Minimal and Maximal Models in Reinforcement Learning

Dimiter Dobrev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
d@dobrev.com

Each test gives us one property which we will denote as *test result*. The extension of that property we will denote as *the test property*. This raises the question about the nature of that property. Can it be a property of the state of the world? The answer is both yes and no. For a random model of the world the answer is negative, but if we look at the maximal model of the world the answer would flip to positive. There can be various models of the world. The minimal model knows about the past and the future the indispensable minimum. Conversely, in the maximal model the world knows everything about the past and the future. If you threw a dice the maximal model would know which side will fall up and would even know what you will do. For example, it would know whether you will throw the dice at all.

Keywords: Artificial Intelligence, Reinforcement Learning, Partial Observability, Event-Driven Model, Double-State Model, Test Property, Test State.

Introduction

We try to understand the world (the environment) and for this purpose will use tests. One example of a test is the following:

If I press the door handle ⇒ the door will open.

Each test contains a condition (prerequisite). The condition in this case is that I must press the door handle. When the condition is met, the test is done and we get the test result, which can be true or false. In our example the door will open or will not open.

Each test gives us one property (*the test result*). The property in our example is "the door is locked". We cannot know the value of that property at each and every point in time. We only know that value at the time points when the test is done.

We will assume that the property is meaningful at any time point and will thus try to extend the characteristic function of that property beyond the subset of time points at which the test is done. The property may not necessarily be total or defined at each and every time point. Say, if somebody stole the door, the question of whether it is locked becomes meaningless. In other words, while we will do our best to extend the characteristic function of the property, we may not always end up with a total function.

What is the idea behind extending the test result to a test property? If I gave you a slice from a cucumber, would you be able to reconstruct the whole cucumber out of that slice? We certainly mean a mental reconstruction of the cucumber, not a physical one. That reconstruction however is not unique as it may generate various objects. E.g. from a cucumber slice we may conceive a rhinoceros if we imagine that the slice comes from its tusks. Certainly, we will try to find extensions which are as simple, natural and credible as possible.

Bear in mind that the cucumber slice is real, while the cucumber as such is imaginary or conjured up. If I let you see the full cucumber it would be easier for you to imagine it; however, you are more likely to imagine a healthy cucumber, whilst the actual one may turn out to be rotten inside. That is, you never get the full information. You always receive a fraction (a slice of information) from which you have to conjure up the whole thing.

Now let us discuss the question of what is this property (the test result and its extension). At different moments its value can be TRUE or FALSE. However, it is not a property of time because it depends on the developments playing out around the time point rather than on the time point proper. Maybe this is a property of the state of the world (the subset of states in which the property is TRUE).

Does this property depend on the past and on the future? Typically, the test takes place within a certain period of time rather that at a single moment. The test result depends therefore on the temporal context (the near past and the near future within the test period). If we take the test property it depends on a wider temporal context and may tell us something about the distant past and the distant future of the world.

Let us have the property "This letter brings good news". The test for this property is "Open the envelope and check what the letter says". The property tells us something about the future. To put it more precisely, it tells us what we will read in the letter after we open the envelope. Is it a property of the world? Does the world know what the letter says before we open it? We tend to think that yes, the world knows, but it can afford not to know, as well. For example, the majority of computer games do not bother to calculate the whole world, but take care only of the fraction of the world the gamer sees now. If the world were such a game, it would decide what the content of the letter is only when you open it. In a similar example, let us imagine life as a TV serial. In series 1354 you receive a letter, but open it 10 series later. When will the scriptwriter decide what is in the letter? When he or she writes the script of series 1354 or the script of series 1354+10? So we see that the world may or may not know in advance how the future will unfold.

Similar is the situation with the past. The property "I am back from a vacation" provides some clues about the past. The test to verify this property is: "I check whether I am on a vacation or on a business trip and then come back". We assume that if you are back from a vacation and are still at home (have not gone elsewhere), then you are still back from a vacation. That is, we extended the property to time points at which it is not tested.

Let us assume that the future of the world does not depend howsoever on whether you are back from a vacation or from a business trip. Then why should the world remember that fact at all? The question which keeps historians awake at night is whether the world remembers the past. Did a historical event leave any documents or other traces of its occurrence? The answer is that the past can be remembered but it is perfectly possible some facts to be totally discarded.

