Lecture 17 Notes — Decision Networks

Alice Gao

December 10, 2020

Contents

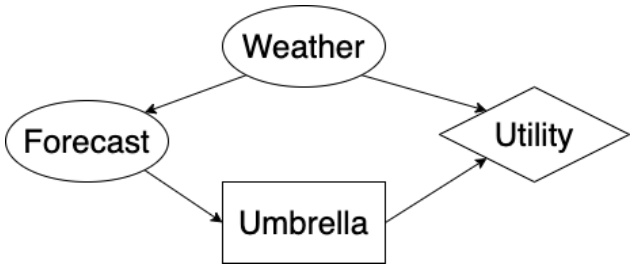
1 Learning Goals

By the end of the lecture, you should be able to

- Model a one-off decision problem by constructing a decision network containing nodes, arcs, conditional probability distributions, and a utility function
- Choose the best action by evaluating a decision network

2 The Weather Decision Network

Let's look at another example of a decision network. This network is called the weather decision network:

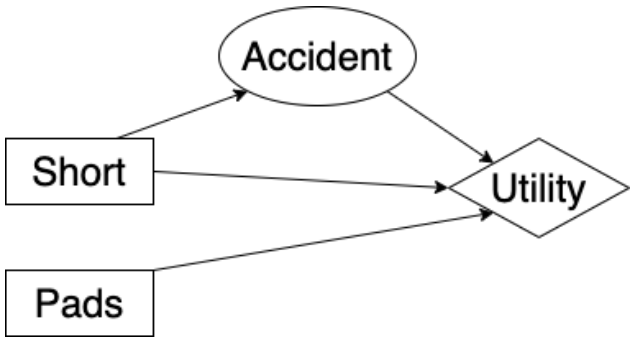


We have the weather condition to represent whether it rains or not on a particular day. Depending on the weather, we need to decide whether to take an umbrella with us, or leave it at home.

Unfortunately, we do not directly observe the weather condition. Instead we observe the weather forecast, which is a noisy signal of the actual weather condition. The forecast can be sunny, cloudy, or rainy. We make the decision of whether or not to bring an umbrella based on the forecast.

We have the utility function which represents the utility or happiness. This depends on the weather condition and whether we decide to take an umbrella or not.

This decision network may look simple at first glance, but it's a bit more complicated than the previous mail pick-up robot example. In the robot example, the decision to take either the short or long path, and the decision to wear pads or not, is made first. Then we observe the consequences of the decisions we've made. The network diagram for the mail pick-up robot is included below for reference:



In the weather decision network, the chance node forecast is the parent of the decision node umbrella. This means that when the umbrella decision is made, all possible values of the chance node forecast must be considered. This is more difficult, since the decision of whether to bring an umbrella or not depends on the forecast.

In general, when a decision node has either a chance node or another decision node as its parent, it's more difficult to work with than when all the decision nodes come first.

What are the numbers that we need from this network?

For weather, we need the prior probability over weather, so the probability of whether it rains or not.

Forecast depends on weather, so we need a conditional probability distribution specifying the probability of each value of forecast given each value of weather.

Umbrella is the decision node, so we choose the value for this when we're solving the decision network.

The utility function depends on two things: weather and umbrella. For each possible value of weather, and each possible decision for umbrella, we have a possible utility for how happy we are in that scenario.

Overall, we need three tables: one for weather, one for forecast, and one for the utility function. The following are those three tables:

$$P(\text{Weather} = \text{rain}) = 0.3$$

Forecast		
Weather	Forecast	$P(\text{Forecast}\|\text{Weather})$
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

The utility function		
Weather	Umbrella	u(Weather, Umbrella)
norain	takeit	20
norain	leaveit	100
rain	takeit	70
rain	leaveit	0

The first expression $P(\text{Weather} = \text{rain} = 0.3)$ says that there is a 30% chance it rains.

The next table describes how the forecast depends on weather. There are six possible combinations because there are two possible values for weather ("rain" or "norain") and three possible values for forecast ("sunny", "cloudy", or "rainy").

Looking at the values in the table will show that the forecast is relatively accurate. When it rains, the forecast has a 60% chance of being "rainy". When it doesn't rain, the forecast has a 70% chance of being "sunny".

The forecast is a noisy signal of weather, so it's not perfect. The other cases are when the forecast is a bad prediction of the weather, but luckily, these numbers are small. The two really bad cases, when weather is "norain" and forecast is "rainy", and when weather is "rain" and forecast is "sunny", have small probabilities of 10% and 15% respectively.

The final table describes the utility function, and there are four possibilities since there are two possible values for weather and two possible decisions for umbrella.

To make sense of the utility function, let's look at the good cases and bad cases.

What are the good cases? The two good cases are when it doesn't rain and we leave the umbrella at home, and when it does rain and we take the umbrella. For the first case, we're happy that it doesn't rain, and we're happy that we don't have to carry an umbrella around, so the utility is 100. For the second case, we're less happy that it's raining, but we're happy that we have an umbrella, so the utility is 70.

What are the bad cases? The two bad cases are when it doesn't rain and we take the umbrella, and when it does rain and we leave the umbrella at home. For the first case, we're happy that it doesn't rain, but we're less happy that we took the umbrella for no reason, so the utility is 20. For the second case, we're unhappy that it's raining, and we're even more unhappy that we left the umbrella at home, so the utility is 0.

