Bayes Nets Assignment Report

Code Analysis-

Function Summaries –

*create\_helper\_variables* – This function reads two JSON files: variables.json and tables.json and creates a named tuple called Helpers to store the output. The Tuple includes a list of Variable objects (imported from Variable) that represent the nodes in the network, a list of Prob objects (imported from probFactors )  for the conditional probability tables (CPTs), and dictionaries that map variable names to their corresponding objects and value mappings. The function first loads the variable definitions, creates Variable objects, and sets up the mappings for easy access. It then loads the CPTs, linking them to their respective variables and their parent nodes. This functions output is further utilized in creating the belief network.

*perform\_exact\_inference* - This function is used for performing exact inference in a Bayesian Network using variable elimination. The function takes in a model (in our case a belief network), the query Variables, evidence variables and variable order of elimination. The function calls VE class from probVE and creates a VE class with the Belief Network. The function calls on the method query that computes the conditional probability by projecting observed variables onto the factors of a graphical model and eliminating unobserved variables in a specified order. It calculates unnormalized probabilities by multiplying the remaining factors and normalizes the results to ensure they sum to 1. Finally, it returns a dictionary of the normalized probabilities for the queried variable.

perform\_approximate\_inference – This function takes a model (belief network in our case) and applies rejection sampling (using the RejectionSampling class). The query method estimates the conditional probability by sampling from the joint distribution of a graphical model. It samples values for the queried variable while respecting observed values, counting how many times each value of queried variable is accepted. Finally, it returns a probability distribution based on the accepted samples, along with raw counts.

compute\_mse – This function helps compute the MSE between predicted and actual values after removing a specific key from the predictions.

The code uses the function ‘create\_helper\_variables’ to create the nodes and Conditional Probability table. These helpers are utilized the crete a belief network using the class BeliefNetwork from probGraphicalModels. The code then performs an exact inference P(Disease | CO2Report = 1, XrayReport = 0, Age = 0). We initially use ascending order based on variable names as the elimination model but then use, we can use min-fill ordering as a better ordering, here we eliminate those variables first whose removal would require the least introduction of edges between the remaining neighbors. The average time taken over 10 runs with both ordering approaches is stored in part1.csv .In Part 2 we use the function perform\_approximate\_inference to use rejection sampling with different number of samples n = 10, n = 100, n = 1000, the average time taken by these three approaches is stored in part21.csv along with this we calculate the mean square error for each of the three stored in part22.csv.

Result Analysis-

Part 1 –

* We noticed that –

Average time taken (alphabetical order): 0.180226 seconds

Average time taken (min-fill order): 0.071373 seconds

* We can see that the min fill ordering finished in less than half the time as it is letting us eliminate the variable’s that create the least amount of extra edges upon elimination. Considering that the order we have was derived from the graph it highlights the usage of the graphical representation of the belief network

Part 2 –

* The approximate inferences using both 10 and 100 samples are borderline meaningless – I got a uniform distribution when I used 10 samples, giving us no actual data. Similarly for a 100 samples multiple values in the distribution were 0. However the distribution achieved over 14445 samples was a lot closer the distribution achieved through exact inference
* The MSE did not change by a lot for the 10 sample and 100 sample (the MSE was higher for the 100 sample approximate inference) We believe this is the case as the difference in the order of magnitude is not too high. We achieved a more reasonable MSE for the approximate inference with 18445 samples.
* The average time take by the program was pretty linear with respect to the number of samples taken indicating that a major part of the program (aka major time taken by the program) is taken by the sampling process