1.0 **Introduction**

Situations where data scientists do not have a domain expert to model (conditional) independence but are asked to make predictions and given sufficient samples can be quite common (survey problems). A common technique to learn the structure in this situation is to consider a Bayesian Network(BN) using maximum likelihood estimation (MLE) and apply penalties of some sort, or to apply the Chow-Liu Tree(CLT) algorithm. We wanted our project to also explore what the Markov Random Field(MRF) assumption and compare these results.

Learning good structures is key to building accurate models, so we expect the results to reflect this. The structure learning should improve a basis case Naive Bayes(NB) inference solution. While a learned structure may not give the complete set of dependencies and independent relationships of the features (especially in our case as we are using a subset of the features available in the dataset) the learned structure is useful for solving inference problems.

1.1 Data

The dataset used for this project is the hacker rank survey dataset on the Kaggle website. The hacker rank website surveyed over 25,000 people. Some were hiring managers, some were working as software developers but not hiring managers, and some were students looking for employment, while others were students not ready to seek employment, yet.

There were over 30 questions asked in total. The 10 our dataset includes are:

1. Age (Categorical Distribution)

2. Age the person began coding (Categorical Distribution)

3. Sex (Bernoulli Distribution)

4. Education (Categorical Distribution)

5. Degree Focus (Categorical Distribution)

6. Job Level (Categorical Distribution)

7. Current Role (Categorical Distribution)

8. Industry (Categorical Distribution)

9. Vim or Emacs (Bernoulli Distribution)

10. Hiring Manager (Bernoulli Distribution)

We used this subset of the features and also a subset of the observations. The data were sanitized to numeric and we also disregarded missing values cases since the goal of the project is oriented around comparing learning structures of BN’s vs MRF’s not handling latent variables in structure learning in MRF’s. The dataset could have included as many as 15,534 observations. The subset of this that we chose to include was limited to 1,000 observations. 80% of these were used for training and 20% were used for testing.

1.2 Goals

Currently we are planning on calculating inference on the query:

1. Marginal Inference P(Vim) or (1-P(Vim))

If time presents itself we would also like to explore the query:

1. Conditional Inference P(Hiring\_Manager | Other\_Variables)

2.0 **Methodology**

2.1 Naive Bayes - basis case

To provide a basis case comparison for the structure learning inference query a ‘simple’ naive bayes model was run which assumed full connectivity of the features.

The model was trained on the 800 training observations, and tested on the 200 test observations. Frequencies were calculated empirically from the training observations and after applying a laplace smoothing, used to answer the inference query on the test observations.

Implementation in code is done in Matlab. There are 3 functions. The train function returns a model given training data and makes calls to the frequency table function which does the actual calculations of the empirical probabilities and is where the smoothing implementation occurs. The third function is the test function which returns both accuracy and predictions given a set of test data and the model.

2.1.1 Laplace Smoothing

The data subset were not necessarily robust enough to provide observations of enough features so an implementation of Laplace smoothing was done to accommodate the “new” instances of the multinomial categorical variables in the test data that were not seen in training. This is also equivalent to using a dirichlet prior, alpha = 1.

2.2 Chow Liu Tree

The BN structure learning algorithm was also implemented in Matlab in multiple functions. The function ‘ChowLiuTree’ is given the training data and returns a tree, a mutual information matrix, unimarginals and pair-marginals. The function calls a sub-function we made named ‘clt\_infoMatrix’ to calculate mutual information. The negative of these are taken as weights and passed to a built-in function, ‘minspantree’ to calculate the minimum spanning tree. Other built-in functions are used to modify this tree such that it is a directed, rooted tree. This main function also uses the training data and passes it to a sub-function we made called ‘Marginals’ to calculate unimarginals and pair-marginals of the features in the training data to be used later for predictions in inference queries.

A very important thing to note about the process required by the code is that the tree resulting from the ‘ChowLiuTree’ code and the one required by the ‘predictCLT’ are slightly different from each other. The edges still need to be redirected by the data scientist towards the feature of interest for the inference query question. This is by design. There is some user-interaction required to answer a question effectively in any real data analysis situation and redirecting edges for the neighbors of a tree-structure should not be too time intensive.

A second main function called ‘predictCLT’ is used with the tree, the unimarginals, the pair-marginals and the specified location of the inference query in the tree along with test data to predict classifications for the test data. It uses some built-in graph functions like “predecessors” but is otherwise self-sufficient.

