Assignment 4 Bayes Net

Due Dec 7 by 12p.m. **Points** 100

Table of Contents

Warning

We know that solutions to this or related problems may exist online. *Do not use these solutions, as this would be plagiarism.* To earn marks on this assignment, you must develop your own solutions.

Also, please consider the following important points:

- *Do NOT add any non-standard imports*. All imports already in the starter code must remain. When in doubt, post a question on Piazza.
- *Do NOT modify any provided code*. Modifying the provided code may cause you to *fail* all the tests.
- Submit your program on MarkUs and run the provided tests *well before* the deadline. The fact that your code runs on your own system but not on MarkUs is *NOT* a legitimate reason for a regrade request.
- The provided tests on MarkUs are an *extremely basic* sanity check. Passing the provided tests only means that your program runs without errors; it does *NOT* mean you will receive full marks on the assignment.

Your Tasks

In this assignment, you will predict the salaries of people who lived in the United States in 1994 by using a variety of attributes. To do so, you will create a Naive Bayes model to represent the relationships between people's salaries and their other attributes such as race, gender, education level, etc. Then, you will implement and use the variable elimination algorithm to calculate probabilities and make salary predictions.

Implementing the Variable Elimination Algorithm

To start, you must implement several functions for the **variable elimination algorithm**. The functions are described briefly below.

- **normalize**: Return a new factor, a normalized version of the input factor. Do not modify the input factor.
- **restrict**: Return a new factor, which restricts the given variable to the given value in the input factor. Do not modify the input factor.
- **sum_out**: Return a new factor, which results from summing out the given variable from the input factor. Do not modify the input factor.
- **multiply**: Return a new factor, which is the product of the list of input factors. Do not modify the input factor list.
- **min_fill_ordering**: This function implements the min fill heuristic, explained below.
- **ve**: This function computes a distribution over the values of the query variable given the values of the evidence variables by using the Variable Elimination Algorithm.

The second last function min_fill_ordering implements **The Min Fill Heuristic**. We will use this heuristic to determine the order to eliminate the hidden variables.

The Min Fill Heuristic says to **eliminate next the variable that creates the factor of the smallest size**. If there is a tie, choose the variable that comes first in the list of factors provided.

For example, consider our complete Holmes network below.