To solve this network, we'll be using two different approaches: enumerating all policies and the variable elimination algorithm.

2.1 Solving the Weather Decision Network - Enumerating all Policies

To solve this network, we can enumerate all the policies, calculate their expected utilities, and find the optimal policy that maximizes the expected utility.

A **policy** specifies what the agent should do under all contingencies. The solution to a decision network is a policy.

What are contingencies? For the mail pick-up robot, the decisions did not depend on anything, as they did not have any parent nodes, whereas in the weather decision network, the umbrella decision node depends on the forecast. Based on the value of forecast, there are a number of possible worlds that we could be in. These possible worlds are the different contingencies that must be considered.

In general, for each decision variable, a policy specifies a value for the decision variable for each assignment of values to its parents. This is similar to planning, except we don't know what's going to happen, so we have to plan for all possibilities.

For the weather decision network, how many possible policies are there? Forecast has 3

possible values ("rainy", "cloudy", and "sunny"), and it's a parent node of the decision variable umbrella. For each value of forecast, there are two possible decisions (take or leave umbrella). The total number of possible policies is $2^3 = 8$.

The approach we're taking in this section, enumerating all policies, is the brute-force approach. There is a fixed number of possible policies. In this case, there are 8 possible policies. We can take each policy, calculate the expected utility of the agent if the agent follows that policy, then choose the policy that maximizes the expected utility of the agent.

This approach always works for every decision network. However, the problem with this approach is that even with relatively simple networks like the weather decision network, there can be a large number of policies. To enumerate through all the policies and calculate the expected utilities for each of them would be a lot of work. This is why we discuss the variable elimination algorithm later in the lecture.

Although enumerating all policies is brute-force, it's important to understand how to calculate the expected utility of a policy in this way.

Example: Consider the policy π_1 below.

- take the umbrella if the forecast is cloudy, and
- leave the umbrella at home otherwise

What is the expected utility of the policy π_1? The following is the expression for the expected utility of π_1:

$$\begin{aligned} EU(\pi_1) = & P(\text{norain}) * P(\text{sunny}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\ & + P(\text{norain}) * P(\text{cloudy}|\text{norain}) * u(\text{norain}, \text{takeit}) \\ & + P(\text{norain}) * P(\text{rainy}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\ & + P(\text{rain}) * P(\text{sunny}|\text{rain}) * u(\text{rain}, \text{leaveit}) \\ & + P(\text{rain}) * P(\text{cloudy}|\text{rain}) * u(\text{rain}, \text{takeit}) \\ & + P(\text{rain}) * P(\text{rainy}|\text{rain}) * u(\text{rain}, \text{leaveit}) \end{aligned}$$

When calculating an expected utility, we often need to enumerate all possible worlds. In this case, there are two possibilities for weather, and three possibilities for forecast, so there are going to be six possibilities.

We know the forecast is a conditional probability. For the first case, given that the weather is "norain", then probability of getting a "sunny" forecast is $P(\text{sunny}|\text{norain})$. The other cases are similar.

Next, we multiply each term by the utilities, so how happy we are in each of the six scenarios. The utility is going to depend on the actual weather condition and what we decide to do with the umbrella. Looking at the policy, we take the umbrella when the forecast is "cloudy", and leave the umbrella at home for the four other cases.

The final step is to plug in all the numbers to figure out the utility, which is done here:

$$\begin{aligned}
EU(\pi_1) &= P(\text{norain}) * P(\text{sunny}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\
&+ P(\text{norain}) * P(\text{cloudy}|\text{norain}) * u(\text{norain}, \text{takeit}) \\
&+ P(\text{norain}) * P(\text{rainy}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\
&+ P(\text{rain}) * P(\text{sunny}|\text{rain}) * u(\text{rain}, \text{leaveit}) \\
&+ P(\text{rain}) * P(\text{cloudy}|\text{rain}) * u(\text{rain}, \text{takeit}) \\
&+ P(\text{rain}) * P(\text{rainy}|\text{rain}) * u(\text{rain}, \text{leaveit}) \\
&= 0.7 * 0.7 * 100 \\
&+ 0.7 * 0.2 * 20 \\
&+ 0.7 * 0.1 * 100 \\
&+ 0.3 * 0.15 * 0 \\
&+ 0.3 * 0.25 * 70 \\
&+ 0.3 * 0.6 * 0 \\
&= 49 + 2.8 + 7 + 0 + 5.25 + 0 \\
&= 64.05
\end{aligned}$$

The expected utility of π_1 is 64.05.

The utility by itself doesn't really mean much, but it's important when we compare the expected utilities of different policies.

Problem: Consider the policy π_2 below.

- take the umbrella if the forecast is rainy, and
- leave the umbrella at home otherwise

What is the expected utility of the policy π_2?

