Assignment sheet 5

Uncertainty

Regard the situation that a medical doctor has to make a diagnosis on the symptoms and lab values that a patient shows. Explain why this is a case of reasoning under uncertainty.

This is a case of reasoning under uncertainty because of uncertain knowledge. A symptom may have many different reasons, for example if the patient has a fever, a quick google search show that these diseases may give a fever:

- A viral infection
- A bacterial infection
- Heat exhaustion
- Certain inflammatory conditions such as rheumatoid arthritis - inflammation of the lining of your joints (synovium)
- A cancerous (malignant) tumor
- Some medications, such as antibiotics and drugs used to treat high blood pressure or seizures
- Some immunizations, such as the diphtheria, tetanus and acellular pertussis (DTaP), pneumococcal or COVID vaccine

Source: https://www.mayoclinic.org/diseases-conditions/fever/symptoms-causes/syc-20352759 , read 22.11.2022

Another reason for uncertainty is that a disease doesn't necessarily produce the same symptoms for each patient. The seasonal flu, for example, may cause the patient to have these symptoms:

- fever* or feeling feverish/chills
- cough
- sore throat
- runny or stuffy nose
- muscle or body aches
- headaches
- fatigue (tiredness)

- some people may have vomiting and diarrhea, though this is more common in children than adults.

Source: https://www.cdc.gov/flu/symptoms/symptoms.htm , read 22.11.2022.

These symptoms happen at a varying frequency, therefore, if the patient show symptoms of coughing and headaches, there's a chance it has the flu. The patient doesn't necessarily need to have all the symptoms to be diagnosed with the flu.

Why is uncertainty modelling important when creating AI systems?

When creating AI systems, we often want to apply it in the real world. In this environment, only a handful of things is certain at best, most things are uncertain. For example, if we train an AI to play chess it will never know for sure which move is the best. This is because the search space is too big. It can't simulate all the possible states that may occur for all the possible move, it would take an eternity. It must take some shortcuts, and to make these shortcuts, uncertainty modelling is important.

Explain the main principles of Mycin's certainty factor

Mycin's certainty factor is based of if-then rules, which are pre-programmed. Each if-then rule is associated with an uncertainty. For example, one rule may say "if patient is coughing, then 90% certain of flu", and another one may say "if patient has fever, then 75% certain of flu". In this case the certainty factor will be calculated by using the formula $X + Y - XY$, where X and Y are certainty factors. This results in a certainty factor of 0.975. The certainty factors can have negative values, to combine certainty factors, other than two positives, it uses either $X + Y + XY, X, Y < 0$ or $\frac{X+Y}{1-\min(|X|,|Y|)}$, othertwise. Source: https://en.wikipedia.org/wiki/Mycin, read 22.11.2022

State and explain Bayes' theorem. Explain all the 4 probability terms which appear in the theorem

Bayes' theorem is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

$P(A|B)$ – The probability of A, given B.

$P(B|A)$ – The probability of B, given A.

$P(A)$ – The probability of A.

$P(B)$ – The probability of B.

It is an equation that helps to update the beliefs of how likely A is, given the evidence B, and prior belief A. The equation updates based on the probability of the evidence, B, given A ($P(B|A)$), the probability of A is (prior belief $P(A)$), and the probability of the evidence ($P(B)$).

Why is this theorem important for reasoning under uncertainty?

This theorem is important very few elements in the real world is 100% certain, for instance a cancer tests. In the event of a positive cancer test, you would be quite certain that you have cancer. But according to Bayes' theorem you still have a quite low chance, even if the test is very accurate.

What kind of information about the diseases is missing to make a solid decision between two 'competing' disease hypotheses disease23 and disease51 if both rules fire for a given patient?

You would need to know the probability of each disease given the blood substance level.