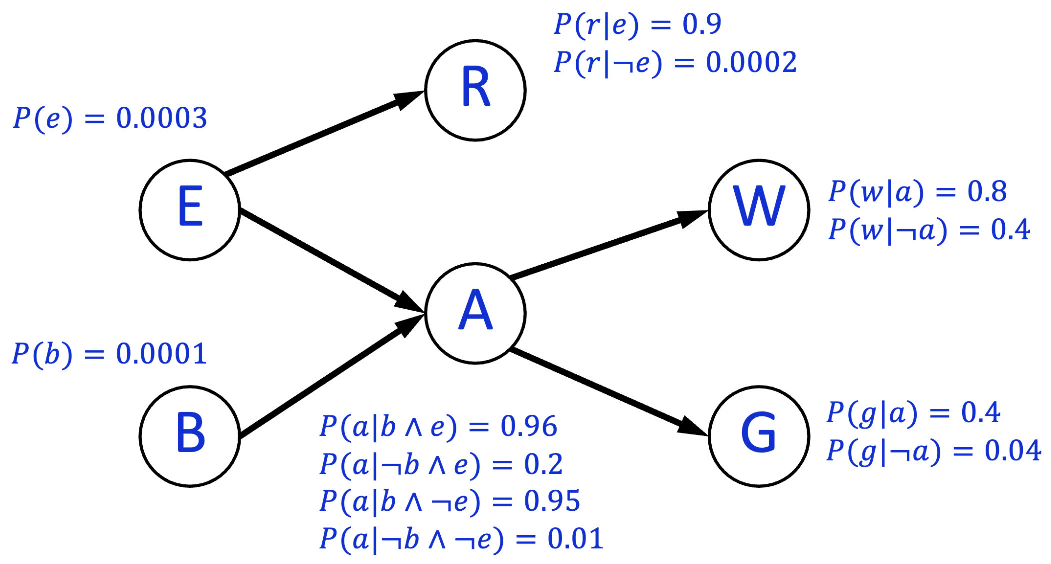


Figure: A Bayesian Network for the Holmes Scenario

Suppose that we are given a list of factors for the variables in this order: P(**E**), P(**B**), P(**A**|B, E), P(**G**|A), and P(**W**|A). Assume that our query variable is **Earthquake**. Among the other variables, which one should we eliminate first based on the Min Fill Heuristic?

- Eliminating B creates a factor of 2 variables (A and E).
- Eliminating A creates a factor of 4 variables (E, B, G and W).
- Eliminating G creates a factor of 1 variable (A).
- Eliminating W creates a factor of 1 variable (A).

In this case, G and W tie for the best variable to be eliminated first since eliminating each variable creates a factor of 1 variable only. Based on our tie-breaking rule, we should choose G since it comes before W in the list of factors provided.

Implementing the Naive Bayes Network

Next, you will construct a Naive Bayes model using the 1994 United States Census Data (https://www.census.gov (http://www.census.gov) /). The training and test data are provided in **adult-train.csv** and **adult-test.csv,** respectively. Each individual in the data set has the following attributes:

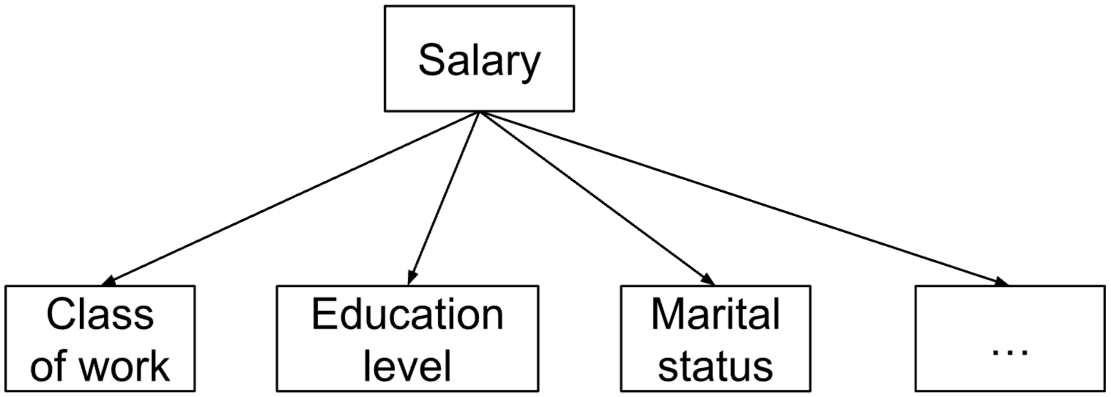
1. Work (e.g. government employee, self-employed, etc).
2. Education (e.g. high school graduate, professional degree, etc)
3. Marital status (e.g. married, single, separated, etc.)
4. Occupation (e.g. military, administrative, manual labour, etc.),
5. Relationship (e.g. wife, husband, unmarried, etc.)
6. Race (e.g. Black, White, Pacific Islander, etc.)
7. Gender (e.g. male, female)
8. Country (e.g. North-American, Asia, etc.)
9. Salary (below $50K per year or over $50K per year)

To create a Naive Bayes model, you will implement the **naive_bayes_model** function in the **naive_bayes_starter.py** file. The naive_bayes_model function starts by reading the data from the given data_file. For convenience, we have defined the categorical variables' domains for you in the starter code.

Take a look at a visualization of the Naive Bayes model below. To start, define one variable for each attribute (e.g. class of work, education level, etc.). Next, you will need to define one factor for each variable. We will use "Salary" as the prior of our model. Every other variable conditions on "Salary."

