University of Illinois at Chicago – Department of Computer Science

Artificial Intelligence II (CS501) - Spring 2010
Instructor: Piotr Gmytrasiewicz

A solution to HW4

04/12/2010

Alessandro Panella (apanel2@uic.edu)

1 Introduction

This homework is about value of information. In particular, we are asked to evaluate the value of the information given by an extra sensor in a simplified wumpus environment. The environment is composed by 16 squares arranged in a 4x4 grid, where the agent can move. One of the squares contains gold, and the agent gets a reward of 1000 if she finds and grabs it, while she gets -1 for every other action. The agent has a sensor that tells her whether she got to a square with gold. The extra sensor we have to evaluate is a "remote gold sensor", which informs the agent about the presence of the gold in one of the neighboring squares (East, North, South, or West of the current agent position.) The sensor is not perfect, and we have to evaluate the value of this additional information for different accuracy values. Let's call the sensor *extraSensor*; mathematically, it works in the following way:

$$P(extraSensor = true | goldInNeighbor = true) = \alpha,$$

$$P(extraSensor = true | goldInNeighbor = false) = 1 - \alpha.$$

Note that the sensor returns false negatives with probability $1 - \alpha$ (not only false positives.) The agent can perform the actions GO_FORWARD, TURN_RIGHT, TURN_LEFT, and GRAB. When moving forward, there is a 0.2 chance that the agent slips and finds herself in either one of the squares on the sides of the current location, which therefore get a 0.1 chance each to be reached. The initial condition is that the agent is at the bottom left square facing East, and she knows it.

2 About the value of information

The value of information is a precise concept, informally defined as the difference in expected value between best actions before and after information is obtained.[1] One intuitive way to think about the value of information is by means of the following question: how much is the agent willing to pay for obtaining a given piece of information? This information can be of different kind. In our case, it is a sensory input. Let's define the Value of Perfect Information (VPI) more precisely. Given some previous evidence E, the agent can compute her expected utility by maximizing over the set of possible actions. In particular, using a self-explaining notation, we have that the value of the best action α is:

$$EU(\alpha|E) = \max_A \sum_i U(Result_i(A)) P(Result_i(A)|Do(A), E).$$

[1] The value of information is described in the Russell-Norvig textbook in Section 16.6, pages 600-604.

It is clear from the formula that we are taking into account different outcomes that are non-deterministically related to the action that is chosen. That is why we sum over i's, or the possible outcomes. What does this expression mean when using an Influence Diagram as the agent's internal representation? It is quite simple to see that the U in the formula is "what comes next" in terms of utility after executing a particular action. Therefore, this value is the sum of the expected future outcomes, computed as far as the agent's horizon goes. This is closely related to Bellman's equation for POMDPs, but is computed online by the agent at every step, and does not include a discount factor. When the agent obtains some new piece of information, say evidence E_j, then the value of the *new* best action α_{E_j} becomes:

$$EU(\alpha_{E_j}|E, E_j) = \max_A \sum_i U(Result_i(A))P(Result_i(A)|Do(A), E, E_j).$$

But E_j is a variable whose value is currently unknown, therefore we have to average over the possible values e_{jk} that it might assume once it is known. It follows that the mathematical definition of the VPI is:

$$VPI_E(E_j) = \left(\sum_k P(E_j = e_{jk}|E)EU(\alpha_{E_j}|E, E_j = e_{jk})\right) - EU(\alpha|E).$$

Note that the VPI is not only function of the information we are evaluating but also clearly depends on the previous evidence E that the agent has acquired.

3 Influence Diagrams

We have developed a collection of Influence Diagrams (ID's) for computing the value of the information in different configurations, that can be found in the folder `nets/` of the project. Their common file name is `MyHW4`, to which we add a suffix that identifies the particular version. They all share the same basic structure, and we have manually extended them in time up to 2, 3 and 4 time steps, which are identified by the suffix `-2s`, `-3s`, and `-4s` respectively. In the following, we describe the nodes of the implemented net. The subscript t is used to indicate the time slice.

- $agentPosition_t$ is a nature node that represents the agent's position in the world, i.e. the square she is in. The possible states are numbered from 0 to 15, starting from the bottom left square and proceeding row by row until the top right square. The initial position ($t = 0$) is set as evidence to be square $L0$, while the value of the node for subsequent time slices depends on the previous position, the agent's direction and the action previously taken, non-deterministically according to the model described in Section 1.