This article will discuss two models of the world – minimal and maximal. In the minimal model the world remembers the indispensable minimum from the past and knows the indispensable minimum about the future. Conversely, in the maximal model the world remembers everything from the past and knows everything about the future. In the maximal model the world knows exactly what is going to happen and even what you (the agent) would do or not do.

What are we looking for?

What is given and what are we trying to find out? In this article we will try to find an explanation of the world. With Reinforcement Learning [1] we have an agent who lives in a certain world. The world is an oriented graph similar to Figure 1. The agent follows the arrows, moves from one state to another and collects certain rewards. It is vital for the agent to explore the world and understand it well, otherwise he would not get to the rewards. Many articles assume that the world is a given entity and what we look for is a policy which would be successful in that world (e.g. [3]). In this article we will assume that the world is unknown.

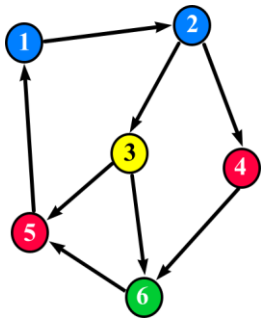

Figure 1

In Figure 1, the possible moves of the agent are represented by arrows and the possible observations are denoted with various numbers and colours. Each arrow should be tagged with a label indicating the action corresponding to that arrow. Although we have not shown these labels, the figure clearly demonstrates that sometimes there is a single arrow (possibility) and at other times there are more possibilities (at states 2 and 3). We will assume that not all actions are possible and that the transitions are nondeterministic. In other words, for a particular state/action there may not be even a single arrow with right label departing from that state, or there may well be more than one arrow with the right label.

By allowing for nondeterministic transitions we have in fact embraced randomness. In [5] we demonstrated that there are two types of randomness, predictable and unpredictable. Here we will work with unpredictable randomness (with a probability in the interval [0, 1]). This kind of randomness is also used in Nondeterministic Finite Automaton (NFA). With NFA something may or may not occur, and we do not know the probability of its occurrence. The Partially Observable Markov Decision Process (POMDP) uses predictable randomness (something occurs with a precisely determined probability). In [5] we demonstrated the equivalence of the four types of models (the deterministic one, the models with the two types of randomness, and the model which combines the two types of randomness). So we can use whichever model suits us best and have thus opted for the model with unpredictable randomness used here.

Full Observability means that we can tell which is the state just by the observation. The opposite is referred to as Partial Observability. Going back to Figure 1, when we are able see the number of the state, we enjoy Full Observability. Seeing the colour only does not tell us the full story. If all we see is a red circle we cannot not know whether it is state 4 or 5.

If we had the world's model, we would be able to foretell the future. Thus, if we find ourselves in state 4 we will predict that the next state will be 6. In a red state we will know that the next state will be either green or blue. Likewise, we can back-tell the past. Suffice it to turn the arrows in

opposite direction and the past will become the future. The only problem is that turning the arrows may force a deterministic graph become nondeterministic, but we have chosen to use nondeterministic graph models anyway.

What is given? The given body of facts is the history until the present time point, that is the sequence of actions (outputs) and observations (inputs or views).

$$a_1, v_2, a_3, v_4, \ldots, a_{t-1}, v_t$$

The numbering notation here identifies the number of the moment and not the number of the step. There are two moments in each step. At the first moment we produce information (this is our action) and in the second moment we enter what we see. That is, the step number will be the moment number divided by two.

Why do we choose to divide time in moments and not in steps? Because of the event-driven models, where the states will change at certain moments. At each step the state may change twice because the step has two moments.

There will be two types of moments: input and output moments. They will also be referred to as even and odd moments.

Note that the inputs and outputs will be vectors of scalars. We will assume that these scalars are finite. We could have stayed with Boolean vectors, but have not done so in order to avoid redundant coding (cf. [4]).