Please name the factors as follows. If you don't follow these naming conventions, you will fail our tests.

- The name of the Salary factor should be called "Salary" without the quotation marks.
- The name of any other factor should be called "VariableName,Salary" without the quotation marks. For example, the factor for Education should be called "Education,Salary".



A Naive Bayes Network for the Salary Prediction Problem

Finally, populate the conditional probability table in each factor by counting the frequency of values in the training set. Let's consider "Salary" as an example. Since "Salary" has no parent in the Bayesian network, its probabilities do not depend on other variables. We can calculate P(Salary<50K) using the formula below.

Probability(Salary<50K) = (# of individuals with Salary <50K) divided by (total # of individuals)

To give another example, let's consider Work. Since Salary is a parent of Work, we need to calculate the probabilities of Work for different Salary values. For example, we can calculate P(Work=Private | Salary<50K) using the formula below.

Probability(Work=Private | Salary<50K)= (# of individuals with Work = Private and Salary <50K) divided by (# of individuals with Salary<50K)

Now, we are ready to use our Naive Bayes model to predict salaries. For example, we can predict that a salary is over $50K, given some evidence E, if the probability of Salary being greater than $50K, given the evidence E, is greater than 0.5. Formally, this means

P(Salary >= $50K | Evidence) > 0.5

By using this formulation, we can make predictions for all the data points in "adult-test.csv." In general, we never use the same data to build a machine learning model and to perform predictions using the model. Instead, we start by building the model using a data set called the **training** set. Then, we may tweak the parameters of our model using a second, separate data set called the **validation** set. Finally, we will evaluate how well our model performs predictions using a third, separate data set called the **test** set.

Evaluating Model Fairness

There are a lot of ways to think about whether or not the predictions that we make with this model are fair. If you're interested in learning some of the ways people have been thinking about measuring fairness, you might be interested in this **fairness tutorial** from Solon Barocas and Moritz Hardt.

Professor Toniann Pitassi has taught a whole course on this topic; you might be interested in looking at **some of the resources she has assembled (https://www.cs.toronto.edu/~toni/Courses/Fairness/fair.html).**

We will explore a few options to measure "fairness" concerning gender here. Let E be the evidence set containing the variables: Work, Education, Occupation, and Relationship.

We might consider predictions to be **fair** if

- The predictions achieve **'demographic parity'**. That is, the distribution of our guesses is the same for men and women.

P(Predicted Salary >$50K | Gender=Female) = P(Predicted Salary >$50K | Gender=Male) = P(Predicted Salary >$50K)

- The predictions are well **'separated'** from gender, which means that given some evidence, our prediction, given we know or don't know the gender, is the same.

P(Predicted Salary >$50K | Evidence, Gender) = P(Predicted Salary >$50K | Evidence)

- The predictions are **'sufficient'**, which means that gender tells us nothing more than our guess about an individual's actual salary.

P(Actual Salary >$50K | Predicted Salary >$50K, Gender) = P(Actual Salary >$50K | Predicted Salary >$50K)

Answer the following six questions by completing the **explore** function in the **naive_bayes_starter.py** file.

1. What percentage of the **women** in the test data set does our model predict having a salary >= $50K?
2. What percentage of the **men** in the test data set does our model predict having a salary >= $50K?
3. What percentage of the **women** in the test data set satisfies the condition: P(Salary=">=$50K" | Evidence) > P(Salary=">=$50K" | Evidence, Gender)
4. What percentage of the **men** in the test data set satisfies the condition: P(Salary=">=$50K"|E) > P(Salary=">=$50K"|E, Gender)?
5. What percentage of the **women** in the test data set with a predicted salary over $50K (P(Salary=">=$50K"|E) > 0.5) have an actual salary over $50K?
6. What percentage of the **men** in the test data set with a predicted salary over $50K (P(Salary=">=$50K"|E) > 0.5) have an actual salary over $50K?