Solution: The following is the solution to this problem:

$$\begin{aligned}
EU(\pi_1) &= P(\text{norain}) * P(\text{sunny}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\
&+ P(\text{norain}) * P(\text{cloudy}|\text{norain}) * u(\text{norain}, \text{leaveit}) \\
&+ P(\text{norain}) * P(\text{rainy}|\text{norain}) * u(\text{norain}, \text{takeit}) \\
&+ P(\text{rain}) * P(\text{sunny}|\text{rain}) * u(\text{rain}, \text{leaveit}) \\
&+ P(\text{rain}) * P(\text{cloudy}|\text{rain}) * u(\text{rain}, \text{leaveit}) \\
&+ P(\text{rain}) * P(\text{rainy}|\text{rain}) * u(\text{rain}, \text{takeit}) \\
&= 0.7 * 0.7 * 100 \\
&+ 0.7 * 0.2 * 100 \\
&+ 0.7 * 0.1 * 20 \\
&+ 0.3 * 0.15 * 0 \\
&+ 0.3 * 0.25 * 0 \\
&+ 0.3 * 0.6 * 70 \\
&= 49 + 14 + 1.4 + 0 + 0 + 12.6 \\
&= 77
\end{aligned}$$

The expected utility of π_2 is 77.

By comparison, policy π_2 is better than policy π_1. If we were to only choose between these two policies, we would choose π_2.

The calculation for the expected utilities of π_1 and π_2 are very similar to each other. In order to solve the entire problem using this approach, we need to go thorough all eight possible policies and calculate the expected utility for each policy, and then decide on the best one. We're not going to go through all the other policies here because it would be too tedious, but the calculations will be similar to what we did for π_1 and π_2.

2.2 Variable Elimination Algorithm for Decision Networks

As mentioned before, enumerating all the policies and calculating their expected utilities is a bit tedious and inefficient because even for a small decision network, there could potentially be a large number of possible policies. As such, we can use the variable elimination algorithm to find the optimal policy and calculate the expected utility of this optimal policy.

Since the weather decision network is more complex than the mail pick-up robot network, we need to modify the variable elimination algorithm slightly from before. The following is the general form of the variable elimination algorithm which is going to work for any decision network where we're able to totally order all of the decision nodes:

Variable Elimination Algorithm

1. Remove all variables that are not ancestors of the utility node.

2. Create factors.
3. While there are decision nodes remaining
   (a) Sum out each random variable that is not a parent of a decision node.
   (b) Find the optimal policy for the last decision node.
4. Return the optimal policies.
5. Sum out all remaining random variables to get the expected utility of the optimal policy.

The first step of the algorithm is to remove all variables that are not ancestors of the utility node. If a variable is not an ancestor of the utility node, it does not influence the utility, so it's irrelevant to our happiness and we can get rid of all such variables.

The next step in the algorithm is to create factors. Similar to how the variable elimination algorithm is applied to Bayesian networks before each chance node that has a conditional probability distribution, we create corresponding factors for each chance node. We also create a corresponding factor for the utility node. However, we don't create factors for decision nodes since we decide on their values through the policies of the decision nodes.

The next step in the algorithm is a loop that is looped through as long as there are decision nodes remaining. We eliminate the decision nodes one by one through each iteration of the loop.

How is a decision node eliminated? While there are still decision nodes remaining, we go through the entire network, find all random variables that are not parents of decision nodes, and sum out all these random variables.

The reason for this is that when we're trying to make a decision for a decision node, only the parents of that decision node are relevant. All other chance nodes are not relevant for that decision node. As long as a random variable is a parent of some decision node, it may be relevant when deciding on that decision. As such, we can sum those random variables that are not relevant out.

Once all the variables that are not parents of a decision node are summed out, we take all of the decision nodes and think of them as being ordered from first to last. Given this ordering in this iteration of the loop, we consider the last decision node. We eliminate that last decision node by finding the optimal policy for that decision node.

In particular, finding the optimal policy of the last decision node will involve finding one factor which contains that decision node and some of its parents. There should be a single factor that contains that decision node with some of its parents, and we use that factor to find the optimal policy for that decision node. Once we find this optimal policy, we eliminate that factor by fixing the decision variable to be the optimal policy that we chose.

This loop is looped through as many times as needed depending on how many decision nodes there are in the network. In every iteration of the loop, we eliminate the last decision node in the sequence and once we eliminate a decision node, we may have more random variables

that are not parents of decision nodes. So we go through another iteration of the loop to eliminate those those random variables.

Once the optimal policies are determined for all decision nodes, we can simply take all of the optimal policies and return them.

Finally, we can multiply all the final factors together and sum out all the random variables, resulting in a single number that will be the expected utility of the optimal policy.

2.3 Solving the Weather Decision Network - Variable Elimination Algorithm

Let's now apply the variable elimination algorithm to the weather decision network. We want to decide whether to take the umbrella with us or not based on the weather forecast. The utility will be based on the actual weather condition and the umbrella decision. We will go through the variable elimination algorithm step by step.

Step 1:

The first step is removing all variables that are not ancestors of the utility node. There are only three other variables besides the utility node and they are all ancestors of the utility node, so there is nothing to be done in this step.

Step 2

The second step is to create factors. Looking at the original network diagram, there is the chance node weather, so we'll create a factor for it called f_1(Weather). Then there is the chance node forecast which has weather as its parent, so we'll create a factor for it called f_2(Forecast, Weather). Umbrella is a decision node, so no factor is needed for that. Finally, there is the utility node which depends on weather and umbrella, so we create a third factor called f_3(Weather, Umbrella).