- $agentDirection_t$ is a nature node representing the direction the agent is facing. It can assume the values N, S, E, W. The initial direction is set to E, while the subsequent ones depend deterministically on the previous direction and the action taken.

- $goldPosition_t$ is a nature node representing the position of the gold in the environment. For $t = 0$, it can assume values from $L0$ through $L15$ and is assigned a uniform distribution. For subsequent time slices, the node can assume the additional value $NOWHERE$, meaning that the gold is no longer in the environment because the agent previously grabbed it.

- $grabsGold_t$ is an intermediate node that simplifies the net dynamics. It is set to true if the agent grabs the gold at time t, and it is false otherwise. Therefore, its parents are $agentPosition_t$, $goldPosition_t$, and the decision node $action_t$.

- $action_t$ is the decision node, corresponding to the action taken by the agent at time t.

- $gold_t$ is a nature node that represents the sensor which deterministically informs the agent about the presence of gold in the current square.

- $extraSensor_t$ is the remote gold sensor whose value of information we want to estimate.

For each time horizon, we have developed four versions of the ID. The first network has no sensor, and it serves as the baseline utility for comparing the other ones. Its file name does not have a further suffix (e.g. `MyHW4-2s.neta` is the name of the 2-step horizon ID with no sensors.) The second network only uses the $gold_t$ sensors, and is identified by the suffix `-gold` in the file name. The third network only uses the $extraSensor_t$ sensor, and is identified by the suffix `-extra`. The fourth network uses both sensors and is identified by the suffix `-both`.

4 Results

We have run tests to evaluate the value of information added by the remote gold sensor ($extraSensor_t$) for different levels of accuracy α. There are two ways for obtaining the value of the best action in Netica (they are described in the Netica help tool.) The first makes use of compilation and is the one that is triggered by normally updating the network. The second is based on *node absorption* and is run by choosing Network → Optimize Decisions from the Netica menu. This second technique only works with networks with only one utility node. Even though it is always possible to merge multiple utility nodes into a single one, the manual insertion of the node table easily becomes intractable.

We choose therefore to use the first method. However, there is something strange in the way Netica computes the expected utility in this case. When an ID is solved by compiling, **Netica ignores the informational links to the first decision to be taken**, if the value of its parents is not set as evidence.[2] This implies that the expected utilities of different actions that are returned by Netica do not take into account the fact that we will have the information given by the parents of the first decision node when the agent will have to deliberate. Note that this "strange" behavior is only adopted by Netica for the first decision; subsequent decisions work as expected and informational links are taken into account. Yet, this is not a big problem for us: to compute the real expected utility *with a sensor* we can simply use the equation of the VPI given in Section 2 "by hand." This amounts to loop on the possible values of the sensor and compute each time the utility is obtained with a particular value fixed as evidence. These utilities are averaged by weighting them on the probability of the corresponding finding, i.e. the probability of the sensor to assume that particular value.

We have implemented a simple test environment in Eclipse using the Java Netica API. The software automatically runs the tests and provides the data that we are about to analyze. We are interested in finding out how valuable is the extra sensor with different accuracy levels. We will evaluate its value in two cases:

- CASE1: The agent is not equipped with a $gold_t$ sensor, or ignores the fact that she has it, and
- CASE2: The agent uses a $gold_t$ sensor together with the $extraSensor_t$.

To carry out a more general comparison, we also assume that the accuracy of $gold_t$ varies in the same way as the extra sensor's one. The results of the tests are reported in Table 1.

The meaning of the columns is the following: for each of the three different time horizons, column GOLD VPI reports the value of information of the gold sensor with respect to the base case (no sensors). Column EXTRA VPI reports the value of information of the extra sensor, with respect to the base case. This answers our CASE1. Column BOTH VPI reports the value of information of the gold sensor and the extra sensor used jointly. The difference between BOTH VPI and GOLD VPI is the answer to CASE2. In order to view the data in a more intuitive way which is easier to interpret, we plotted it in three different charts, one for each time horizon, which are represented in Figure 1.

It is easy to see that in the 2 steps horizon case the VPI is the same for both the gold sensor and the extra sensor, and it decreases and increases linearly with accuracy. The plot is not clear because the three

[2] This comes from what I have noticed using Netica. I could not find any explanation of this online. This is not a bug, it is just the way it works. Indeed, this can be observed also on the Umbrella example used in the Netica documentation. I think there is a reason behind this choice, even though I am not sure about it. We can discuss it it you would like.

