

1 Introduction

1.1 Learning from chosen actions

1.2 Recognition of real images patterns

2 Litterature review

3 Proposed algorithm

3.1 Description

3.2 Remarks

3.2.1 Regressor function

3.2.2 Signal function and discounted values

4 Asymptotic behaviour of the algorithm

4.1 Convergent estimates for equivalent underlying states

4.2 Long term gain maximization

5 Implementation for the tic-tac-toe game

5.1 Particularities of the implementation

5.2 Necessary adjustments

5.3 Performance analysis

6 Conclusion

7 References

Algorithm 1

Initialized variables:

X : empty m -columns feature matrix
 y : empty target vector
 $w^* = w_0$: random weights for the regressor
 $\gamma \in [0, 1]$ discount rate
 $\Delta = 0$: number of actions

Other variables:

$f()$: regressor function
 m : number of features
 $p_t(x^*)$: stochastic process, function of the last existing chosen state
 X' : m -columns simulated states matrix
 $x_t(p_t) \in \mathbb{R}^m$: current state vector
 $\Omega_t(x_t)$: set of possible actions
 ϵ : randomness rate
 $s(x^*) \in \mathbb{R}$: signal value potentially triggered by the modified state

WHILE the learning process is on:

record x_t

$X' := \emptyset$: empty m -columns matrix

FOR x' in $\Omega_t(x_t)$:

append x' to X'

With probability $1 - \epsilon$

$x^* := \arg \max_{x' \in X'} f(w^*, X')$

With probability ϵ

$x^* := \text{random sample } x' \in X'$

append x^* to X

IF $s(x^*) \neq 0$:

FOR δ in $1:\Delta$:

$value = d(\gamma, \Delta, \delta, s(x^*))$

append $value$ to y

$w^* := \arg \min_w (l(f(w, X), y))$

$\Delta := 0$

ELSE:

$\Delta += 1$