Step 3

The third step is to step into the loop which considers the decision nodes from last to first and eliminates them one by one until there are no decision nodes remaining.

Step 3 (a):

To do this, we first need to sum out any random variable that is not a parent of any decision node. Forecast is a parent of umbrella which is a decision node, so we should not sum it out. However, weather is not a parent of any decision node, so we should sum it out.

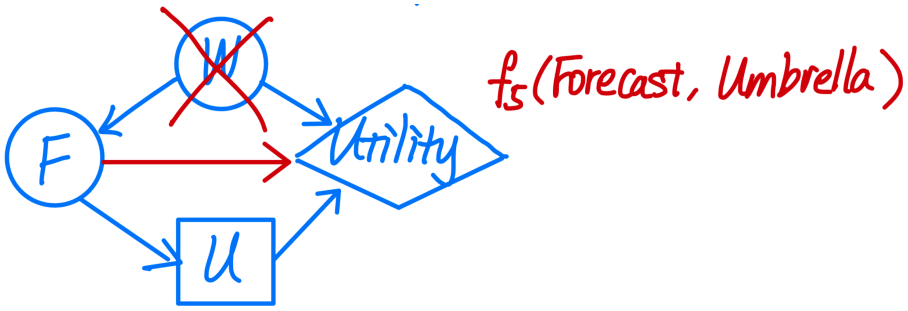
How do we sum out weather? We do this by looking at which factors contain the variable weather, multiply all these factors together to produce a new factor, and then sum out weather from this new factor. There are three factors that contain weather: $f_!$, f_2, and f_3. We multiply these factors together to product a factor called f_4 which contains weather, forecast, and umbrella, so f_4(Weather, Forecast, Umbrella). Then we sum out weather from f_4 to get f_5(Forecast, Umbrella).

The following is the result of calculating f_4 and f_5:

f_4(Weather, Forecast, Umbrella)			
Weather	Forecast	Umbrella	val
norain	sunny	takeit	9.8
norain	sunny	leaveit	49
norain	cloudy	takeit	2.8
norain	cloudy	leaveit	14
norain	rainy	takeit	1.4
norain	rainy	leaveit	7
rain	sunny	takeit	3.15
rain	sunny	leaveit	0
rain	cloudy	takeit	5.25
rain	cloudy	leaveit	0
rain	rainy	takeit	12.6
rain	rainy	leaveit	0

f_5(Forecast, Umbrella)		
Forecast	Umbrella	val
sunny	takeit	12.95
sunny	leaveit	49
cloudy	takeit	8.05
cloudy	leaveit	14
rainy	takeit	14
rainy	leaveit	7

Thinking about this variable elimination process graphically, the summing out of weather from the network is like getting rid of the weather node from the original network and re linking the rest of the nodes. This results in the single factor f_5 which contains Forecast and Umbrella, but also contains the effect of weather "blended" into it as well. This is show visually in the image below:



Step 3 (b):

The next step in the loop is finding the optimal policy for the last decision node. Since umbrella is the only decision node in the network, we need to find the optimal policy for it. In the general case, we may still have multiple factors left. Among these factors, we look for a single factor which contains the decision node umbrella, plus some subset of its parents. Since there's only one factor left, f_5, that's the candidate.

We can verify that f_5 is the candidate by looking for a few things. Firstly, it contains the decision node we're interested in which is umbrella. It also contains the parent of umbrella, which is forecast. Therefore, it fits the general requirements.

Now that we have factor f_5 as the candidate, we need to eliminate umbrella. The way we do this is to maximize the value in the factor by choosing a values for umbrella for every value assignment to its parents. In this case, for every value assignment of forecast, we choose a value for umbrella to maximize value.

Looking at the f_5 table, when the forecast is sunny, the expected utility if we take an umbrella is 12.95 and the expected utility is 49 if we leave the umbrella. Therefore, it's better to leave the umbrella s the best policy in this case is to leave the umbrella. Similar logic is used for when the forecast is cloudy or rainy, and the result is the optimal policy for umbrella, put into table form:

The optimal policy for Umbrella	
Forecast	Umbrella
sunny	leaveit
cloudy	leaveit
rainy	takeit

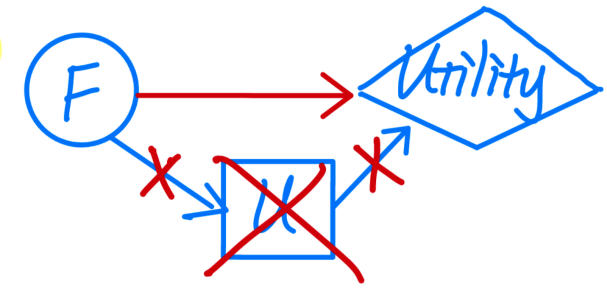
In this case, we only take an umbrella if the forecast is rainy, and leave it at home otherwise.

For the weather decision network, umbrella is the only decision node, so once the optimal policy for umbrella is found, we can stop if the purpose is just to find the optimal policy. However, to show the entire variable elimination algorithm, we can keep going and apply the algorithm to calculate the expected utility of the optimal policy.

