To Teach or not to Teach? Decision Making Under Uncertainty in Ad Hoc Teams

Supplemental Material

Peter Stone and Sarit Kraus

ABSTRACT

This file complains supplementary material, consisting of a proof and the details of an algorithm, to accompany a paper that appears in the proceedings of AAMAS 2010. The main paper is available at `http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-AAMAS2010-adhoc.html`

3. PROOF OF THEORM 3.1

THEOREM 3.1. *It is never optimal for the teacher to pull Arm_2.*

PROOF. By induction on the number of rounds left, r.

Base case: $r = 1$. If the teacher starts by pulling Arm_2, the best expected value the team can achieve is $\mu_2 + \mu_1$. Meanwhile, if it starts with Arm_*, the worst the team expects is $\mu_* + \mu_2$. This expectation is higher since $\mu* > \mu_1$.

Inductive step: Assume that the teacher should never pull Arm_2 with $r - 1$ rounds left. Let π^* be the optimal teacher action policy that maps the states of the arms (their μ_i, n_i, and $\bar{x}_i$) and the number of rounds left to the optimal action: the policy that leads to the highest long-term expected value. Consider the sequence, S, that begins with Arm_2 and subsequently results from the teacher following π^*. To show: there exists a teacher action policy π' starting with Arm_* (or Arm_1) that leads to a sequence T with expected value greater than that of S. That is, the initial pull of Arm_2 in S does not follow π^*.

In order to define such a policy π', we define $S_1(n)$ and $S_2(n)$ as the number of pulls of Arm_1 and Arm_2 respectively after n total steps of S. As shorthand, we denote $S(n) = (S_1(n), S_2(n))$.

Similarly, define the number of pulls of Arm_1 and Arm_2 after n steps of T (e.g. when using π') as $T(n) = (T_1(n), T_2(n))$.

Next, define the relation $>$ such that $T(n) > S(m)$ iff $T_1(n) \geq S_1(m)$ and $T_2(n) \geq S_2(m)$ where at least one of the inequalities is strict. That is $T(n) > S(m)$ if at least one of the arms has pulled more times after n steps in T than after m steps in S, and neither arm has been pulled fewer times.

Finally, we define the concept of the teacher *simulating* sequence S based on the knowledge of what values would have resulted from each of the actions, starting with the teacher's pull of Arm_2 at step 1.[1] It can only do that as long as it has already seen the necessary values — otherwise it does not know what the state of the sample averages would be when it is the learner's turn to act. After n steps of the sequence T, let the number of steps that it can simulate in the S sequence be $\text{Sim}(n)$. Specifically, $\text{Sim}(n)$ is the largest value m such that $T(n) \geq S(m)$.

By way of illustration, let the values that will be obtained from the first pulls of Arm_2 be $u_0, u_1, u_2, \ldots$ and let those that will be obtained from the first pulls of Arm_1 be $v_0, v_1, v_2, \ldots$. Consider the following possible beginning of sequence S where pulls of Arm_* are marked with a^*, n is the step number, the teacher's actions are in the row marked "Teacher" and the learner's actions are in the row marked "Learner" (note that by the induction hypothesis, the teacher never pulls Arm_2 after the first step).

n:	1	2	3	4	5	6	7	8	9	10	...
Teacher:	u_0		v_1		a^*		a^*		v_4		...
Learner:		v_0		v_2		u_1		v_3		v_5	...

In this sequence, $S(0) = (0,0), S(1) = (0,1), S(2) = (1,1), S(3) = (2,1), S(4) = S(5) = (3,1)$, etc.

Meanwhile, suppose that the teacher's first action in sequence T is Arm_* and the learner's first action is Arm_1, leading to v_0. Then $T(0) = T(1) = (0,0)$ and $T(2) = T(3) = (1,0)$.

Until the learner sees a pull from Arm_2 in sequence T, it cannot simulate any steps of S: $\text{Sim}(1) = \text{Sim}(2) = \text{Sim}(3) = 0$. If the teacher's second action in T is Arm_* and learner's 2nd action is Arm_2, then in the example sequence above, $\text{Sim}(4) = 2$.

We are now ready to define the teacher's policy π' for generating T. Let n be the total number of actions taken so far. Then:

1. If $n = 0$, $T(n) > S(\text{Sim}(n))$ or $\text{Sim}(n)$ is odd, then select Arm_*;
2. Else ($T(n) = S(\text{Sim}(n))$ and $\text{Sim}(n)$ is even), select the next action of S (i.e. the action π would select if there were $r - \frac{\text{Sim}(n)}{2}$ rounds left).