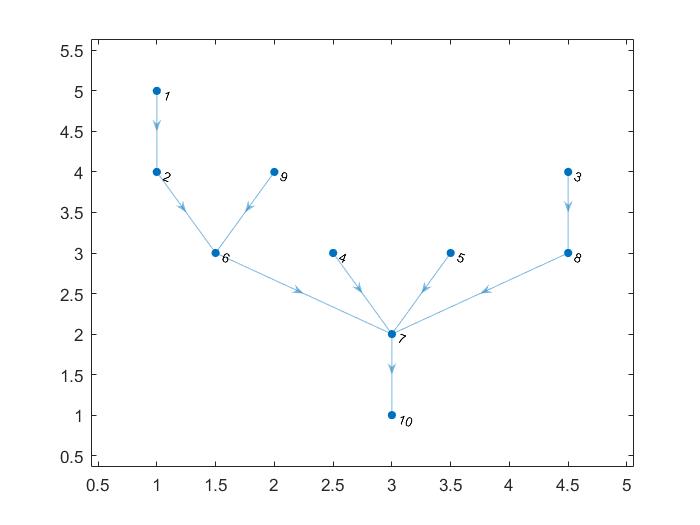
2.3

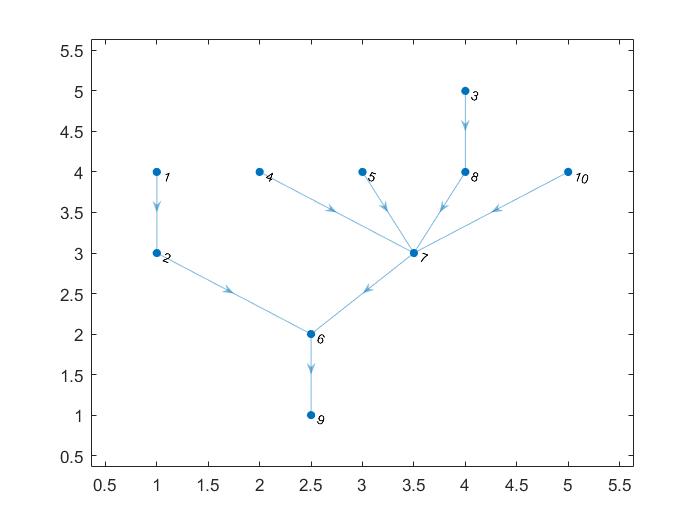
3.0 **Results**

**3.1** Structures Discovered

3.1.1 Chow-Liu Tree

Feature 10 is “Vim or Emacs?”, so the tree is directed towards this node for answering the inference query. The structure discovered by the chow-liu learning algorithm indicates it is primarily reliant on the “current role” feature.



For the additional query on marginal probability of hiring manager, which is feature 9, the edge 9,6 would need to be flipped and it would rely on feature 6 “Job Level”. The tree redirected to feature 9 is shown below for clarity on this point but is otherwise unused.

3.2 Inference Question Results: To VIM or not to VIM?

Table:

|  |  |  |
| --- | --- | --- |
| Model | Structure Description | Accuracy |
| Naive Bayes | Fully Connected | 58% |
| Chow-Liu Tree | Conditional Dependency Reduced to One Feature | 84% |

4.0 **Conclusions and Discussion**

As hoped, the structure learning exercise not only reduces conditional dependency of number of features from all to fewer - in this case 1, but also increases accuracy of inference queries.

Additionally, when looked at with the lense of common sense, the structure indicates dependencies between features that seem very obvious for example: “hiring manager” and “job level”.

The Vim or Emacs dependency on “current role” could be explained by “latent variables”. For example, the “current role” would affect an employee’s team’s software budget and available resources (colleague’s knowledge of either Emacs or Vim) in learning the software tools. It makes sense that similar job roles would have similar preferences for Vim or Emacs.

5.0 **References**

Ruozzi, Nicholas. “Structure Learning.” UTD, Apr. 2018.

[**https://www.utdallas.edu/~nrr150130/cs6347/2018sp/lects/Lecture\_17\_Struct.pdf**](https://www.utdallas.edu/~nrr150130/cs6347/2018sp/lects/Lecture_17_Struct.pdf)

Koller, Daphne, and Nir Friedman. *Probabilistic Graphical Models Principles and Techniques*. MIT Press, 2012.