Assignment 4

Due: 10. December 2021, 10:15 CET

General information: This assignment should be completed in groups of **three to four people**. Submit your assignment via the corresponding LernraumPlus activity. You can only upload one file, therefore make sure to place all your files required to run your solution (.py, .ipynb, .pdf etc) into **one archive** and just upload that archive.

Python exercise information: You are only allowed to use the external libraries that we provide or explicitly mention on the assignment sheet (or that are already within the provided skeleton files). Your solutions' functionality will all be tested against automated tests before being reviewed further. **You need to ensure** that your solution can directly be **imported as a Python3 module** for automatic tests and at least contains the name of the skeleton file. The easiest way to achieve this, is to directly develop your solution in the assignment4.py file and submit that. The LernraumPlus contains information about the test environment. You will also find an example test-file alongside the skeleton file that you can use to ensure that your solution works in the evaluation environment.

Exercise information: In this assignment you will be implementing your first inference methods by using the previously developed Bayesian Network classes as well as a factor class that I provide. You will again find the provided reference implementations (for the graph and the factor class) in the *ccbase* package in the *assignment4.zip* alongside the *assignment4.py* skeleton file, the example test file and the generated documentation. Pay attention to the provided docstrings as they further explain what is expected from the different functions. As with the last assignments, feel free to use your own graph class. You are free to use **numpy** as a third party library.

Exercise 1: (5 Points)

The order in which variables are being eliminated has a significant effect on the performance (runtime) of variable elimination algorithms. Finding an optimal elimination order is NP-hard, but different heuristics exist that can yield mostly good orders, such as the *MinFillOrder* heuristic [1]:

When removing a variable, we need to update the interaction graph by connecting the variable's neighbors that were not connected before. In order to keep factors small, we want to eliminate the variable that causes the least amounts of edges to be added between neighbors when removing that variable.

Task 1: (3 Points) Implement the *get_elimination_order(bn)* function, which computes the elimination order according to the *MinFillOrder* heuristic.

Task 2: (2 Points) Name and explain another reasonable heuristic for determining the elimination order. The explanation should include the differences to the *MinFillOrder* heuristic.

Exercise 2: (7 Points)
Implement exact inference in Bayesian networks following the *variable elimination* algorithm. Make use of the provided factor class.

To this end you should implement:

Task 1: (2 Points) The *initialize_factors(bn, evidence)* function, which creates a set of factors from the given Bayesian network and applies the given evidence to them.

Task 2: (3 Points) The *sum_product_elim_var(factors, variable)* function, which takes a set of factors and removes the given variable from this set of factors.

Task 3: (2 Points) The *calculate_probabilities(bn, variables, evidence=None)* function, that actually computes the (joint) probability of the given variables, considering the provided evidence.

Your algorithm should handle the following cases:

- query and calculate prior marginals for every variable in the network
- query and calculate posteriors given evidence observed for one or more other nodes (evidence can be empty)

Exercise 3: (8 Points)
Apart from probability queries there are also value queries, i.e. queries for the instantiations of all or a subset of variables that maximize the joint probability.

Task 1: (2 Points) The *maximize_out(factor, variable)* function, which removes the given variable from the factor via maximization.

Task 2: (2 Points) The *max_product_elim_var(factors, variable)* function, which takes a set of factors and removes the given variable from this set of factors by maximizing the resulting factor with respect to that variable.

Task 3: (2 Points) The *traceback(factors, order)* function, which will return the instantiations for all variables that maximize the joint probability based on the factors from which variables were removed in *max_product_elim_var* and the given elimination order.

Task 4: (2 Points) The *calculate_MAP(bn, evidence=None)* function, that actually computes probability of the *most probable explanation* (MPE) as well as the instantiations of all variables that make up this MPE.

Hint: Maximizing out a variable from a factor is defined as [1]:

$$(max_x f)(\mathbf{y}) = max_x f(x, \mathbf{y})$$

Hint 2: You can check your result for the MAP function by comparing the probabilities of the joint distribution with your MPE probability.

General hint: Keep in mind how factors change with evidence and that valid probability distributions need to sum up to 1. Also keep in mind that you can still use a function from a previous exercise/task, even if you did not solve it yourself. As long as you use the functions correctly you will still get full points for the subsequent exercises.

References

[1] Darwiche, Adnan. *Modeling and reasoning with Bayesian networks.* Cambridge University Press, 2009.