The history will also include all the incorrect (bad) moves we have tried before we play our next move.

$$bad_1, a_1, v_2, bad_3, a_3, v_4, \ldots, bad_{t-1}, a_{t-1}, v_t$$

Here we can imagine the ***bad*** element as a list or as a set of incorrect moves (because the order in which we have tried the incorrect moves is not essential). We will assume that the incorrect moves were tried at the same moment when the correct move was played (that is why the set of moves ***bad*** and the next correct move ***a*** have the same index).

Definition: For our purposes, *life* is a history which cannot be continued.

A history cannot be continued when it is either infinite or terminates in a state which does not have any arrows coming out of it. The term we used for these states in [6] was “sudden death”.

The Double-State Model

Figure 1 depicts the standard model, which is based on *steps*. We will use a *double-state* model, which is based on *moments*.

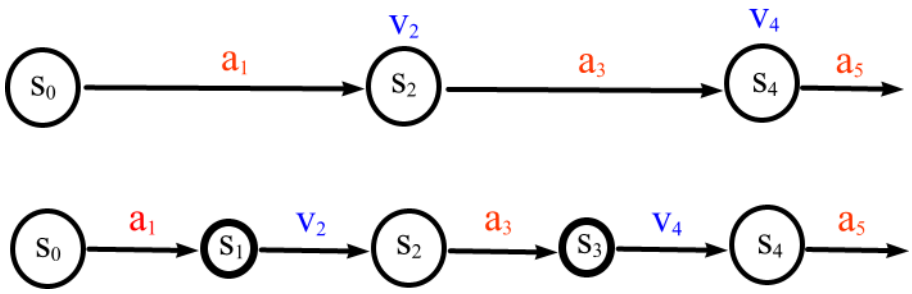

Figure 2

Figure 2 explains the difference between the standard model and the double-state model. The latter is obtained from the first by replacing each state with two states connected with an arrow. All arrows which previously entered the one state now enter the first one of the two states. The arrows which previously departed from the one state now depart from the second one of the two states. The former label of the state now goes to the arrow which connects the new states. In the double-state model, only arrows have labels and states have none. We have used smaller circles for odd-number states in order to stress that there are two types of states.

Note: Here we will deem that a state relates to a *time period* while an arrow relates to a *time point* which is the onset of the text time period. In event-driven models however, we will deem that states relate to longer time periods and arrows relate to very brief periods (lasting one moment or bit more).

Although it appears that the double-state model includes twice as many states, this is not actually the case, because when two states are equivalent from a future perspective they can be merged in a single state. The standard model allows the merging of only those states which are equivalent from both present and future perspective. I.e. in the standard model each state must remember the present, but need not do so in the double-state model. This is one of the reasons why we introduce that model. In the present article we attach paramount importance to the information we can derive from the state – what can the state tell about the past and the future. The present is part of the past because it has already occurred.

As the name suggests, the double-state model deals with two types of states: post-input and post-output. We will call them evens and odds, meaning even-number and odd-number states. Evens are more important because they are the states in which we think. In odd states we do not think, but just wait to see what information the world will give us. (It may be assumed that the world does the thinking in the odd states. So we take turns – at one time point we put the thinking hat on and at the next time point we pass the thinking hat over to the world.)

Let use as an example the world in which we play chess against an imaginary opponent. Our actions (outputs) will be vectors in the form of (x_1, y_1, x_2, y_2). These vectors describe our move. We will be able to see the opponent's move and the reward. That is, the input will be a (x_1, y_1, x_2, y_2, R) vector. The states in the double-state model will be the positions on the chessboard. Even-number states will be those in which it is the white's (ours) turn to make a move. If our move terminates the game (e.g. checkmate) the opponent must play some idle move and return to us only the reward of the game. The idle move is a vector in the form of (0, 0, 0, 0, R) and must lead to the initial position so that the game can be restarted.

Things with the standard model will be more complicated. The states will only be the positions when it is white's turn to move, but the model must also remember the opponent's move which produced that position. Hence there will be more states because "white ahead" positions will be twice less, but if there are averagely 100 ways to produce a position (from 100 different positions with 100 different moves of the opponent) the standard model will end up with 50x more states.