Note that by the definition of Sim, it is always the case that $T(n) \geq S(\text{Sim}(n))$. Further, note that at the beginning we are in step 1 of the strategy: $T(2) = (1,0) > (0,0) =$

[1]Such simulation relies on an assumption that the payoffs from an arm are queued up and will come out the same no matter when the arm is pulled: they are not a function of the times at which the arm is pulled, or the payoffs from any other arms. However, our argument still holds if the payoffs are time-dependent and/or dependent on other arms as long as the teacher has no knowledge of the nature of this dependency.

$S(\text{Sim}(2))$. It remains to show that the sequence T resulting from using this policy π' has an expected value greater than that of S. We prove this in two cases.

Case 1: There is a least n, call it n', such that $T(n) = S(\text{Sim}(n))$ and $\text{Sim}(n)$ is even.

Until that point, the teacher keeps pulling Arm_*. We can thus show that $\text{Sim}(n') < n'$ as follows. After n' steps, there are exactly $\frac{n'}{2}$ u's and v's in the T sequence ($T_1(n') + T_2(n') = \frac{n'}{2}$). But after n' steps, there are at *least* $\frac{n'}{2}+1$ u's and v's in the S sequence ($S_1(n')+S_2(n') \geq \frac{n'}{2}+1$) because the first value is a u and all the learner's actions are u's or v's. Thus the simulation of S always lags behind T in terms of number of steps simulated: $\text{Sim}(n') < n'$.

Note that if it is ever the case that $T(n) = S(\text{Sim}(n))$ and $\text{Sim}(n)$ is odd (it is the learner's turn to act in S), then the teacher will pull Arm_* once more after which the learner will do what it would have done in sequence S after $\text{Sim}(n)$ steps. That will cause both $T(n)$ and $S(\text{Sim}(n))$ to increment by the same amount, and $\text{Sim}(n)$ to be even. Thus in the subsequent round, the teacher will switch to step 2 of its strategy.

Once the teacher has switched to step 2 of its strategy, then it will continue using that step: sequence T will follow S exactly for its remaining $2r-n'$ steps. To see that, observe that in each round, $T(n)$ and $S(n)$ will increment by the same amount, and $\text{Sim}(n)$ will increment by exactly 2, thus remaining even.

Now compare the sequences T and S. Up until the point of step n' in T and $\text{Sim}(n')$ in S, the only difference between the sequences is that there are $n' - \text{Sim}(n')$ extra pulls of Arm_* in T. There then follow $2r - n'$ steps in the two sequences that are identical. The final $n' - \text{Sim}(n')$ steps in S include at least one pull of Arm_1 or Arm_2 (the learner's first action). Thus the expected value of $T - S$ (the difference between the sum of their expected values) is at least $\mu_* - \mu_1 > 0$.

Case 2: It is never the case that $T(n) = S(\text{Sim}(n))$ and $\text{Sim}(n)$ is even. Then the teacher continues playing Arm_* throughout the T sequence (r times).

First, by the same argument as above, since the teacher always pulls Arm_*, it is always the case that $\text{Sim}(n') < n'$.

Next, we argue that $T_2(2r) = S_2(\text{Sim}(2r))$. That is, after $\text{Sim}(2r)$ steps, the next step in S is a pull of Arm_2 (because $\bar{x}_2 > \bar{x}_1$). Otherwise, S could be simulated another step further by consuming another v value from T. We show this by induction on the number of steps in the T sequence i, showing that it is always the case that $T_2(i) = S_2(\text{Sim}(i))$.

This equation holds at the beginning (e.g. when $i = 2$): $T(2) = (1, 0)$, $S(\text{Sim}(2)) = (0, 0)$, so $T_2(2) = S_2(\text{Sim}(2)) = 0$.

Now assume $T_2(i-1) = S_2(\text{Sim}(i-1))$. There are three possibilities for the next action in T. If it is a pull of Arm_* or Arm_1, then $T_2(i) = T_2(i-1)$ and $\text{Sim}(i) = \text{Sim}(i-1) \implies S_2(\text{Sim}(i)) = S_2(\text{Sim}(i-1))$, so the condition still holds. If it is a pull of Arm_2, then $T_2(i) = T_2(i-1)+1$ and $S_2(\text{Sim}(i)) = S_2(\text{Sim}(i-1))+1$ because the new u value can be used to continue the simulation of S by at least one step, and there are no additional u's in T to increase $S_2(\text{Sim}(i))$ any further. Therefore $T_2(i) = S_2(\text{Sim}(i))$.

Note that in general, $S_1(\text{Sim}(i))$ could be much greater than $S_1(\text{Sim}(i-1))$: there could be several v values from T that are then able to be used for simulating S. But if all of the available v's from T are used, we get that $T(i) = S(\text{Sim}(i))$, which violates the Case 2 assumption and puts us into Case 1 above (or will put us there one round later if $\text{Sim}(i)$ is odd).