Explain what kind of statistical information (in terms of probabilities Prob(event) and/or Prob(eventA | eventB)) would be needed to apply Bayes' rule to this decision problem.

$$P(disease23),$$
$$P(disease51),$$
$$P(blood\ substance08\ level|disease23),$$
$$P(blood\ substance09\ level|disease51),$$
$$P(blood\ substance08\ level),$$
$$P(blood\ substance09\ level).$$

Why are the above example rules not perfectly suitable to perform statistically optimal reasoning? What is missing, or only coarsely represented, in these rules? Hint: the expected solution has to do with the fixed numeric threshold values in these rules.

I think that it is because threshold values in the rules. Whether the blood substance08 level is 0.008 or 0.016 likely changes the probability of having disease 23.

Imagine you have to design a continuously learning expert system for medical diagnosis (like the above application case). In which way could a system, starting from an initial reasonable configuration, learn to improve its decisions? Imagine that during this training phase, there is an experienced medical doctor available that knows the correct diagnosis in each case. Hint: the solution has to do with probabilities...

The system could improve its decisions by adjusting the probability of having the disease given the blood substance level, and the probability of having the disease. For example, if the system knows that 1 out of 10,000 people has a disease, and then tests 10,000 more people where 2 people have the disease, the probability can be updated to 3 of 20,000. The system can also to the same with the probability of having the disease given the substance level.

How can an agent be rational when knowledge is uncertain? Explain the Maximum expected utility principle.

An agent is rational if it does the action which is expected to give the most utility. Consider two bets, in the first, you bet 10$ on a coin to land on heads and can win 20$, in the second bet you also bet 10$, and you win 15$ if the coin land on heads. You don't know if you are going to make or lose money. But the expected return on the first bet is 10$, whereas the second on is 7$ and 50 cents. To maximize your return, you should opt for the first bet, since the expected return, i.e., utility is higher. Therefore, it's the logical thing to do, and an agent needs to be aware of this principle to act rational.

Naïve Bayes

Using the data in golf.csv, compute the probability tables of going golfing conditioned on each criterion. P (Golfing|Outlook), P (¬Golfing|Outlook), P(Golfing|Temperature), P(¬Golfing|Temperature), P(Golfing|Humidity), etc...

The output of the program, as well as the code is given in the appendix in the end of the assignment.

Using the conditional probability tables from the previous task, decide whether or not to go golfing the following days using the naive Bayes classifier

The output of the program, as well as the code is given in the appendix in the end of the assignment.

What are the main drawbacks of using the Naive Bayes classifier?

One of the main drawbacks is that the Naïve Bayes Classifier assumes independent relations. For example, in the data in golf.csv, the classifier assumes that "Sunny" outlook isn't related to "High" or "Normal" temperatures.

Another drawback is that all the criteria are equally important. Again, from the example data, the outlook, temperature, humidity, and wind are equally important. One could think that heavy rainfall would make golfing seem less attractive than when the temperature is cold.

Bayesian Network

Explain what a Bayesian network is, and why it is useful

A Bayesian network is a probabilistic graphical model. It represents variables and their conditional dependence, causation, by representing their dependence as an edge in a directed acyclic graph (DAG).

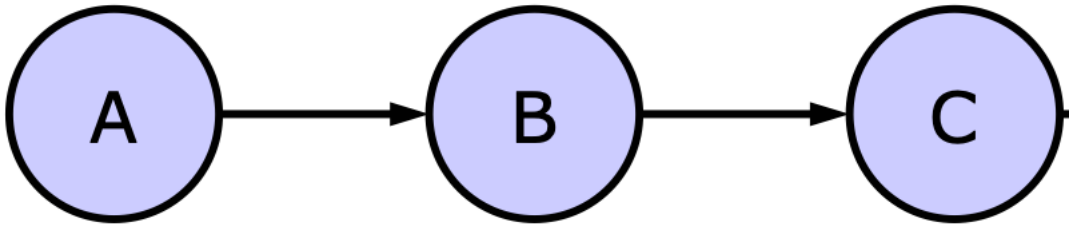
It is useful for predicting the likelihood of the contributing causes to an event. For example, which diseases are likely to cause the given symptom in a patient.

