Fitted Q Iteration

Firas Safadi
University of Liège
June 2011

Part I

Theory

Environment

* State described by

$$\boldsymbol{s} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}) \in \mathbb{R}^{\boldsymbol{n}}$$

* Action defined by

$$\boldsymbol{a} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_m}) \in \mathbb{R}^{\boldsymbol{m}}$$

* Environment responds with reward and new state

$$\begin{cases} \boldsymbol{r} \in \mathbb{R} \\ \boldsymbol{s'} \in \mathbb{R}^{\boldsymbol{n}} \end{cases}$$

Reward function

* Maps state-action pairs to rewards

$$\boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a}) \rightarrow \boldsymbol{r}$$

* We need $\boldsymbol{R}$ to take good actions!

Maximizing reward over a horizon

* Assume a horizon

$$\boldsymbol{T} \in \mathbb{N}$$

* Maximum cumulated reward

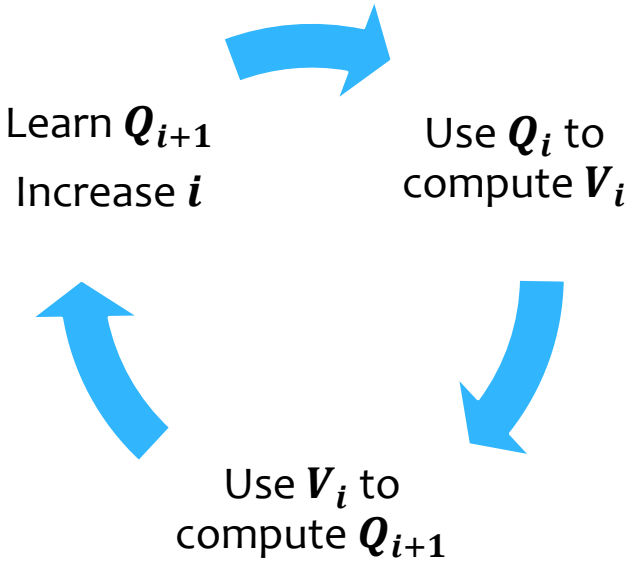
$$\boldsymbol{V_T(s)} = \max_{\boldsymbol{a},\boldsymbol{a}',\ldots} \underbrace{(\boldsymbol{R(s,a)} + \boldsymbol{R(s',a')} + \cdots)}_{\boldsymbol{T}}$$

* Maximum cumulated reward for state-action pairs

$$\boldsymbol{Q_{T+1}(s,a)} = \boldsymbol{r} + \boldsymbol{V_T(s')}$$

Fitted Q idea

* Start with a set of samples $\mathrm{U}_j(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')_j$
* Incrementally build $\boldsymbol{Q_T}$ using supervised learning



Fitted Q algorithm

* **Input**
  - $\boldsymbol{s}$ *set of states*
  - $\boldsymbol{a}$ *set of actions*
  - $\boldsymbol{r}$ *set of rewards*
  - $\boldsymbol{s'}$ *set of next states*
  - $\boldsymbol{K}$ *number of samples*
  - $\boldsymbol{T}$ *horizon*

* **Output**
  - $\boldsymbol{Q_T}$

$$\boldsymbol{Q(s,a) \leftarrow r}$$
$$for\ \boldsymbol{i = 1}\ to\ \boldsymbol{T - 1}$$
$$\quad for\ \boldsymbol{j = 0}\ to\ \boldsymbol{K - 1}$$
$$\quad\quad \boldsymbol{R_j = r_j + \max_{a'} Q(s'_j, a')}$$
$$\quad end$$
$$\quad \boldsymbol{Q(s,a) \leftarrow R}$$
$$end$$

Part II

Experiment

Problem

- **World:** 10×10 surface
- **State:** (x, y, a, b) in $[0,10]^4$
- **Action:** (u, v) in $[-1,1]^2$
- **Goal:** reach target (dist. ≤ 1)
- **Reward:** 1 if dist. ≤ 1, else 0

- Random initial position
- Random initial target position
- Only 10 moves available