As mentioned already, the standard model needs to remember the present while the double-state model does not. How about the future? When we play chess against a deterministic opponent, then both models produce deterministic graphs. Let us assume that the opponent is nondeterministic and enjoys the liberty to choose among several possible moves at each position. In that case, the standard model would produce a nondeterministic graph (a single move from us triggers several different responses/counter-moves and their respective positions). The double-state model will remain deterministic, because our move creates a single, determinate position. Two or more arrows, which correspond to the possible moves of the opponent, will depart from that position (if the opponent were deterministic, there would be just one arrow).

Does the double-state model always produce a deterministic graph? Not always, but we can always transform the graph to a deterministic one. Now we will go back to our chess game example, but will slightly change the rules: we will not be able to see the opponent's move, but only the piece moved by the opponent. So we will not see (x_2, y_2). Now that we know the piece, but not the place to which it was moved, we will have several possible positions. While the obvious graph representing this kind of game is nondeterministic, we can perfectly apply a deterministic graph. If the state of the world is not a specific position on the chessboard, but a set of possible positions, then each move will trigger a single arrow which points to a set of possible positions. That is, in the first (nondeterministic) scenario the world knows more about the future than in the second scenario. In the deterministic scenario, the world does not know the exact position. What the world knows is the set of possible positions. When will the world get to know the exact position, however? Not until a later time point, when the exact position will transpire from the input (observations). Same as the letter in our TV serial. In the first case the world knows what is in the letter, while in the second case it will not decide what the letter says until you open the letter.

The Minimal Model

In our concept, a double-state model of the world will be minimal when the world's knowledge about the past and the future is limited to the indispensable minimum. In the minimal model, if two states are equivalent vis-à-vis the future, they coincide. In other words, the model does not remember anything about the past unless it is necessary in order to determine the future.

Furthermore, the minimal model is a deterministic graph. This means that the branches are pushed forward (to the future) as much as possible. Hence, nothing of the future will be known in advance (the thing will become known only when it has left its imprint on the observation, not earlier).

A minimal model is not tantamount to a least-state model. Minimalism does reduce the number of states with regards to the past, but tends to increase them with regards to the future (because we have replaced the specific possibilities with sets of specific possibilities).

The determinization procedure can always be applied so we can always obtain a deterministic graph. A deterministic graph does not necessarily mean a deterministic agent or world. For the agent to be deterministic there must not be any branches from the even states. (Here by saying that an agent is deterministic we meant that the agent is forced to play a deterministic game, because the agent has only one possible move. Generally speaking, however, a deterministic agent is understood as an agent who plays deterministically without being forced to do so.) For the world to be deterministic there must not be branches from the odd states. A deterministic graph means that the states know only the indispensable minimum about the future. (If two states are different, their past has been different, too. Therefore, the differentiating factor is their different past, not their different knowledge about the future).

The Total Model

An incorrect move is one which has not any arrow for it, i.e. this move simply cannot be made. However, we would like to enable the agent try incorrect moves without further implications. That is, the agent will only receive information that the move is not correct, but will remain in the same state.

For this purpose, to each even state (at which incorrect moves exist) we will add an odd state (see Figure 3). All incorrect moves from the even state will point to the new odd state. From the new odd state we will go back to the even state with an arrow labeled "bad". That label will be a special new vector, which we have added for our purpose and which will be received as an input only when we try an incorrect move.

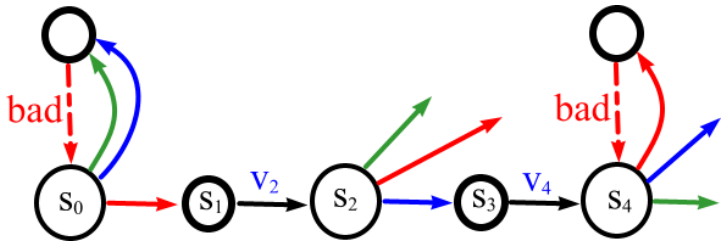

Figure 3

In Figure 3, the possible moves are represented by red, blue and green arrows. There are two incorrect moves at state s_0 and only one incorrect move at s_4. State s_2 has not an additional odd state due to the absence of incorrect moves at s_2.

This gives us a total model, where all moves can be tried, but only the correct ones change the state of the world, while the incorrect ones only return information which confirms that they are not correct. Thus we obtain a whole new total model, which describes the same world as the previous model, but has an added value in that it lets us try the incorrect moves.