Suppose there are more decision nodes that need taken care of. When we figure out the optimal policy for umbrella, we also eliminate it from the factors being considered. Removing umbrella from f_5 results in a new factor f_6:

f_6(Forecast)	
Forecast	val
sunny	49
cloudy	14
rainy	14

We can remove umbrella from this factor because it's deterministic given forecast. There is a deterministic way of figuring out what to do with umbrella, so given this policy, it doesn't influence utility in any way. Similar to how we eliminated the weather node from the graphical representation, we can get rid of the umbrella node along with its corresponding links:



In general, we may want to loop through the last two steps multiple times if there were multiple decision nodes, but in this case, there is only one decision node, so we are done with the third step in the algorithm.

Step 4:

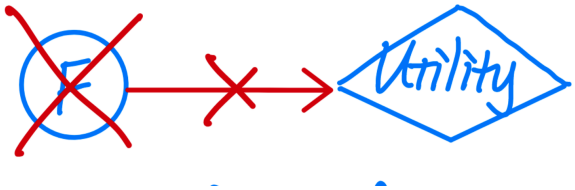
In the fourth step, we simply return the optimal policies. In this case, we just return $f_6(\text{Forecast})$.

Step 5:

For the final step, we use the final remaining factors to calculate the expected utility. If there were multiple factors left, we would want to multiply all the remaining factors together to get the expected utility. In this case, however, there is only one factor left.

We ultimately end up with one remaining factor, and with this, we sum out all the remaining random variables from this factor. In this case, we sum out forecast, and the result will be just a number, which is the expected utility of the optimal policy.

To sum out forecast, we just need to take all the values and add them together, which ends up in an expected utility of $49 + 14 + 14 = 77$ for the optimal policy. Representing this graphically, this is equivalent to getting rid of the Forecast node and the link:

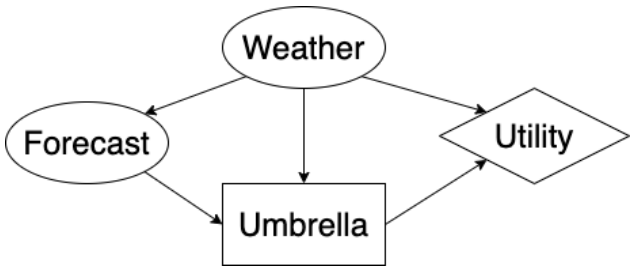


Now we have the only utility node left which corresponds to the expected utility of 77 that we calculated.

2.4 The Value of Perfect Info on Weather

Now, let's ask a different question about the weather decision network. In the original network, we use forecast as a noisy signal of weather, and we base our decision on this noisy signal.

The new question to think about is what if we can observe weather directly? How much can we increase the expected utility if we're able to observe weather directly. We modify the decision network we worked with before slightly by adding a directed edge from weather to umbrella, since now, we can observe it directly:



The decision node umbrella now depends on forecast and weather. How does this effect the utility? One would expect the utility to increase since we have better information than before.

We can calculate how much it increases by applying the variable elimination algorithm step by step to solve this problem.

Step 1:

The first step is to remove all the random variables that are not ancestors of the utility node. All of the nodes in the graph are ancestors of the utility node, so similar to before, nothing is done in this step.

Step 2:

The second step is to define the initial factors. This step is exactly the same as before. The three factors that we define are for the weather node, the forecast node, and the utility node. The three respective factors are f_1(Weather), f_2(Forecast, Weather), and f_3(Weather, Umbrella).

Notice that in the new decision network, umbrella has a new parent. However, the initial factors didn't change. This is because we didn't define a factor for umbrella, since we don't define factors for decision variables when using the variable elimination algorithm.

Step 3:

The third step is entering the loop.

Step 3 (a):

The first step in the loop is to sum out all the random variables that are not a parent of any decision node. Looking at the new diagram, there is still only one decision node: umbrella. There are two chance nodes in the diagram: weather and forecast. Both are parents of umbrella, so there are no random variables that are not parents of a decision node. Therefore, there is nothing to be done at this step.

Step 3 (b):

The second step in the loop is to find the optimal policy for umbrella. To do this, we look for a factor that contains umbrella and a subset of umbrella's parents.

The factor we're looking for here is f_3. It contains umbrella, and it contains one parent of umbrella which is Weather. With this factor, we need to determine the best policy for umbrella. Below is f_3(Weather, Umbrella):

f_3(Weather, Umbrella)		
Weather	Umbrella	u(Weather, Umbrella)
norain	takeit	20
norain	leaveit	100
rain	takeit	70
rain	leaveit	0

When the weather is norain, the best thing to do is leave the umbrella at home, which achieves a utility of 100. When the weather is rain, the best thing to do is take the umbrella,

which achieves a utility of 70. From this, we produce a new factor f_4(Weather) which is the result of eliminating Umbrella from f_3(Weather, Umbrella):

f_4(Weather)	
Weather	value
norain	100
rain	70

At this point, if this is the solution we were looking for, then we're done. However, to apply the entire variable elimination algorithm, we keep going until the end where we calculate the expected utility of the optimal policy. Remember that the question we want to answer is how much better can we do by having direct information about weather.