Sketch a simple Bayesian network with five nodes and explain the meaning associated with these nodes. Choose a different, but reasonable problem than in the lecture slides. Provide plausible conditional probability tables (the numeric values can be made up).

See the appendix.

How can we use the Bayesian network for inference? Explain at least one inference method

A Bayesian network can be used for inference by query the Bayesian network. For example, what is the probability distribution over X given the data d ($P(X|d)$)? To answer questions like the example, one could use the enumeration method in the Bayesian network. This method involves the joint distribution of the variables and is compactly described as $P(X) = \sum_{ABC} P(A, B, C, X)$. In a simple Bayesian network of three nodes, looking like this:



The formula is written out like this:

$$P(C) = \sum_{AB} P(C|B)P(B|A)P(A).$$

For more variables or many outcomes per variable, this leads to an enormous amount of work. One must cut down the number of calculations. The first thing to do is writing it like this:

$$P(C) = \sum_{B} P(C|B) \sum_{A} P(B|A)P(A).$$

To further reduce the amount of work done, use matrices:

$$f_1(B) = \begin{bmatrix} P(B_1|A_1)P(A_1) & \cdots & P(B_1|A_n)P(A_n) \\ \vdots & \ddots & \vdots \\ P(B_m|A_1)P(A_1) & \cdots & P(B_m|A_n)P(A_n) \end{bmatrix} = \begin{bmatrix} \sum_A P(B_1|A)P(A) \\ \vdots \\ \sum_A P(B_m|A)P(A) \end{bmatrix}$$

The we can rewrite the equation to:

$$P(C) = \sum_b P(C|B)f_1(B)$$

Appendix

main.py

```python
from naive_bayes import NaiveBayes


def read_data(filepath: str) -> tuple[list[str], list[list[str]]]:
    """ Read data from file. Return criterias and data."""
    data: list[list[str]] = []
    with open(filepath, "r", encoding="UTF-8") as file:
        for line in file:
            data.append(line.strip().split(","))
    return data[0], data[1:]


def main():
    # Read data
    criterias, data = read_data("assignment5/code/data/golf.csv")

    model = NaiveBayes(criterias)

    # Train model
    model.train(data)

    print("Task 5.11".center(100, "-"))

    probabilities = model.probabilites
    for criteria in criterias[:-1]:
        print(f"{criteria}:")
        for value in probabilities[criteria]:
            print(
                f"\tP(Golfing|{value}): {probabilities[criteria][value]['Yes']}")
            print(
                f"\tP(¬Golfing|{value}): {probabilities[criteria][value]['No']}")
    print("Golfing:")
    print(f"\tP(Golfing): {probabilities[criterias[-1]]['Yes']}")
    print(f"\tP(¬Golfing): {probabilities[criterias[-1]]['No']}")

    print("Task 5.12".center(100, "-"))
    day1 = {"Outlook": "Sunny", "Temperature": "Cool",
            "Humidity": "High", "Windy": "True"}
    day2 = {"Outlook": "Overcast", "Temperature": "Mild",
            "Humidity": "Normal", "Windy": "False"}
    day3 = {"Outlook": "Rainy", "Temperature": "Mild",
            "Humidity": "Normal", "Windy": "False"}

    print(f"day1={day1.values()} = {model.predict(**day1)}")
    print(f"day2={day2.values()} = {model.predict(**day2)}")
    print(f"day3={day3.values()} = {model.predict(**day3)}")


if __name__ == "__main__":
    main()
```

