Reinforcement Learning

(chapter 10)

Kai Goebel, Bill Cheetham

GE Corporate Research & Development

goebel@cs.rpi.edu

cheetham@cs.rpi.edu

1

What we will talk about

Temporal Differences

Dynamic Programming

NN Implementation

2

Introduction

Supervisor gives feedback in form of score

desired outcome is NOT fed back

Feedback is both positive and negative

Tendency to reproduce the result leading to positive feedback is strengthened (and vice versa)

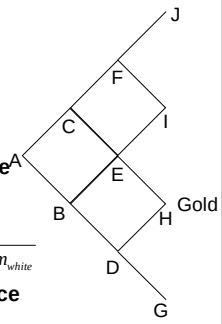
3

Reinforcement Learning: Policy

- Find path to gold pot
- Record path (different colored stones)
- If successful, add stone of the same color (reward)
- If not successful, remove stone
- Repeat many times

$$P_{down} = \frac{Num_{black}}{Num_{black} + Num_{white}}$$

- Works well for small state space



4

Temporal Difference (TD) Learning

Credit is assigned based on the difference between (temporally) successive predictions

The (discounting) recency parameter λ determines the type of learning

(ordinary)		classical
Supervised	$TD(1) \Leftarrow TD(\lambda) \Rightarrow TD(0)$	Dynamic
Learning		Programming

λ =1: cost in future as important as now, propagate all the way through path

λ =0: just consider next state

Adjustments k steps in the past are exponentially weighted

5

TD(λ)

Update rule

$$\Delta w_t = \alpha\left(V_{t+1} - V_t\right)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w V_k$$

where

V_t is the prediction value at time t

λ is the recency parameter (between 0 and 1)

k determines how many steps in the past to look

Advantage over supervised learning:

Complete data not available

If an unlikely "bad" event happens, change only weights between two states

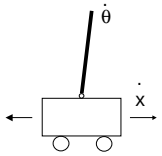
6

Pole Balancing Using Reinforcement

failure $\theta > \theta_{max}$

$x < x_{min}$

$x > x_{max}$



reward:-1 for failure
0 otherwise

decision: move to left or right

4 state variables:

cart position, velocity, pole position, velocity
discretized into 3x3x3x6=162 states

7

Dynamic Programming vs Reinforcement Learning

Motivation:

Find optimal path from one point to another by considering the best choice at each arc.

DP Formulation:

•Define optimal value function
- **Rule that assigns values to subproblems**

•Write recurrence relation
- **Set of formulae relating different nodes S**

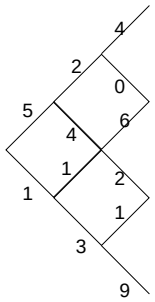
•Note boundary conditions

Absorbing state: no cost after that (goal: reach that state)

8

Simple Reinforcement Example

Consider the following path problem:
- **numbers above and under arcs denote cost between two states**
- **impose probability for the transition between two states:**
  - 3/4 to go up
  - 1/4 to go down



9

Values of the state change according to:

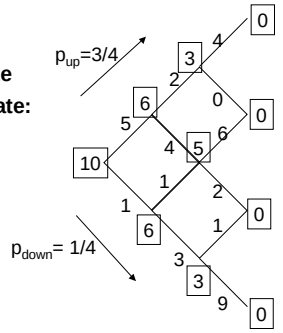
$$S(t) = p_{up}\left(cost_{up} + S(t+1)_{up}\right) + p_{down}\left(cost_{down} + S(t+1)_{down}\right)$$

where
- **S(t)=value of current state**
- **p_{up}=probability to go up**
- **$cost_{up}$=cost of the arc going up**
- **$S(t+1)_{up}$=value of the succeeding state from S(t) going up**
- **p_{down}=probability to go down**
- **$cost_{down}$=cost of the arc going down**
- **$S(t+1)_{down}$=value of the succeeding state from S(t) going down**

10

Desired Result

Example to calculate expected cost of state:
3/4(0+4)+1/4(0+0)=3
3/4(0+6)+1/4(0+2)=5
3/4(3+2)+1/4(5+4)=6



11

Reinforcement lite

The error between two states is:

$$E = 0.5\left(c_t + S(t+1) - S(t)\right)^2$$

where
- **S(t)=value of preceding state**
- **S(t+1)=value of succeeding state**
- **c_t=cost between two states**
- **The gradients of the error wrt the states consists of the terms**

$$\frac{\partial E}{\partial S(t)} \quad \text{and} \quad \frac{\partial E}{\partial S(t+1)}$$

but average of change of going into node cancels out with coming out of node (on average, for stochastic problem) => no convergence
Consider only dE/dS(t)

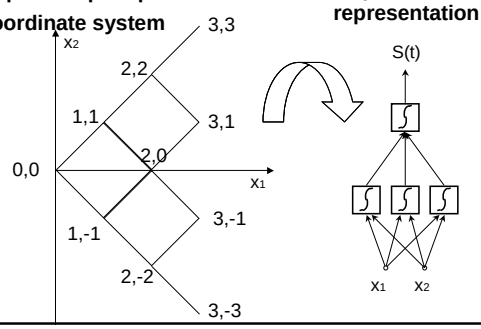
12

Soft Computing: Reinforcement Learning

NN Implementation

Represent path problem in coordinate system

Map to NN representation



13

Soft Computing: Reinforcement Learning

Procedure

- **Start with random weights**
- **Run network**
- **Get output values**
- **Compute error between 2 states**
  (look up cost between 2 states)
- **Change weights using backprop**
  (with incomplete gradient)
- **Go to next state and repeat**
  until last node (x=3) is reached
- **Continue many (!) times**
  until average of output has settled

14

Soft Computing: Reinforcement Learning

Observations

Manipulating the parameters was crucial

β needed to be < 0.01

Varying numbers of hidden nodes was used (1-10)

n>3 needed for convergence

changing η seemed to have no effects

15

Soft Computing: Reinforcement Learning

Results

beta	eta	# cycles (exp)	S(0,0)	S(1,1)	s(1,-1)	S(2,2)	s(2,0)	s(2,-2)	SSE
0.01	0.1	3	10.282	6.617	6.378	2.95	3.838	3.512	2.22
0.01	0.1	4	10.276	6.633	6.35	2.944	3.923	3.469	1.983
0.01	0.1	5	10.301	6.675	6.311	2.929	4.013	3.379	1.765
0.01	0.1	6	10.321	6.696	6.309	2.939	4.049	3.516	1.857
0.1	0.1	6	9.883	6.601	4.195	3.101	4.755	4.487	5.914
1	0.1	6	9.009	6.475	7.812	3.921	5.073	5.638	12.304
0.01	0.001	6	10.009	5.993	6.133	3.194	4.72	4.293	1.905
0.01	0.0001	6	9.991	6.356	5.533	3.627	3.619	2.739	2.739
0.01	0.1	5	10.301	6.675	6.311	2.929	4.013	3.379	1.765

16

Soft Computing: Reinforcement Learning

last slide

17