Since there was only one decision node in the network, we are done with the third step in the algorithm

Step 4:

In the fourth step, we simply return the optimal policies. In this case, we just return f_4(Weather).

Step 5:

The final step of the algorithm is to sum out all the remaining variables to get the expected utility of the optimal policy. At this point, there are three factors left: f_1(Weather), f_2(Forecast, Weather), and the new factor f_4(Weather). These are shown below:

f_1(Weather)	
Weather	value
rain	0.3
norain	0.7

f_2(Forecast, Weather)		
Weather	Forecast	P(Forecast|Weather)
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

f_4(Weather)	
Weather	value
norain	100
rain	70

The remaining variables in the remaining factors are weather and forecast. Let's sum out forecast first since it only appears in one factor. We take f_2(Forecast, Weather) and sum out Forecast to get f_5(Weather):

f_5(Weather)	
Weather	value
norain	1
rain	1

When we sum out forecast, we add up all the probabilities, and they all sum up to 1 for each values of weather. There is a reason for this which is explained a little later.

Now, we are left with factors f_1(Weather), f_4(Weather), and f_5(Weather):

f_1(Weather)	
Weather	value
rain	0.3
norain	0.7

f_4(Weather)	
Weather	value
norain	100
rain	70

f_5(Weather)	
Weather	value
norain	1
rain	1

Since the only variable left is weather, we need to multiply all the factors together to sum it out. Multiplying all three factors together, we get the final f_6(Weather) with a value of 21 if the weather is rain, and 70 if the weather is norain:

f_6(Weather)	
Weather	value
rain	21
norain	70

Summing out weather, we get the final value $21 + 70 = 91$ for the expected utility of the optimal policy for this decision network.

This is the end of the variable elimination algorithm for this network.

Now, let's compare the expected utilities for the original decision network, and the network with direct knowledge of weather. The expected utility of the original decision network was 77. The expected utility of the new network we just calculated is 91.

The difference between these two expected utilities is what is called the value of getting perfect information on weather. In this case, this value is $91 - 77 = 14$.

Why would we want to get this value? In the real world, when we have a random variable, it's something that we don't have control over. However, we can often perform tests or do additional things to get better information about a particular random variable. We can use this kind of analysis to figure out what effect this better information has.

There is something interesting about this decision network when comparing it to the original decision network. In the original decision network, we figure out the optimal policy for umbrella based on forecast. For the new decision network, umbrella technically has two parents: weather and forecast.

However, for the optimal policy for the new decision network where we directly know what weather is, it only depends on weather. It's as if the edge between forecast and umbrella doesn't exist. Why is this?

You may have thought of this before, but once we can directly observe weather, we don't need forecast anymore. It's not useful. Forecast is a noisy signal that does not give us any additional information, so the optimal policy does not use it.

There are two places in the algorithm where we can see this happening.

The first place is when we're finding the optimal policy for umbrella, where utility does not directly depend on forecast.

The second place is when we sum out forecast from factor f_2(Weather, Forecast) to produce f_5(Weather) and we get values of 1 for both rain and norain. This happens when a variable is considered that is irrelevant to whatever we're trying to solve. In this case, it's the optimal policy. Since this factor only contains 1's, it doesn't effect the final result in any way after performing the multiplication to sum out weather.

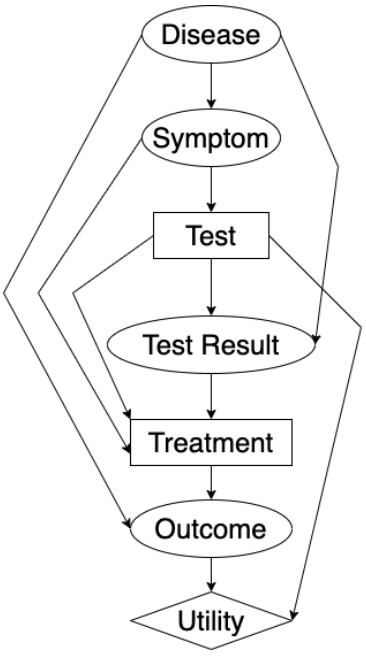
At this point, you should be able to figure out the value of getting perfect information of a particular random variable by assuming that this information is directly available, calculating the expected utility in this scenario, and then calculating the improvement in expected utility.

3 Applying the Variable Elimination Algorithm on a Medical Diagnosis Scenario

So far, we have only seen examples dealing with networks with one decision node. In the following example, we're going to look at the application of the variable elimination algorithm on a network with multiple decision nodes. This is to illustrate the full extent of the variable elimination algorithm, where we consider the decision nodes in reverse order.

We won't use any numbers in this example. We'll just walk through the steps of the algorithm, including the operations that will need to be performed at each step.

The network we're going to be working with relates to a general medical diagnosis scenario, and is represented graphically below:



A patient has some disease. Given this disease, the patient is going to show a certain symptom. The doctor cannot observe the disease directly, but they could observe the symptom.

Given the symptom, the doctor would decide on what kind of test to perform. This is the first decision node in the network.

Given the test performed, and given the actual disease that the patient has, the test will produce some test result. After the test result is observed, the doctor would decide on what treatment to prescribe, which is based on the symptoms, the test performed, and the test result. This is the second decision node in the network.