naive_bayes.py

```python
from collections import defaultdict
from dataclasses import dataclass
from fractions import Fraction


@dataclass
class NaiveBayes:
    """
    The last criteria in the list of criterias given, is the outcome, so we don't need to calculate the
probabilities for it. The last criteria need to be on the form "Yes" or "No"."""
    criterias: list[str]
    probabilites: dict[str, dict[str, dict[str, Fraction]]]

    def __init__(self, criterias: list[str]) -> None:
        self.criterias = criterias
        probabilites = {}
        for criteria in criterias[:-1]:
            probabilites[criteria] = defaultdict(lambda: {"Yes": 0, "No": 0})
        probabilites[criterias[-1]] = {"Yes": 0, "No": 0}
        self.probabilites = probabilites

    def train(self, data: list[list[str]]) -> None:
        """
        Train the model with the given data, i.e. calculate the probabilities for each criteria.
        """
        # Find the number of times each criteria has a specific value
        for line in data:
            assert len(line) == len(self.criterias), "Invalid data"
            self.probabilites[self.criterias[-1]][line[-1]] += 1
            for criteria, value in zip(self.criterias[:-1], line[:-1]):
                self.probabilites[criteria][value][line[-1]] += 1

        # Turn the number of times into probabilities
        for criteria in self.criterias[:-1]:
            for value in self.probabilites[criteria]:
                self.probabilites[criteria][value]["Yes"] = Fraction(
                    self.probabilites[criteria][value]["Yes"], self.probabilites[self.criterias[-1]]["Yes"])
                self.probabilites[criteria][value]["No"] = Fraction(
                    self.probabilites[criteria][value]["No"], self.probabilites[self.criterias[-1]]["No"])
        self.probabilites[self.criterias[-1]]["Yes"] = Fraction(
            self.probabilites[self.criterias[-1]]["Yes"], len(data))
        self.probabilites[self.criterias[-1]]["No"] = Fraction(
            self.probabilites[self.criterias[-1]]["No"], len(data))

    def predict(self, **kwargs) -> str:
        """ Predict the outcome for the given criteria. """
        yes = self.probabilites[self.criterias[-1]]["Yes"]
        no = self.probabilites[self.criterias[-1]]["No"]
        for criteria in self.criterias[:-1]:
            try:
                yes *= self.probabilites[criteria][kwargs[criteria]]["Yes"]
                no *= self.probabilites[criteria][kwargs[criteria]]["No"]
            except KeyError:
                print(f"No value for {criteria} given")
                return "Wrong input"
        print(f"Yes: {yes}, No: {no}")
        return "Yes" if yes > no else "No"
```

Result

```
-----------------------------------------------Task 5.11-----------------------------------------------
Outlook:
        P(Golfing|Rainy): 2/9
        P(¬Golfing|Rainy): 3/5
        P(Golfing|Overcast): 4/9
        P(¬Golfing|Overcast): 0
        P(Golfing|Sunny): 1/3
        P(¬Golfing|Sunny): 2/5
Temperature:
        P(Golfing|Hot): 2/9
        P(¬Golfing|Hot): 2/5
        P(Golfing|Mild): 4/9
        P(¬Golfing|Mild): 2/5
        P(Golfing|Cool): 1/3
        P(¬Golfing|Cool): 1/5
Humidity:
        P(Golfing|High): 1/3
        P(¬Golfing|High): 4/5
        P(Golfing|Normal): 2/3
        P(¬Golfing|Normal): 1/5
Windy:
        P(Golfing|False): 2/3
        P(¬Golfing|False): 2/5
        P(Golfing|True): 1/3
        P(¬Golfing|True): 3/5
Golfing:
        P(Golfing): 9/14
        P(¬Golfing): 5/14
-----------------------------------------------Task 5.12-----------------------------------------------
Yes: 125/341, No: 4092/113467
day1=dict_values(['Sunny', 'Cool', 'High', 'True']) = Yes
Yes: 1, No: 0
day2=dict_values(['Overcast', 'Mild', 'Normal', 'False']) = Yes
Yes: 1000/1243, No: 3729/441229
day3=dict_values(['Rainy', 'Mild', 'Normal', 'False']) = Yes
```

Bayesian Network