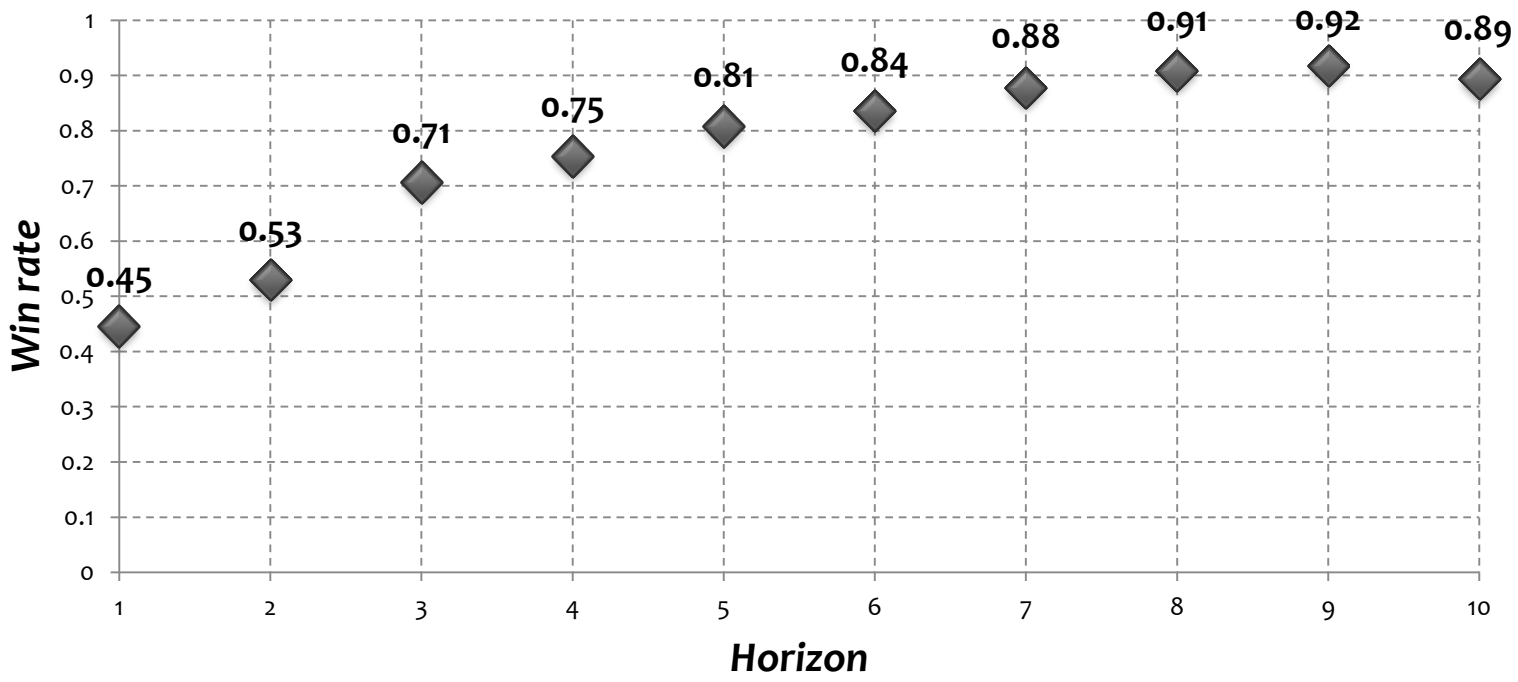
Learning

* Feed-forward backpropagation neural network
* Approx. 16,000 samples
* Simulate 100 random actions to estimate optimum

Assessing the impact of the learning horizon on performance

* Learn for horizon = 1, 2, …, 10 and play 1,000 games
* Repeat 10 times
* Total of 10,000 games per horizon

Results


Win rate
Horizon
0.45
0.53
0.71
0.75
0.81
0.84
0.88
0.91
0.92
0.89
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
2
3
4
5
6
7
8
9
10

Part III

Wrap-up

Advantages of fitted Q iteration

* Offline
* Model-free
* Works with random trajectories

Future work

* Try random forests and compare with neural networks performance
* Try different sampling methods
  * Generate samples around edges
  * Generate complete trajectories
  * Resampling
* Try on bigger problems with larger state space (i.e., MASH)

Part IV

Acknowledgments

References

* **Charles Desjardins**
  * Neural Fitted Q-Iteration (Martin Riedmiller, ECML 2005), 2007
* **Damien Ernst**
  * Computing near-optimal policies from trajectories by solving a sequence of standard supervised learning problems, 2006
* **Yin Shih**
  * Neuralyst User Guide, 2001

Thanks

* **Jean-Baptiste Hoock**
  * Implementation (MASH)
* **Nataliya Sokolovska**
  * Testing
* **Olivier Teytaud**
  * Concepts (fitted Q iteration, benchmark)

The End

Thanks for listening!