Gene Heredity: Probabilistic Inference Assignment

September 20, 2023

Objective

The objective of this assignment is to explore probabilistic inference in the context of gene heredity. You will be provided with a Python code template that implements a simplified probabilistic model for heredity genetic traits. Extended explanations about the reasoning of the exercise can be found at this link. Your task is to complete the code by implementing the missing parts and then answer a series of questions related to probability and Bayesian networks.

Instructions

Part 1: Completing the Code

1. Download the provided Python code template and save it as `heredity.py`.
2. Study the code template carefully. It contains functions for loading data, computing joint probabilities, updating probabilities, and normalizing probabilities for a given Bayesian network.
3. Complete the missing parts of the code. You will need to implement the following functions:
    - `joint_probability`: Calculate the joint probability for a given set of individuals with specific gene and trait characteristics.
    - `update`: Update the probabilities dictionary with the computed joint probability.
    - `normalize`: Normalize the probabilities to ensure they sum to 1.
    - `main`: Calculate all relevant probabilities by using the functions you have completed.

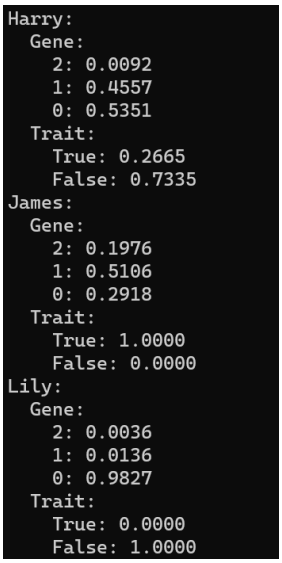

Figure 1: Example of output you should get in the terminal by inputting family0.csv.

Detailed instructions

The `joint_probability` function

The `joint_probability` function calculates the joint probability for a given set of individuals with specific gene and trait characteristics. It is crucial to correctly compute this probability to make accurate Bayesian network inferences. Here's how you should implement it:

- For each individual in the set of individuals passed to the function:
- Determine how many gene copies (0, 1, or 2) they have based on their parents' gene copies and mutation probabilities. You'll need to consider all possible combinations of gene inheritance.
- Calculate the probability of having or not having the trait based on the number of gene copies and the trait probabilities from the `PROBS` dictionary.
- Multiply these probabilities together to compute the joint probability for that individual.

The `update` function

The `update` function updates the probabilities dictionary with the computed joint probability for a given set of individuals. Here's how you should implement it:

- For each individual in the set of individuals passed to the function:
- Determine the number of gene copies (0, 1, or 2) they have based on their parents' gene copies and mutation probabilities. You'll need to consider all possible combinations of gene inheritance.
- Update the appropriate entry in the probabilities dictionary for that individual, incrementing the count of individuals with that specific gene and trait combination.
- Repeat this process for all individuals in the set, and ensure that the probabilities dictionary is correctly updated with the joint probabilities.

The `normalize` function

The `normalize` function is responsible for ensuring that the probabilities in the probabilities dictionary sum to 1, which is a fundamental property of probability distributions. Here's how you should implement it:

- For each individual in the probabilities dictionary:
- Calculate the sum of probabilities across all possible gene and trait combinations. This involves summing up the probabilities in both the gene and trait dictionaries.
- For the gene probabilities, there are three possibilities (0, 1, or 2 gene copies), and for the trait probabilities, there are two possibilities (True or False trait).
- Divide each individual probability by the total sum to normalize it. This ensures that the probabilities for each gene and trait combination sum to 1.
- Update the probabilities dictionary with the normalized probabilities for that individual.

Part 2: Running the Code

1. Open a terminal or command prompt.
2. Navigate to the directory where you saved the `heredity.py` file.
3. Run the code by executing the following command:

   ```
   python heredity.py data.csv
   ```

   Replace `data.csv` with the path to a CSV file containing genetic and trait data for a set of individuals (`family0.csv`, `family1.csv`, `family2.csv`). The CSV file should have the following columns: `name`, `mother`, `father`, and `trait`. The `trait` column should contain values 0, 1, or be blank if the trait is unknown.

Part 3: Answering Questions

1. The problem presented in this assignment can be modeled as a Bayesian Network (BN). List the variables of such network, and indicate, for each variable, its domain and its parents (if there are any).

2. Explain the concept of conditional independence, and explain how it relates to Bayesian Networks.

3. Can you explain briefly the concept of inference? How is inference carried out in the code?

4. Are BNs an example of symbolic or sub-symbolic AI? Motivate your answer.

5. Consider the assignment. Would you consider the agent you created knowledge-based? If so, can you identify prior knowledge in the code?

6. During the course, we have seen how many problems in AI can be modeled as search problems. Describe the elements of a search problem and make three examples of instances that can be described as search problems - these can either be examples that we have seen in the lectures or other examples.

Submission

Submit the following for your assignment:

1. The completed `heredity.py` code file.

2. A document containing your answers to the questions in Part 3.

Upload your final files on Moodle. Feel free to carry out the exercise in pairs, and indicate your names in the name of the file.