The treatment will cause some outcome to occur, which is dependent on the treatment, and the actual disease that the patient has. Once there's an outcome, the patient has some utility in the scenario. The utility is based on the outcome, but is also based on the test performed, since performing the test will incur some costs which will have a negative impact on the utility for the patient.

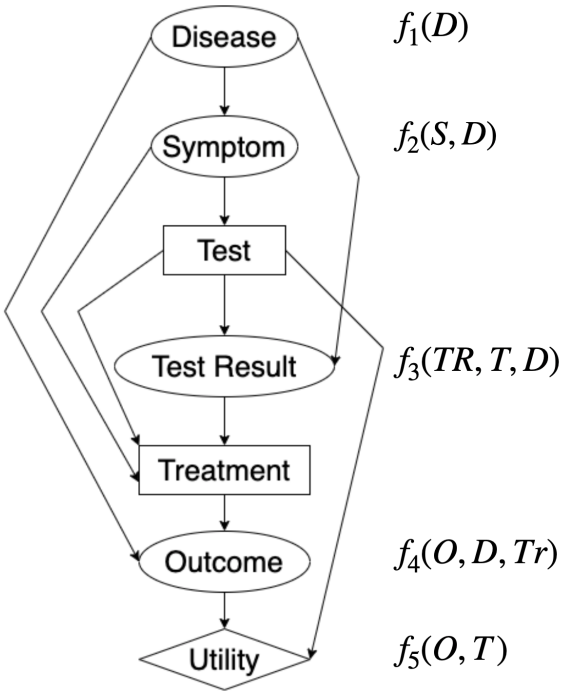
This is a fairly generic medical diagnosis scenario with two decisions: the doctor must choose the test, and choose the treatment. We can apply the variable elimination algorithm step by step.

Step 1:

The first step is to remove any variable that is not an ancestor of the utility node. This network is designed in such a way that the utility node is the last one, so every node is an ancestor of the utility node. Nothing is to be done at this step.

Step 2:

The second step is to define a factor for each conditional probability distribution, and for the utility node. We start with the five factors shown below:



There are five starting factors.

The first factor is for disease, and since it doesn't have a parent, disease is the only variable in that factor: $f_1(D)$.

The second factor is for symptom which has disease as its parent: $f_2(S, D)$.

The third factor is for test result which depends on test and disease: $f_3(TR, T, D)$.

The fourth factor is for outcome which has disease and treatment as parents: $f_4(O, D, Tr)$. TR is used to represent test result, and Tr is used to represent treatment in this notation.

The final factor is for the utility node which depends on outcome and test: $f_5(O, T)$.

Step 3:

The third step is to enter the loop. Since we iterate through the loop multiple times in this example, we label each loop at Loop 1, Loop 2,... to see which step we're in.

Loop 1:

Step 3 (a):

The first step in the loop is to sum out each random variable that is not a parent of a decision node.

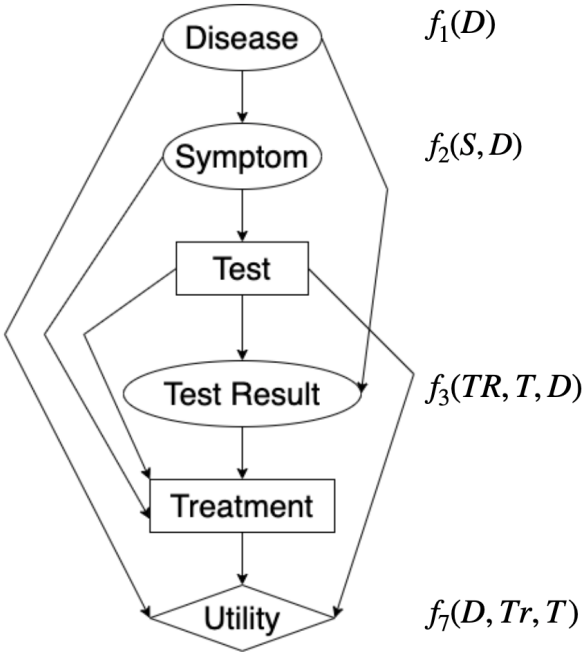
Symptom is a parent node of test, which is a decision node, and test result is a parent node of treatment, which is also a decision node. There are two random variables, disease and outcome, that are not parent nodes of a decision node, so we must sum these two variables out.

Let's sum out outcome first. How do we do this?

We need to look for factors that contain outcome. There are two factors that contain outcome: $f_4(O, D, Tr)$ and $f_5(O, T)$. We multiply these two factors to get

$$f_4(O, D, Tr) \times f_5(O, T) = f_6(O, D, Tr, T).$$

Then we sum out outcome from f_6 to get $f_7(D, Tr, T)$. The following is the updated decision network after summing out outcome:



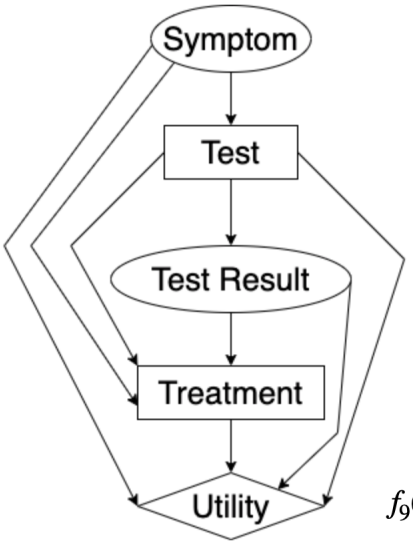
The effect of summing out outcome from the network is simply removing the node, but once it's removed, other nodes that are connected to it will be connected with each other. For the resulting network, outcome used to be connected to treatment, disease, and utility. In the resulting network, utility is directly connected to disease and treatment.