The Maximal Model

Now that we saw the minimal model well and alive, we might think there is a maximal model, too. That should be the model where the state knows everything about the past, everything about which moves are correct and everything about the future.

Knowing everything about the past is easy as long as we do not stumble into branches when we walk back (in the direction opposite to that of the arrows). In other words, if the model is tree-shaped then the state will know everything about the past (we will able to reconstruct the entire history by going back from that state).

We can easily add also the information about which moves are incorrect, because this information is finite.

If we were to know everything about the future, we must dispense with any nondeterminism. We want to know which side will fall up before we throw the dice. We will thus construct a model where all nondeterminism is precipitated in the initial state. Once we select the initial state (in a nondeterministic way), the way ahead will only be deterministic.

Let us take the tree of all reachable states. (This is a true tree, because equivalent states are *not* merged into one.) We will make determinization on that tree, although this is not strictly necessary. From that tree we will obtain all policies of the world. These are sub-trees, which have no branches at observation points, but keep their branches at action points. These trees are many (cardinality of the continuum). We will take all those trees and make a model where the initial states will be the roots of all these trees.

“Policy” is maybe not the most appropriate word to use here, because a policy implies a certain objective. Here we assume that only the agent has an objective, and the world has none. If we claim that the entire world tries to help us or disrupt us, that would be far too egocentric. Nevertheless, we will imagine that in the world there are agents who are up to something (have their objectives). So, the world has not an objective as such, but we will still use “policies” to denote the various behaviours of the world.

We have thus made a model which consists of all policies of the world. Yet before life begins, the world randomly chooses one of its policies and follows it to the end of the agent’s life. The idea is to preconceive how to play the game before the games begins. We may decide that if the opponent plays a rook, we will respond with a bishop and so forth. This preconception is a policy and is depicted with an infinite tree. These trees are uncountable. We may decide to play using a certain deterministic program, but in this way we can preconceive only a computable policy. The number of computable policies is smaller (they are countable).

The so-obtained model is equivalent to the one we began with, because any history which is possible in one of these models is also possible in the other model. With the new model however the world behaves in a deterministic pattern except for the initial time point when the initial state is selected.

In that world, any randomly selected state knows almost everything about the future. The only thing it does not know is what action will the agent choose. We would like to construct a model which makes sure that the state of the world knows even this missing piece.

There is one hurdle, however. We tend to assume that the world is given and the agent is random. Therefore, the world cannot know what the agent will do because it does not have any idea which agent will come by. Now let us assume that the agent is fixed and the world may know something about the agent. The world may know, for example, that in a certain situation the agent will not play a certain move, although the move is correct and doable. In the model graph, the move not to be played by the agent will be shown with a missing arrow.

We will further imply that before life begins both the world and the agent have figured out how to play. They have selected their policies and will stick to them right to the end of the agent's life. This can be described by a tuple of two infinite trees or by one life (an infinite path in the tree). This is because the result from the application of two fixed policies is a fixed path.

Thus we arrived at the maximal model of the world. It consists of all possible lives (paths in the tree of reachable states). The only missing piece is information about the incorrect moves. To bridge this gap, we will add loops similar to those in the total model. But, this will not be a total model, because loops will be added only for incorrect moves, and not for all missing arrows. This gives the model depicted on Figure 4. (No loop at s_2 because there are no incorrect moves from that state.)

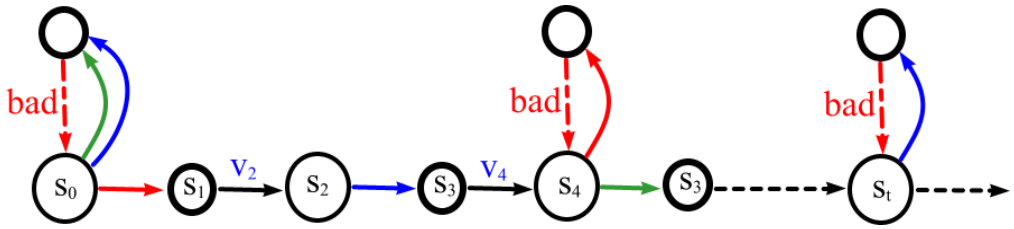

Figure 4

Figure 4 depicts only one life rather than all possible lives which form the maximal model. We care only about one life and this is the life we live in. This lets us assume that the maximal model has only one life, namely the life we are interested in. So we will lose the equivalence with the initial model, but the so-obtained model is what we need, because all other possible lives are not relevant.