	2 Steps Horizon			3 Steps Horizon			4 Steps Horizon		
Accuracy	*Gold VPI*	*Extra VPI*	*Both VPI*	*Gold VPI*	*Extra VPI*	*Both VPI*	*Gold VPI*	*Extra VPI*	*Both VPI*
0.0	56.37	56.37	56.37	56.36	5.69	56.36	108.16	51.93	108.16
0.05	50.74	50.73	50.74	45.62	2.3	49.53	89.76	48.14	96.29
0.1	45.1	45.1	45.11	35.39	0.0	41.6	72.52	44.24	83.37
0.15	39.47	39.47	39.48	25.67	0.0	32.92	56.37	40.88	69.57
0.2	33.83	33.83	33.84	16.45	0.0	23.76	41.24	36.78	55.12
0.25	28.2	28.2	28.21	7.74	0.0	14.4	27.07	31.72	40.74
0.3	22.57	22.57	22.58	0.0	0.0	5.85	19.06	25.85	27.8
0.35	16.93	16.93	16.95	0.0	0.0	0.0	13.68	19.33	20.82
0.4	11.3	11.3	11.31	0.0	0.0	0.0	8.29	12.33	13.59
0.45	5.66	5.67	5.68	0.0	0.0	0.0	2.91	5.01	6.09
0.5	0.06	0.06	0.06	0.0	0.0	0.0	0.0	0.0	0.0
0.55	5.66	5.67	5.68	0.0	0.0	0.0	2.91	5.01	6.09
0.6	11.3	11.3	11.31	0.0	0.0	0.0	8.29	12.33	13.59
0.65	16.93	16.93	16.95	0.0	0.0	0.0	13.68	19.33	20.82
0.7	22.57	22.57	22.58	0.0	0.0	5.85	19.06	25.85	27.8
0.75	28.2	28.2	28.21	7.74	0.0	14.4	27.07	31.72	40.74
0.8	33.83	33.83	33.84	16.45	0.0	23.76	41.24	36.78	55.12
0.85	39.47	39.47	39.48	25.67	0.0	32.92	56.37	40.88	69.57
0.9	45.1	45.1	45.11	35.39	0.0	41.6	72.52	44.24	83.37
0.95	50.74	50.73	50.74	45.62	2.3	49.53	89.76	48.14	96.29
1.0	56.37	56.37	56.37	56.36	5.69	56.36	108.16	51.93	108.16

Table 1: Comprehensive test results.

lines are completely overlapping. The symmetry displayed by the result is justified by the fact that *in our model* a sensor with accuracy α provides the same information as a sensor with accuracy $1 - \alpha$. When the accuracy level is equal to 0.5, the VPI is zero for all sensors: this behavior is expected, since a 0.5 accuracy level indicates that the sensor is completely uninformative. From the table we can see some small increases in expected utility when they are used together. However, I could not determine whether those minimal differences are due to some glitches in the computation. In general, it does make sense that the combined use gives some better results. A possible question is: is it possible that the extra sensor does not increase the expected utility when used in combination with the gold sensor? The answer is surely yes: the additional sensor does change distribution over the values of some nodes in the net, but the change is "not enough" to modify the plan of action. This concept is explained in the Russell and Norvig textbook on page 602.

The situation is more interesting with a 3 steps look-ahead. In this case, we can observe that the extra sensor provides some advantage with respect to the base case *only* if the accuracy is high (or low) enough. It seems that as we compute the utility more precisely (expanding the time horizon) the value of the extra sensor decreases. However, this explanation is too simplistic. The reason is that, even though the agent is *pretty* sure of the presence (or the absence) of the gold in a neighboring square, her first action will be GRAB, because executing any other action (e.g. GO_FORWARD) would result in only one possibly successful GRAB action in the 3-steps plan, which is the only action that potentially provides some positive utility. On the other hand, if the agent is *very* confident about the gold being close to her, she wants to increase her chances to get to that location, by choosing GO_FORWARD as her first action. Moreover, we observe in the 3-steps look-ahead case that the gold sensor has equal or higher VPI than the extra sensor in all cases, and that the combined use of the two provides even better utilities.

