

**Advanced Artificial Intelligence**

CMP9132M | Assessment Item 1

Peter Hart | 12421031

The following report will entail an appraisal and analysis of a range of artificial intelligence techniques and consideration for the application of effective decision-making, problem solving and learning for solving a given issue. The solutions that are being described for each task have been developed in the programming language ‘MATLAB’ and can be located within the appendices of this report.

**Task 1.A**

The first task of this assignment involves the ‘rare disease and test problem’ scenario where the system environment being considered situates a total of two variables, ‘*d*’ – the person has the disease and ‘*t*’ – the test is positive. The problem at hand consists of a computation for the probability that the person has the disease given the test was positive, the variable equivalent representation being: ‘’. Several probability variables are known and are assigned an initial probability distribution value at run-time by the user, these include ‘’: the prior probability of having the disease, ‘’: the probability that the test is positive given the person has the disease and ‘’: probability that the test is negative given the person does not have the disease.

One issue presented with the task at hand is that it is difficult to calculate the probability of ‘’ directly due to no or limited information, however more information could be grasped by applying the Bayes Rule which enables the probability of ‘’ to be rearranged and computed in terms of ‘’. In essence, the Bayes Rule entails calculating the probability of ‘’ by multiplying the likelihood of ‘’ by the prior probability of ‘’ divided by the likelihood that ‘’ will occur. By substituting the probabilities involved in this system, the Bayes Rule is evaluated as shown in the formula below:

However, this in turn presents another issue, this being the lack of information regarding the probability of ‘’. The outcome of this is that it obstructs the possibility of being able to utilise Bayes Rule, as it is currently impossible to substitute a value for the prior probability ‘’ and therefore incapable of completing the Bayes Rule substitution. Therefore, a critical objective for this task was to estimate the probability distribution of ‘’ by inferring information from the known probabilities within the system. Overall, the probability of ‘’ can be found by evaluating both the positive and negative cases of ‘’ and ‘’, which were provided as input variables by the user. Assuming that these variables contain normalised values, the sum of the positive and negative cases of both variables should sum to one. Therefore, the negative cases of these variables can be established by subtracting the values of the positive cases away from 1.0.

As a result, these estimated negative variables can be substituted back into the previously discussed Bayes Rule, where the previously discussed equation can be rearranged to estimate the value of ‘’.

As a consequence, the objective of estimating a probability value for the variable ‘’ has been completed, therefore this value can then be input back into the original Bayes Rule equation to estimate a final value which is indicative of the true value for the probability variable ‘’. Overall, the solution for this task has been adapted such that the program will prompt the user to input the initial probability values for ‘’, ‘’ and ‘’ on each iteration that the program runs, so a different result can be easily generated on each iteration if desired.

**Task 1.B**

This next task involved a burglary problem which was provided in the form of a Bayesian network which features a series of events which could potentially occur in the system. The system in question features a total of five variables: ‘*B*’ – there is a burglary, ‘*E*’ – there is an earthquake, ‘*A*’ – the alarm went off, ‘*J*’ – John called and ‘*M*’ – Mary called. However, the conditional probability table for each variable is unknown and therefore the direct probability for each possible value within the domain size of each variable is unknown.

As a result, the first objective to be handled is acquiring an appropriate conditional probability table which can be argued to accurately describe the probability distributions involved in each variable for all possible cases. To accomplish this, a technique called parameter learning can be used for the purpose of using a sample input dataset to learn about the data distributions within the Bayesian network for each variable, which in turn can effectively be used as the probability values contained within the conditional probability tables.

Specifically, this was achieved by employing the maximum likelihood estimation algorithm, this system contains alternate versions of the maximum likelihood estimation algorithm depending on how many variables are being considered by the algorithm. The function for calculating the maximum likelihood estimation for one variable operates by iterating through the all of the data observations contained within the input dataset and counting the total amount of instances that each possible unique value appears in each variable. For example, the system that was developed features a variable called ‘*B*’ or ‘*Burglary*’, which in turn contains two possible values: True or False. At the outcome of this, this count value would then be input into the numerator of the maximum likelihood estimation formula, thus the estimated probability distribution value would be calculated by total count of a particular variable value with an addition of one divided by the number of observations within the dataset plus the domain size of the variable, the size of which is indicative of the total number of unique values that are contained in a variable. The formula for this version of the maximum likelihood estimation is defined below:

Similarly, this maximum likelihood function can be applied for instances where there may be more than one variable involved in the Bayesian network, such as ‘’. In this case, all instances where the values of variables ‘*x*’ and ‘*y*’ equal to true would be counted together and input into the numerator of the formula with an addition of one. In contrast however, the total count of these values is then divided by the total instances a value of the evidence variable appears plus the domain size of the query variable. The formula for the two-variable alteration of the maximum likelihood function can be observed below:

Furthermore, this algorithm was also adapted to consider cases where there are three variables involved, for example ‘’. This version of the maximum likelihood estimation algorithm operates similarly to the case where two variables might be considered by the algorithm, where instead the third new variable is simply added as an additional evidence variable in the numerator and denominator of the formula. As a result, only instances where variables ‘*x*’, ‘*y*’ and ‘*z*’ are equal to a value within the variable are counted, for example in this case all instances where variables ‘*x*’, ‘*y*’ and ‘*z*’ equal to true would be counted by the function. This is then divided by the total count of instances where only the evidence variables, ‘*y*’ and ‘*z*’, are equal to the same conditional value plus the domain size of the query variable. This can be represented in the three-variable formula below:

By utilising these three versions of the maximum likelihood estimation algorithm, it is subsequently possible to perform parameter estimation and estimate possibility distribution values for all the variables in the conditional probability table based on the input dataset in accordance to the Bayesian network structure. With this, it is possible to estimate the probability values for all of the variables that will be considered as parameters for this particular Bayesian network which can subsequently be input into the next phase of the system architecture: the inference algorithm.