Submit a file named **bnet_reflection.txt** to MarkUs. In this file, respond to the two questions below.

1. Explain what the calculations above might or might not tell you about the "fairness" of your Naive Bayesian network. Explain in **50-100 words**.

2. Would you be willing to use your model to recommend starting salaries for employees at a firm? Why or why not? Explain in **50-100 words**.

Starter Code

To support you in completing this assignment, we have provided several files to get you started.

- **adult-train.csv**: A sample training set for learning the Naive Bayes model.
- **adult-test.csv**: A sample test set for making predictions using the Naive Bayes model.
- **bnetbase.py:** This file contains the class definitions for Variable, Factor, and BayesNet.
- **naive_bayes_starter.py**: This file contains functions to implement the variable elimination algorithm, create the Naive Bayes Network, and calculate the six probabilities.
- **bnet_reflection.txt**: This file contains two reflection questions that you need to complete.

Grading Scheme

We have set up three projects on MarkUs: A4-Week1, A4-Week2 and A4. The first two projects (A4-Week1 and A4-Week2) contain the extra unit tests for the optional early deadlines. They are not worth any marks.

To earn marks for this assignment, you must submit **naive_bayes.py** and **bnet_reflection.txt** to the A4 *project on MarkUs*.

The table below contains detailed information regarding the available tests and the grading scheme.

MarkUs Test Availability and Grading Scheme

Category	**Test Descriptions**	**A4 - Week 1**	**A4 - Week 2**	**A4**	**Marks**
		Nov 16 - 23	**Nov 23 - 30**	**Nov 16 - Dec 7**	
Variable	normalize	1 public test		1 public test + 4 hidden tests	5
Elimination	restrict	1 public test		1 public test + 4 hidden tests	10
Algorithm	sum_out	1 public test		1 public test + 4 hidden tests	10
	multiply	1 public test		1 public test + 4 hidden tests	15
	min_fill_ordering		1 public test	1 public test + 4 hidden tests	15

Category	Test Descriptions	A4 - Week 1 Nov 16 - 23	A4 - Week 2 Nov 23 - 30	A4 Nov 16 - Dec 7	Marks
Naive Bayes Model	ve		1 public test	1 public test + 4 hidden tests	10
	naive_bayes_model		1 public test	1 public test + 4 hidden tests	20
Calculate Probabilities	explore		1 public test	1 public test (same as week 2) + 5 hidden tests	10
Reflection	bns_reflection.txt			Marked for completion only.	5

The Optional Early Feedback Deadlines

To encourage you to start working on the assignments early and tackle them incrementally, we have introduced *two optional early feedback deadlines* for each assignment. During the three weeks, we will provide some extra unit tests during the first two weeks. If you submit during either week, you can get additional feedback from these extra unit tests. Note that these extra unit tests will be *exclusively* available during their respective weeks and will no longer be available in the third week. These optional early feedback deadlines are not worth any marks.

We created these optional early feedback deadlines for the following reasons.

- *Encourage you to start the assignments early*: If you start early, you can take advantage of these extra unit tests, which won't be available later on.
- *Help you tackle the assignment incrementally*: We break down the assignment into smaller tasks so that the task for each week is more manageable.
- *Provide extra support*: The extra unit tests make it easier to debug and test your program by identifying problems in your program.
- *Avoid adding extra stress*: The early deadlines are not worth any marks. Therefore, if you are busy with other obligations and cannot take advantage of the early deadlines, you are not missing out on any marks, and you can still potentially get full marks on the assignments.

We have set up three projects on MarkUs: A4-Week1, A4-Week2 and A4. The first two projects (A4-Week1 and A4-Week2) contain the extra unit tests for the optional early deadlines. Please check the table above for the public tests available. Remember that the public tests in A4-Week1 and A4-Week2 are different from those in the A3 project on MarkUs.