The situation is different for 4-steps plans. Both sensors provide a positive VPI for accuracy levels different than 0.5. This can be justified by the fact that as the time horizon increases, the agent is "more free" to choose the plan of actions, and every information becomes more valuable. In particular, we observe

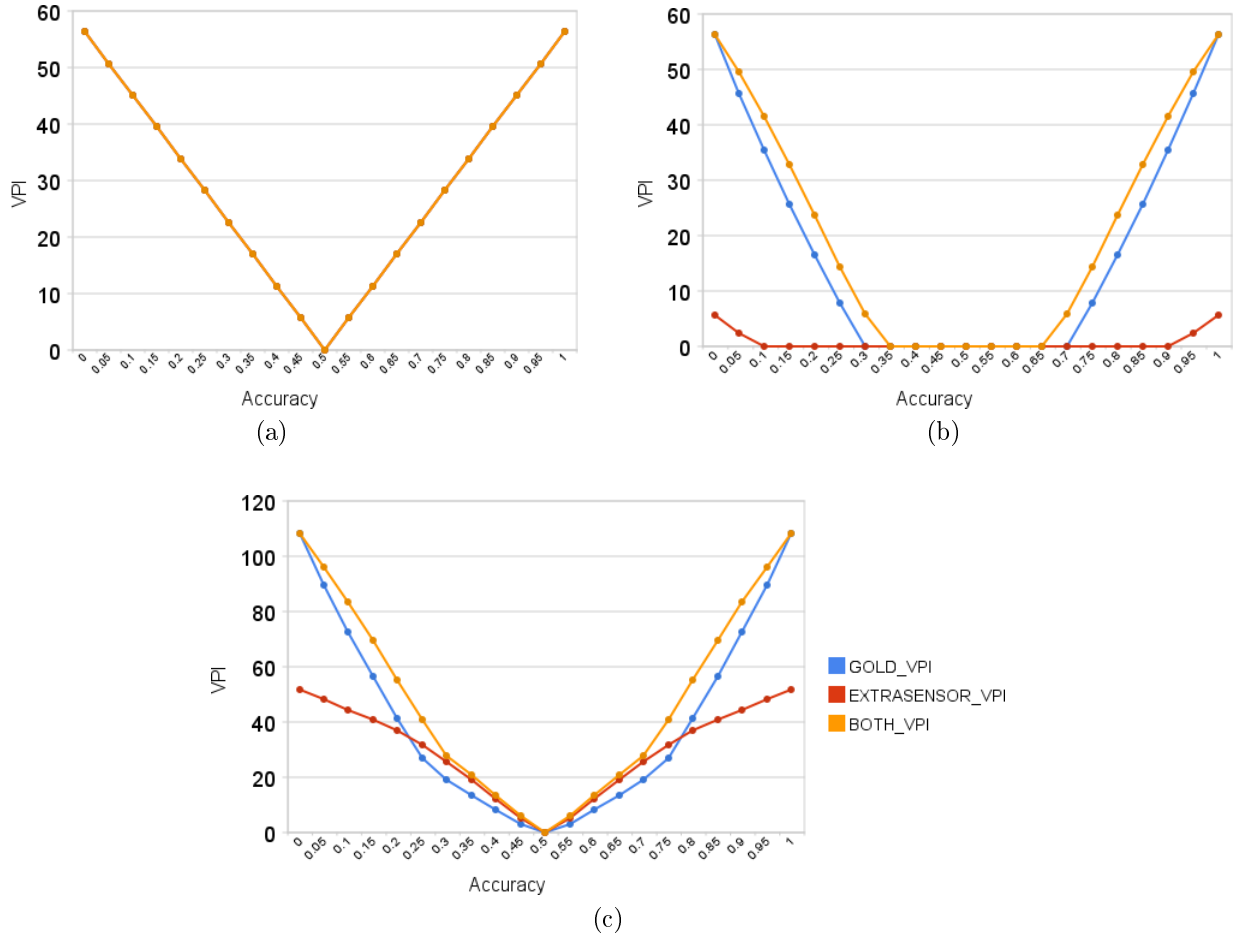

Figure 1: Plots of VPI's for different time horizons: (a) two steps, (b) three steps, and (c) four steps.

that for low accuracy, the extra sensor is now better than the gold one, while the situation is reversed for high accuracies. The combined use of both sensor is always better than the two taken singularly.