The inference algorithm stage of the system involves the system capacity for deriving new uncertain probabilities based on the known probabilities, Russel and Norvig (2003) surmise that the task for a probabilistic inference system is to compute the posterior probability distributions for a set of query variables given an observed event describing the assignment of values for a set of evidence variables. However, there are multiple methodologies for accomplishing the inference stage of the system, for example one approach would be to perform an exact inference methodology such as inference by enumeration.

Conversely, the methodology selected to be implemented in the system being developed was the methodology entitled ‘rejection sampling’. This operates by first generating a sample for a possible prior probability of the variable, which is calculated by randomly selecting a number and comparing it to the probabilistic values that were previously generated for each variable involved in this Bayesian network through parameter learning. For example, in the case of the system being developed, the prior sample function should return a randomly selected value that could exist with the domain size of the ‘*Mary called*’ variable, such as ‘*+m*’ or ‘*-m*’. The outcome of this is then sent to the main rejection sampling function, whereby the sample generated is compared and checked to see if it is consistent with the conditional probability that is being queried by the user. In the case of the variable , the function would first compare the evidence variables ‘*+m*’ and ‘*+j*’ and check if they are consistent with the prior sample that was randomly generated. If it is found that the evidence of the sample generated is consistent with the conditional probability in question, then the rejection sampling algorithm counts the value contained within the query variable with respect to each possible value within the domain size of the variable. On the other hand, if the evidence variables within the sample are not consistent with the evidence variables contained within the conditional probability being calculated, then the sample is rejected. This algorithm continues to recursively iterate with different random prior samples being generated in each iteration until the number of samples generated reach a total sample limit specified pre-set integer value of ‘*N*’, thus a total count for the number of cases where the conditional probabilities and are true should be found.

The final objective of the inference phase of the system is to normalise the conditional probability value based on the total counts observed in all cases of the query variable. A normalised probability refers to ensuring that the estimated probability values for all cases of a variable will sum to ‘1.0’, which otherwise implies that the final calculated probability distribution is an accurate representation for the conditional probability value. The normalised probability is calculated by one divided by the sum of the counts for all of the unique values of the conditional variable multiplied by the individual count value for the conditional probability being estimated. This is represented in the formula below:

Overall, the rejection sampling methodology provides a means for calculating a consistent estimate of the true probability for an unknown conditional probability efficiently. In comparison to alternative methodologies, the rejection sampling methodology can be argued to be faster depending on the accuracy of the resulting conditional probability that is desired. For example, as discussed by Russel and Norvig (2003), a more accurate probability distribution which is justifiably closer to the ‘true’ probability distribution of the unknown conditional variable will require a larger number of samples to be generated which in turn will require more time to compute. On the other hand, if the system can be designed such that the accuracy is permitted to deviate an amount away from the ‘true’ probability distribution, then a smaller number of samples can be generated which in turn will provide a much faster solution.

-Compare with enumeration and likelihood estimation and justify

**Task 2**

A Markov model can be considered as a stochastic, statistical model which can be utilised for modelling randomly changing systems. For example, the Markov Chain model models the system state with a random variable which is assumed to change as time passes, where it is assumed that the probability distribution for the system only adheres to a dependency on the probability distribution observed in the previous state. However, the system presented with this task merely provides the emission distribution, transition distribution and the initial starting state probability which is assumed to be equiprobable. This in turn presents the implication that while this system has observations related to the system state, the state sequence and some of the system state sequence is hidden, thus the Markov chain model can be argued to not be an appropriate methodology for this particular system. In contrast, a Hidden Markov Model operates similar to a Markov Chain model but for a system where the observations are partially observable and contain hidden states. The Hidden Markov Model operates under the belief that each state in the system has a probability distribution relative to the possible output values, and therefore the sequence of output values can be used to determine more information about the system state sequence.

To embody this concept into the design of the solution that was being developed, the system required more information about the emission probability distribution the user could desire at each new state within the sequence of states of the system being modelled. This operates under a recursive loop which constantly requests for new user input during each iterations, whereby the user is prompted to input an appropriate sequence of symbols from the vocabulary defined within the system, these being either: “Warm”, “Cold”, “Hot”, “Freezing”. The input received from the user will refer to a series of possible emission probability distributions that are known and pre-set in the system.

The emission probability distributions describe the raw probabilities for each emission symbol being output. In contrast, the transition probability distributions describe the raw probabilities of what the next state could be given the values that reside in the current state. However, as previously discussed, this system features hidden states which describe the operational status of the heater, these being either “ON” or “OFF”. Imperatively, the Hidden Markov model can be used to estimate the values of the hidden states through time as per sequence specified by the user, based on the output probability distribution of the previous states.

The Hidden Markov model fundamentally operates under two assumptions in regards to the data involved with the model, the first assumption being the First-Order Markov Chain assumption. This being that the next step in the sequence of states depends on the current state only and none of the previous states. As a result, it can be assumed that the transition to the next state will only depend on the state the system is currently in.

Sensor Markov Assumption

**Conclusion**

**Reference List**

Russel, S.J. and Norvig, P. (2003) *Artificial Intellgience: A Modern Approach,* 2nd edition. New Jersey: Pearson Education Incorporated.