Any state in our maximal model can be used to reconstruct the full history or even the full life. The forward and backward paths are branchless. Our added loops will not count as branches, because the *bad* symbol does not occur at observations after correct moves. Here the state knows which moves are incorrect, but does not know which of those moves have been tried by the agent. This information is not important to the world, because it has not any implications for the past or for the future. The information is certainly essential to the agent, because he may not know which moves are incorrect and will obviously benefit from knowing which moves he has already tried.

Conclusion

We try to understand the world, i.e. to find its model. The problem however is that there is not a single model but a raft of models. There may be unreachable and equivalent states, but this is not much of a problem. There may be parts of the world which we have not visited and will never

visit. Let us take the following example: Is there life on planet Mars? We have never been there and will never go there, so the matter is irrelevant (will not howsoever affect our life).

The most serious problem is that there are models in which the world knows more and models in which the world knows less. Which model are we looking for? The answer is that we need the maximal model so that we can obtain maximum insight of the past and maximum foresight of the future. We will even venture to predict our own behaviour, because we are part of the world and trying to understand the world means that we should also try to predict our own behaviour in that world.

In order to describe the maximal model, we will use the so called extended model. That model will present the state of the world as a vector with a huge number of coordinates (variables). They will be thousands or even countless. In theory they are countless, but for practical purposes we will select only the most interesting ones. Probably this is the model referred to by Sutton in [2] (state representation which contains many state variables).

The first coordinates (variables) to be applied in the vector which describes the extended state will represent what we see at the moment. With the double-state model, an observation is not a function of the state of the world, because if the state is even-numbered there may be many arrows with different observations pointing to it. Accordingly, an odd state may have many arrows departing from it. But, here we discuss the maximal model, which always has one incoming and one outgoing arrow. In the maximal model, therefore, it is the state of the world which determines what we see at the moment.

To what we see at the moment we will add the test properties of various tests. These are not the real-world cucumber slices. These are imaginary cucumbers which we have conjured up from the real slices. Thus, the extended model will be an imaginary rather than a real thing.

Question: If we have Full Observability, do we need to add test properties to the state vector in the extended model? Answer: Yes, we need to do so, because in Full Observability case we know the state of the world, but this state comes from another model, not from the maximal one. Having Full Observability with a maximal model would make the world so simple that it will be not interesting at all.

How can we design a test property? Consider the property “Is the door locked”. We test this property and sometimes get “locked”, sometimes “unlocked”. It would be fairly difficult to create a model which tells us when exactly the door is locked or unlocked. We would be much more successful if we imagine that the doors are more than one (especially if the doors are actually more than one). In the new model we should have an idea about which door we are standing in front of now. Some doors in that new model can be always locked, others would be always unlocked, and a third group of doors would change their state according to certain rules. In this case, we will not add to the extended model just one variable which reflects the test property. We will add many variables – one for each door (which reflects the state of the door) and one variable which indicates the exact door we are standing in front of now. In [5] this presentation was termed *test state*.

Which door are we standing in front of right now? This is determined by event-driven models which will be discussed in the next article.

References

[1] Richard Sutton, Andrew Barto (1998). Reinforcement Learning: An Introduction. *MIT Press, Cambridge, MA (1998).*

[2] Richard Sutton (2008). Fourteen Declarative Principles of Experience-Oriented Intelligence. www.incompleteideas.net/RLAIcourse2009/principles2.pdf

[3] Johan Åström. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications. 10: 174–205.*

[4] Dimiter Dobrev (2013). Giving the AI definition a form suitable for engineers. *April, 2013.* arXiv:1312.5713.

[5] Dimiter Dobrev (2017). How does the AI understand what's going on. *International Journal "Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.*

[6] Dimiter Dobrev (2018). The IQ of Artificial Intelligence. *Jun 2018. arXiv:1806.04915.*