Thus we have shown that after all $2r$ steps of T, the next action in the simulated version of S (step $\text{Sim}(2r)+1$) must be Arm_2.

Finally, we compare the expected values of T and S. As above, there are several values in common between the two sequences, namely exactly the u's and v's from T that were used to simulate the first $\text{Sim}(2r)$ steps of S (as well as possibly some pulls of Arm_*). Let the sum of these u and v values be called COMMON.

Now consider the values of T and of S that are not in common: those values from T that were not used to simulate S, and those values in S that come after the simulation ended (after step $\text{Sim}(2r)$), plus all of the pulls of Arm_*. All of these "uncommon" values in T are from Arm_* and Arm_1. In fact, exactly r of the values are from Arm_* and exactly $T_1(2r) - S_1(\text{Sim}(2r))$ of them are from Arm_1. The uncommon values from S include at most $r-1$ from Arm_* (because the first teacher action was Arm_2), and at least one from Arm_2 (step $\text{Sim}(2r)+1$).

Thus the expected values of the two sequences satisfy the following inequalities.

$\text{EV}(T) \geq r * \mu_* + [T_1(2r) - S_1(\text{Sim}(2r))] * \mu_1 +$ COMMON

$\text{EV}(S) \leq (r-1) * \mu_* + [T_1(2r) - T_1(\text{Sim}(2r))] * \mu_1 + \mu_2 +$ COMMON

Thus $\text{EV}(T)$-$\text{EV}(S) \geq \mu_* - \mu_2 > 0$.

Therefore in both cases, the expected value of sequence T exceeds that of sequence S. Since S is the best the teacher can do if it starts with Arm_2, and T is a lower bound on how well it can do otherwise, the teacher should never pull Arm_2. □

4. ALGORITHM FROM SECTION 4

The dynamic programming algorithm is detailed as pseudocode in Algorithm 1 and explained below. For every reachable combination of values, the algorithm computes the optimal action for the teacher (Arm_1 or Arm_*), denoted Act[·]; and the expected long-term value of taking that action, denoted Val[·]: the expected sum of payoffs for the optimal action and all future actions by both the teacher and the learner.

First, in Line 1, the expected value with zero rounds remaining is defined to be 0 since there are no more actions to be taken. Then, in the body of the nested `for` loops (Lines 7–49), the expected values of both teaching by pulling Arm_1 (EV_t) and not teaching by pulling Arm_* (EV_{nt}) with r rounds remaining are computed based on the stored values for the possible resulting states with $r-1$ rounds remaining.

The values of these possible resulting states are denoted as EV_{abcd} where a, b, c, and d denote the increments to m_1, n_1, m_2, and n_2 respectively between rounds r and $r-1$ (Lines 7–18). For example, Line 27 computes the expected value for not teaching when $n_1, n_2 > 0$ and $\frac{m_1}{n_1} > \frac{m_2}{n_2}$. In the current round, the teacher exploits (does not teach) by pulling Arm_* and the learner pulls Arm_1, leading to an expected return of $p_* + p_1$. This value is then added to the expected value of the resulting state with $r-1$ rounds remaining. Due to the learner's action, the value of n_1 is incremented by 1. With a probability of p_1, this action returns a payoff of 1, causing m_1 to be incremented as well. With a probability of $1-p_1$, m_1 is not incremented. Thus the expected value after the

current round is $p_1\text{EV}_{1100}+(1-p_1)\text{EV}_{0100}$. Note that there are special cases for the situations in which n_1 and/or n_2 are 0 corresponding to the assumed learner behavior as specified in Section 2 of the main paper.

Once the expected values of teaching and not teaching have been computed, they are compared in Line 43, and the Act[·] and Val[·] entries are set according to the result. Finally, the appropriate action with R rounds remaining is returned (Line 55). Note that by storing the optimal actions along the way (Act[·]), the algorithm eliminates the need to do any additional computations in the future as the number of rounds remaining (r) decreases to 1: for all possible results of the teacher's and learner's actions, the optimal teacher action in all future rounds is already stored.

Algorithm 1 TeachOrExploit(M_1, N_1, M_2, N_2, R)

