Vishwanath Dattatreya Doddamani A0286188L e1237250@u.nus.edu

CS5340 ASSIGNMENT 2 – Part 1 REPORT

Junction Tree Algorithm

Details of function implementation:

➔ **Basic Factor Operations**

1. **factor_product()** The code for this function is same as assignment 1 where two factors are multiplied.
2. **factor_marginalize()** The code for this function is also same as assignment 1 where a factor is marginalized based on the input variables provided to marginalize.
3. **`factor_evidence()`** For each evidence, only the slice of the factor table is retained while deleting the observed variable in the factor.

➔ **`Junction Tree Algorithm`**

`In jt_construction.py,`

a. **`_get_jt_clique_and_edges()`** This function first computes the cliques using networkx library. Then edges are added along with weight. Weight being the cardinality of intersecting set between the cliques. Then maximum spanning tree is computed using networkx library. Cliques and edges are returned accordingly.
b. **`_get_clique_factors()`** The code for this function computes the factors of cliques by going through the input list of factors. If the set factor variables is a subset of set of clique variables, then that factor is multiplied. Once the factor is used, it is deleted from the factor list so that it is not multiplied again as it causes inconsistency. The final product is returned for each clique.

`In main.py,`

a. **`_get_clique_potentials()`** Given cliques, clique edges and clique factors, it computes clique potentials using sum product algorithm similar to assignment 1. Parent of nodes are identified, then the children of all nodes (cliques) are identified. Subsequently, from children to node the messages are collected and then distributed to all the children.
b. **`_get_node_marginal_probabilities()`** Given clique potentials, this function marginalizes all the variables of the clique except the node, then normalizes. Finally, returns the node marginal probabilities of all the nodes.

Challenges Faced:

1. While creating junction tree, connecting the non-connected subgraph needed to be taken care as in **_update_mrf_w_evidence()** the edges are removed corresponding the evidence. A lot of debugging had to be done to identify this seemingly obvious issue.
2. Figuring that handling corner cases is essential, took a toll. However, once identified it was implemented without much difficulty.

Conclusion:

Part 1 of Assignment 2 makes one appreciate the complicated reality of real world application where a problem has to be reconstructed in other format in order reach a solution (in this case graph to junction tree).