Next, we sum out disease. How do we do this?

All four factors contain disease, so we need to multiply all four factors to get

$$f_1(D) \times f_2(S, D) \times f_3(TR, T, D) \times f_7(D, Tr, T) = f_8(D, S, TR, T, Tr)$$

Then we sum out disease from f_8 to get $f_9(S, TR, T, Tr)$. The following is the updated decision network after summing out disease:



Disease used to be connected to test result and utility. Now, test result is directly connected to utility.

At this point, we have summed out all the variables that are not parent nodes of a decision node, so we can move on to the next step.

Step 3 (b):

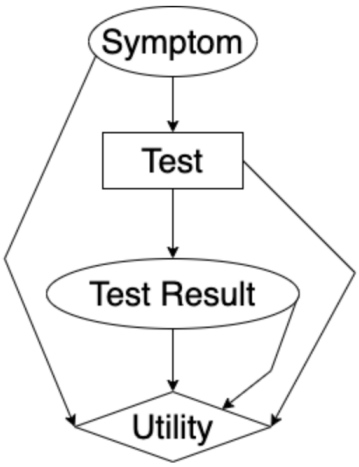
For the second step in the loop, we must find the optimal policy for the last decision node.

How do we determine the ordering of the decision nodes? The earlier decision node test is a parent of the later decision node treatment, so this parent-child relationship is the ordering we follow. The parent node test is first in the ordering, and the child node treatment is last in the ordering.

Therefore, we need to find the optimal policy for treatment. There is only the factor $f_9(S, TR, T, Tr)$ in the network at this point. f_9 contains some of the parents of treatment and treatment itself, so it's the factor we will deal with.

If we were calculating this, we would have a table for f_9 with columns for symptom, test result, test, and treatment, and a final column for expected utility. Given this table, we would find the optimal policy for treatment by choosing the value for treatment that maximizes expected utility for each combination of symptom, test, and test result. This will result in a new factor $f_{10}(S, TR, T)$.

Once we do this, we essentially eliminate the treatment node from the network. The following is the updated decision network after finding the optimal policy for treatment:



$f_{10}(S, TR, T)$

The treatment node was originally connected to every other node, so once it's eliminated, it will result it the other nodes being connected to each other, though those nodes were already connected to each other, so no new connections were added.

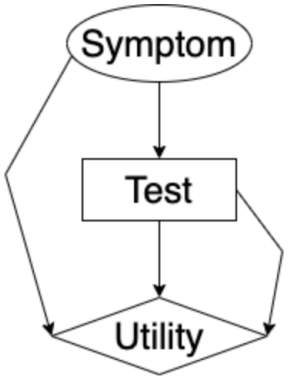
This is the end of the first iteration of the loop. We've taken care of one decision node. Since there is still a decision node remaining, we must go through another iteration of the loop.

Loop 2:

Step 3 (a):

The first step in the loop is to sum out each random variable that is not a parent of a decision node.

Now that treatment has been eliminated, test result is no longer a parent node of any decision node, so we must eliminate it. Since there's only one factor, we don't need to do any multiplication. We sum out test result from f_{10} to get $f_{11}(S, T)$. The following is the updated decision network after summing out test result:



$f_{11}(S, T)$

At this point, there are no more random variables that are not parent nodes of a decision node, so we can move on to the next step.

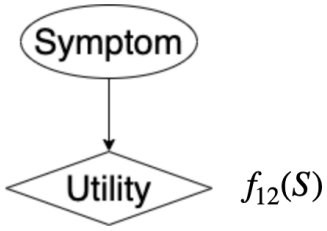
Step 3 (b):

For the second step in the loop, we must find the optimal policy for the last decision node.

In this case, there's only one decision node left, so we find the optimal policy for the decision node test. There is only the factor $f_{11}(S, T)$ which contains test itself, and its only parent symptom.

Again, we imagine this factor as a table with two columns for symptom and test, and an additional column for expected utility. Given this table, we need to choose the value for test that maximizes the expected utility for each value of symptom. This will result in a new factor $f_{12}(S)$.

We also eliminate the test node from the network. The following is the updated decision network after finding the optimal policy for test:



At this point, there are no more decision nodes left in the network, so we can step out of the loop.

Step 4:

In the fourth step, we simply return the optimal policies. In this case, we just return $f_{12}(S)$.

Step 5:

The final step of the algorithm is to sum out all the remaining variables to get the expected utility of the optimal policy. Since there is only one factor left, we take $f_{12}(S)$ and sum out symptom to get one number which is the expected utility of the optimal policy that we're looking for. After summing out symptom, the only node remaining is the utility node, which means that we've completed the variable elimination algorithm.