & \text{if, from } s \text{ info, the player is considered as likely to lose,} \\ \rho(s) > 0 & \text{if, from } s \text{ info, the player is considered as likely to win,} \\ \rho(s) = 0 & \text{otherwise.} \end{cases}$$

- Based on expert knowledge

Problem Statement

4. Problem Formalization

- Games follow discrete-time dynamics:

$$\tau : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \mid (s_t, a) \mapsto s_{t+1} \text{ for } a \in \mathcal{A}_{s_t}, \quad t = 0, 1, \ldots$$

- Let R_ρ be an objective function whose analytical expression depends on ρ:

$$R_\rho : S \times \mathcal{A} \to \mathbb{R} \mid (s, a) \mapsto R_\rho(s, a) \text{ for } a \in \mathcal{A}_s.$$

Problem Statement

4. Problem Formalization

- $R_\rho(s, a)$ is considered uncomputable from state s

  $\rightarrow$ Difficulty to simulate side-trajectories in large-scale games

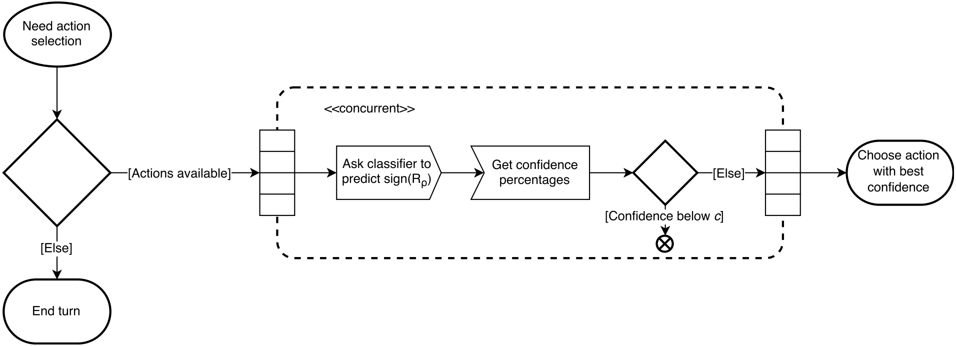
- Find an action selection policy h such that

$$h : S \rightarrow \mathcal{A} \mid s \mapsto \underset{a \in \mathcal{A}_s}{\operatorname{argmax}} \; R_\rho(s, a).$$

Getting Intuition on Actions from State Scoring Differences

- Our analytical expression for R_ρ:

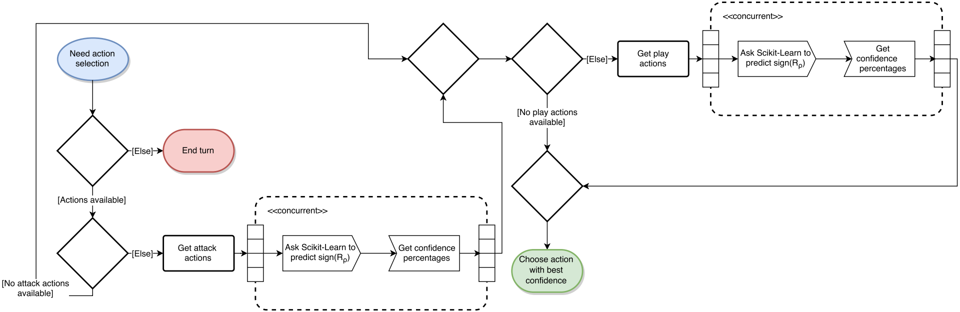
$$R_\rho(s, a) := \rho(\tau(s, a)) - \rho(s).$$



Report erratum – In Figure 3.2, the classifier is asked to predict the sign of R_ρ, and not ρ.

Nora: Design & Results

Action Selection Process



Report erratum – In Figure 4.5, the classifiers are asked to predict the sign of R_ρ, and not ρ.