Require: p_1, p_2, p_*
1: Define $\text{Val}[m_1, n_1, m_2, n_2, 0] = 0, \forall m_1, n_1, m_2, n_2$
2: **for** $r = 1$ to R **do**
3: **for** $n_1 = N_1$ to $N_1 + 2(R - r)$ **do**
4: **for** $m_1 = M_1$ to $M_1 + (n_1 - N_1)$ **do**
5: **for** $n_2 = N_2 + \max(0, R - r - (n_1 - N_1))$ to $N_2 + (R - r) - \max(0, n_1 - N_1 - (R - r))$ **do**
6: **for** $m_2 = M_2$ to $M_2 + (n_2 - N_2)$ **do**
7: $\text{EV}_{1100} = \text{Val}[m_1 + 1, n_1 + 1, m_2, n_2, r - 1]$
8: $\text{EV}_{0100} = \text{Val}[m_1, n_1 + 1, m_2, n_2, r - 1]$
9: $\text{EV}_{0011} = \text{Val}[m_1, n_1, m_2 + 1, n_2 + 1, r - 1]$
10: $\text{EV}_{0001} = \text{Val}[m_1, n_1, m_2, n_2 + 1, r - 1]$
11: $\text{EV}_{2200} = \text{Val}[m_1 + 2, n_1 + 2, m_2, n_2, r - 1]$
12: $\text{EV}_{1200} = \text{Val}[m_1 + 1, n_1 + 2, m_2, n_2, r - 1]$
13: $\text{EV}_{0200} = \text{Val}[m_1, n_1 + 2, m_2, n_2, r - 1]$
14: $\text{EV}_{1111} = \text{Val}[m_1+1, n_1+1, m_2+1, n_2+1, r-1]$
15: $\text{EV}_{1101} = \text{Val}[m_1 + 1, n_1 + 1, m_2, n_2 + 1, r - 1]$
16: $\text{EV}_{0111} = \text{Val}[m_1, n_1 + 1, m_2 + 1, n_2 + 1, r - 1]$
17: $\text{EV}_{0101} = \text{Val}[m_1, n_1 + 1, m_2, n_2 + 1, r - 1]$
18: **if** $n_1 = 0$ and $n_2 = 0$ **then**
19: $\text{EV}_{nt} = p_* + .5(p_1(1+\text{EV}_{1100}) + (1 - p_1)\text{EV}_{0100}) + .5(p_2(1+\text{EV}_{0011})+$
20: $(1 - p_2)\text{EV}_{0001})$
21: **else if** $n_1 = 0$ **then**
22: $\text{EV}_{nt} = p_*+p_1(1+\text{EV}_{1100})+(1-p_1)\text{EV}_{0100}$
23: **else if** $n_2 = 0$ **then**
24: $\text{EV}_{nt} = p_*+p_2(1+\text{EV}_{0011})+(1-p_2)\text{EV}_{0001}$
25: **else if** $\frac{m_1}{n_1} > \frac{m_2}{n_2}$ **then**
26: $\text{EV}_{nt} = p_*+p_1+p_1\text{EV}_{1100}+(1-p_1)\text{EV}_{0100}$
27: **else**
28: $\text{EV}_{nt} = p_*+p_2+p_2\text{EV}_{0011}+(1-p_2)\text{EV}_{0001}$
29: **end if**
30: **if** $n_2 = 0$ **then**
31: $\text{EV}_t = p_1 + p_2 + p_1p_2\text{EV}_{1111} + p_1(1 - p_2)\text{EV}_{1101} + (1 - p_1)p_2\text{EV}_{0111}+$
32: $(1 - p_1)(1 - p_2)\text{EV}_{0101}$
33: **else if** $\frac{m_1}{n_1+1} > \frac{m_2}{n_2}$ **then**
34: $\text{EV}_t = 2p_1 + p_1p_1\text{EV}_{2200} + 2p_1(1 - p_1)\text{EV}_{1200} + (1 - p_1)(1 - p_1)\text{EV}_{0200}$
35: **else if** $\frac{m_1+1}{n_1+1} < \frac{m_2}{n_2}$ **then**
36: $\text{EV}_t = p_1 + p_2 + p_1p_2\text{EV}_{1111} + p_1(1 - p_2)\text{EV}_{1101} + (1 - p_1)p_2\text{EV}_{0111}+$
37: $(1 - p_1)(1 - p_2)\text{EV}_{0101}$
38: **else**
39: $\text{EV}_t = p_1(1 + p_1(1+\text{EV}_{2200}) + (1 - p_1)\text{EV}_{1200}) + (1 - p_1)(p_2(1+\text{EV}_{0111})+$
40: $(1 - p2)\text{EV}_{0101})$
41: **end if**
42: **if** $\text{EV}_{nt} > \text{EV}_t$ **then**
43: $\text{Act}[m_1, n_1, m_2, n_2, r] = \text{Arm}_*$
44: $\text{Val}[m_1, n_1, m_2, n_2, r] = \text{EV}_{nt}$
45: **else**
46: $\text{Act}[m_1, n_1, m_2, n_2, r] = \text{Arm}_1$
47: $\text{Val}[m_1, n_1, m_2, n_2, r] = \text{EV}_t$
48: **end if**
49: **end for**
50: **end for**
51: **end for**
52: **end for**
53: **end for**
54: Return $\text{Act}[M_1, N_1, M_2, N_2, R]$