Autonomous Orchard Solutions

The real equation used for the harvest reward was:

$r_{ht} = -2c_{at} + c_{bt} + 3c_{ct} \pm 1$

1. First set up arbitrary weights for harvesting
   $\mathbf{w}_{h0} = (3 \quad 2 \quad 1)$
   We also choose a learning rate $\alpha = 0.01$
   Let the feature vector representing state s_t be $\mathbf{c}_t$ which is a vector of concentrations of chemicals A, B, and C

$$\overset{\text{New Weights}}{\mathbf{w}_{at+1}} = \underset{\text{Weights}}{\mathbf{w}_{at}} + \overset{\text{Learning rate}}{\alpha}[\underset{\text{Target}}{T_t} - \overset{\text{Q value of action taken}}{Q_t(s_t, a_t)}]\underset{\substack{\text{Weight gradient of Q for action taken} \\ \text{i.e A vector representing the change} \\ \text{of Q(s,a) for a change of 1 in each weight}}}{\nabla_{\mathbf{w_a}} Q_t(s_t, a_t)}$$

To calculate Q from the feature vector s:
$Q_t(s_t, harvst_t) = \mathbf{c}_t \mathbf{w}_{ht}$

For MC:
$T_t = R_t$

We calculate the Q(s,harvest) values for the first sample:

$Q(s_t, harvest_t) = \begin{pmatrix} c_{at} \\ c_{bt} \\ c_{ct} \end{pmatrix} (w_{hat} \quad w_{hb} \quad w_{hc}\ )$

$Q(s_0, harvest_0) = C_{a0}w_{ha0} + C_{b0}w_{hb0} + C_{c0}w_{hc0}$
= 4*3+ 7*2 + 1*1
= 27

We now calculate the gradient

$\nabla Q_t = \left(\frac{dQ_t}{dw_{hat}} \quad \frac{dQ_t}{dw_{hbt}} \quad \frac{dQ_t}{dw_{hct}}\right)$

$\nabla Q_0 = (C_{a0} \quad C_{b0} \quad C_{c0})$
$= (4 \quad 7 \quad 1)$

Now we can calculate the new weights:
$\mathbf{w}_{h1} = (3 \quad 2 \quad 1) + 0.01[3 - 27](4 \quad 7 \quad 1)$
$= (2.04 \quad 0.32 \quad 0.76)$

Repeating for sample 2:
$Q(s_1, harvest_1) =$ 10*2.04+6*0.32+0*0.76
= 22.32

$\mathbf{w}_{h2} = (2.04 \quad 0.32 \quad 0.76) + 0.01[-15 - 22.32](10 \quad 6 \quad 0)$
$= (-1.69 \quad -0.72 \quad 0.76)$

Repeating for sample 3:
$Q(s_2, harvest_2)$ = 20*-1.69+1*-0.72+15*0.76
= -23.12
$\mathbf{w}_{h3} = (-1.69 \quad -0.72 \quad 0.76) + 0.01[5 + 23.12](20 \quad 1 \quad 15)$
$= (3.93 \quad -0.44 \quad 4.98)$

Repeating for sample 4:
$Q(s_3, harvest_3)$ = 4*3.93+19*-0.44+3*4.98
= 22.3
$\mathbf{w}_{h4} = (3.93 \quad -0.44 \quad 4.98) + 0.01[21 - 22.3](4 \quad 19 \quad 3)$
$= (3.88 \quad -0.69 \quad 4.94)$

2. For TD(0)

$T_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$

Where a_{t+1} is chosen ε-greedily. For this question we assume it always picks the actions in the episode provided.

We can set $\gamma = 1$ as there is an absorbing state but we could also select a lower value.

Need initial wait weight values.

$\mathbf{w}_{w0} = \mathbf{w}_{w4} = (3 \quad 2 \quad 1)$

Day 1:

Wait

$Q(s_4, wait_4)$ = 6*3+7*2+2*1
= 34

We also need to calculate:

$Q(s_5, harvest_5)$ = 3*3.88+8*-0.69+4*4.94
= 25.88

$\mathbf{w}_{w5} = (3 \quad 2 \quad 1) + 0.01[-1 + 25.88 - 34](6 \quad 7 \quad 2)$
$= (2.45 \quad 1.36 \quad 0.82)$

Day 2:

Harvest

$Q(s_5, harvest_5)$ = 25.88

The state after harvest is an absorbing state so Q is 0.

$\mathbf{w}_{h6} = (3.88 \quad -0.69 \quad 4.94) + 0.01[19 + 0 - 25.88](3 \quad 8 \quad 4)$
$= (3.67 \quad -1.24 \quad 4.6648)$

3. We need to find Q values for each day and choose our actions greedily.
Using the final weight vectors:
$\boldsymbol{w}_w = (2.45 \quad 1.36 \quad 0.82)$
$\boldsymbol{w}_h = (3.67 \quad -1.24 \quad 4.6648)$

Day 1:
$Q(s_6, wait_6)$ = 20*2.45+6*1.36+1*0.82
= 57.98
$Q(s_6, harvest_6)$ = 20*3.88+6*-0.69+1*4.94
= 78.4

So my episode immediately ends in a harvest.
For the other days:
Day 2:
$Q(s_7, harvest_7)$ = 10*3.88+7*-0.69+2*4.94
= 43.85
Day 3:
$Q(s_7, harvest_7)$ = 5*3.88+8*-0.69+4*4.94
= 33.64
So the orchard harvested on the highest value of Q(s, harvest) that It would have had in those 3 days.

If the orchard had waited on the third day then the closest day to harvesting would have been the one with the smallest value of $Q(s_t, wait_t) - Q(s_t, harvest_t)$

4. Increasing the learning rate overall increases the effect each sample has on determining the weights. If you set it too high, then the weights will never converge as each small error will create too large a change in the weights. If you set the learning rate too low, then the weights will take a long time to converge and will be vulnerable to getting stuck in local maxima.
5. For the inverted pendulum problem, a robot is trying to balance a rod on a hinge on top of it. As the rod can be at any angle the problem would be better modelled continuously. For a chess AI every possible state of the board can be a state in the model and so a discrete approach would be better.