Caveats

- Memory usage

  → Approx. 14GB is needed to keep the models in RAM

  → Fix: tree pruning and parameters tuning

- Play actions classifier underestimates the value of some actions

  → Random target selection is assumed after playing an action that needs a target

  → Fix: Two-step training

Results

Matchup	**Win rate**
Nora vs. Random	93%
Nora vs. Scripted	10%...

But compared to the random player performance...

Results

Matchup	Win rate
Nora vs. Random	93%
Nora vs. Scripted	10%
Random vs. Scripted	$<$ 1% !

- Nora applies some strategy the random player does not
- Qualitatively, this translates into a board control behavior

  $\rightarrow$ Never target her allies with harmful actions, even though it is allowed

  $\rightarrow$ Accurate understanding of the Fireblast special power

Conclusion

Any questions?

Thank you for your attention.

Appendix – Why Extremely Randomized Trees?

- Ensemble methods can often surpass single classifiers

  $\rightarrow$ From a statistical, computational and representational point of view

- Decision trees are particularly suited for ensemble methods

  $\rightarrow$ Low computational cost of the standard tree growing algorithm

  $\rightarrow$ But careful about memory...

- Random trees suited for problems with many features

  $\rightarrow$ Each node can be built with a random subset of features

- Feature importances

  $\rightarrow$ Useful for designing the projection operator $\sigma : \mathcal{W} \rightarrow \mathcal{S}$

Appendix – Computation of the ExtraTrees Classifier Confidence

- It is the predicted positive class probability of the classifier
- Computed as the mean predicted positive class probability of the trees in the forest
- Predicted positive class probability of a sample s in a tree:

$$\frac{\#\{s' \in \text{leaf in which } s \text{ falls} \mid s' \text{ labelled positive}\}}{\#\{s' \in \text{leaf in which s falls}\}}$$

Appendix – Basics of Hearthstone: Heroes of WarCraft

- Stylized combat game
- Cards are obtained by drawing from your deck
  → Your hand is hidden to your opponent

Goal: Make the enemy player's hero health go to zero.

Appendix – Basics of Hearthstone: Heroes of WarCraft

- Cards are played using a resource: the Mana

  → Minions that join the battle

  → Spells

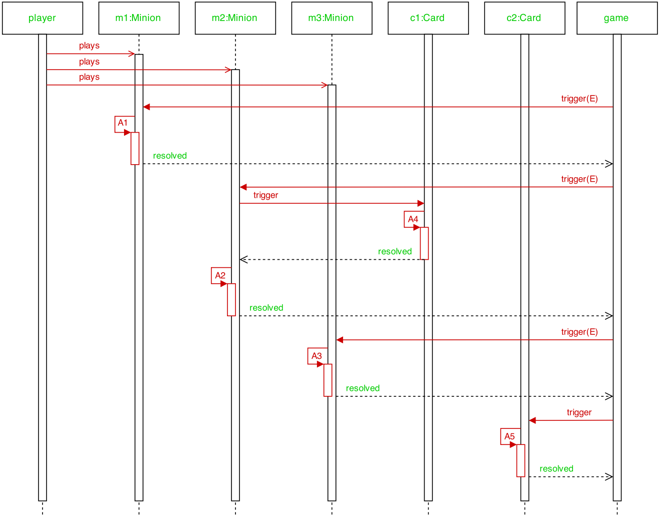
- Rules are objects in the game

  → Game based on creating new and breaking/modifying rules

Appendix – Basics of Hearthstone: Heroes of WarCraft

Appendix – Basics of Hearthstone: Heroes of WarCraft

Things Might Get Tricky...!



Appendix – The simulator

- *Hearthstone: HoW* simulator created with C++/Qt 5
  - → Modular, extensible
  - → Cards are loaded from an external file
  - → Quite a challenge!
- Definition of JARS for describing cards in a user-friendly way
  - → Just Another Representation Syntax
  - → Context-aware, JSON-based language
  - → Makes it easy to create and edit cards without coding