Investigating Improvements for the Chow-Liu Algorithm

Frank-Edward Nemeth

Abstract

When consulting a learning problem, a useful representation of variable dependencies is a graph network. Graphical models serve to be useful tools that contain information about conditional probabilities, mutual information, as well as variable parameters. Once using a graphical model, the Chow-Liu algorithm is a powerful, polynomial time algorithm that can find an optimal tree structure. However, it would be interesting to see if there are any possible improvements to the algorithm, such as efficiency, memory space, and execution time.

The objective of this paper is to investigate two literary sources that improved the Chow-Liu algorithm for certain data problems. The concepts of the sources will be presented and analysed. The focus of the selected literature is to improve the accuracy and efficiency of the Chow-Liu algorithm.

This paper also tries to create a proof of concept simulation as a basis for future work. This uses some of the techniques introduced by the literary sources in order to test the viability of combining the suggested enhancements to a new modified Chow-Liu algorithm.

Introduction

Graphical models are useful tools to present learning problems. However, in order to learn a graphical model, some sort of algorithm must be used. The Chow-Liu algorithm is a polynomial time algorithm that returns the optimal tree structure with the optimal distribution. This introductory section will go over the relevant information regarding the traditional Chow-Liu algorithm, adapted as per the ECE1504 Course notes [1].

Tree Learning Definition

Suppose q is an arbitrary multivariate distribution, such that:

$$q(x) = q(x_1, x_2, \dots, x_d)$$

Where x is d-dimensional. In order to solve for this distribution, we attempt to approximate $q(x)$ with some distribution $p(x)$, such that $p(x)$ can factorize on a simple graphical model. Let T_Φ denote the set of all undirected trees with d- nodes and let $T_{\hat{\Phi}}$ denote the tree with the optimal distribution, $\hat{p}(x)$. Then,

$$T_{\hat{\Phi}} = \underset{T \in T_d}{argmax} \sum_{(i,j) \in T} I_q(x_i; x_j) \tag{1}$$

Where $I_q(x_i; x_j)$ is the mutual information between x_i and x_j with to distribution $q(x)$.

Chow-Liu Algorithm

Consider the following pairwise distribution:

$$\hat{q}(x_i, x_j) = \frac{1}{N} \sum_{k=1}^{N} 1\{x_{ki} = x_i, x_{kj} = x_j\} \tag{2}$$

This empirical count can be computed efficiently. The mutual information can then be found with respect to this $\hat{q}$, with notation $I_{\hat{q}}(x_i; x_j)$, and likewise,

$$T_{\Phi} = \underset{T \in T_d}{argmax} \sum_{(i,j)\in T} I_{\hat{q}}(x_i; x_j) \tag{3}$$

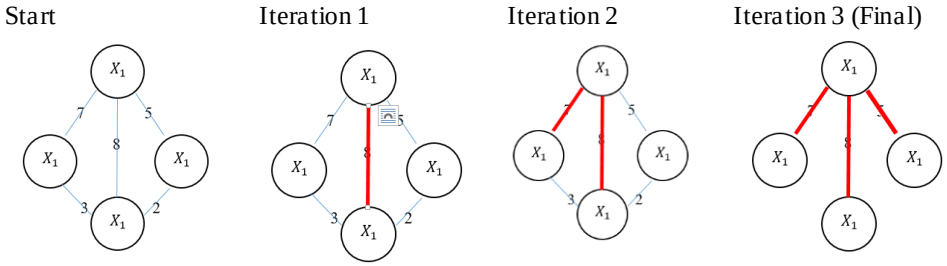
This optimization can be done by constructing a maximum spanning tree for the graphical model of the problem.

Steps for iterative Chow-Liu

1. Calculate the mutual information $I_{\hat{q}}(x_i; x_j)$ between all points (nodes) of the graphical model. The mutual information can be considered the weights between the nodes.
2. Initialize the tree with respect to an arbitrarily selected node
3. Add one edge to the tree. Add the edge that connects node pair x_i, x_j with the maximum mutual information (maximum weight)
4. Find all the vertices not yet included in the tree, and continue adding edges as per 3, where the edge has the largest possible mutual information without adding a vertex already in the graph

Figure 1 depicts an example of the algorithm

Figure 1: Example Chow-Liu Algorithm on a Graph



Objective

While the Chow-Liu algorithm is an efficient way to solve a graphical model, it is not without flaws. Specifically, there was interest in whether the traditional Chow-Liu algorithm could be improved in terms of efficiency and accuracy.

The goal of this paper is to look at two literary sources that present modifications to the Chow-Liu algorithm and potentially improve its performance on large datasets. Specifically, entropy estimations and large node (node combinations) approximations will be reviewed as potential tools to improve the Chow-Liu algorithm.

This paper will go over two literary sources, as indicated in the following sections. The key concepts of the papers will be summarized, and in some cases expanded upon. Some sections will reflect more critical thinking and work beyond what was presented in the paper.

Literary Source 1: Beyond Maximum Likelihood: Boosting the Chow-Liu Algorithm for Large Alphabet[2]

Subsections are adapted as per the literary source[2].

Problem statement

The Chow-Liu algorithm is designed for learning tree graphical models. From a closer inspection, the Chow-Liu algorithm performs optimally when the number of samples grow towards infinity. However, this is not the case with datasets that lack observations relative to the alphabet size. The Chow-Liu algorithm is highly sub-optimal in these high dimension

distributions, in the regime where the alphabet size of each node is comparable to the number of observations.

Improving the Chow-Liu algorithm

The original implementation of the Chow-Liu algorithm attempts to solve the Maximum Likelihood Estimator using a tree. Specifically, a Maximum Weight Spanning Tree (MWST) is formed using the mutual information between the nodes of a graph (as depicted in the Introduction and Figure 1).

However, as referenced in [2], in high dimensional regimes, finding the mutual information can be sub-optimal. This is best describe in the theorem described in the paper, as it appears below for convenience:

> ***Theorem 1***[2]: Suppose we have two random variables $X_1, X_2 \in X, |X| < \infty$. Also note for reference, $|X|$ and S are interchangeable. The minimax sample complexity in estimating the mutual information $I(X_1; X_2)$ under mean squsred error is $\Theta\left(\frac{|X|^2}{\ln|X|}\right)$, while the sample complexity required by the empirical mutual information to be consistant is $\Theta(|X|^2)$

Interpreting this theorem means that it is sufficient to take fewer samples, $n \gg \frac{|X|^2}{\ln|X|}$ if using a minimax rate-optimal estimator to compute the mutual information. At the same time however, there exists distributions where $n \gg |X|^2$ in order to bound the error.

This indicates a possible improvement for the traditional Chow-Liu algorithm. Specifically, finding a more computationally efficient mutual information estimator could reduce the sample size complexity from $\Theta(|X|^2)$ to $\Theta\left(\frac{|X|^2}{\ln|X|}\right)$, or even $\Theta(|X|)$.

Current progress towards Entropy Estimation

The paper discuses a series of recent advancements in functional estimation. In a previous paper by the same authors, a linear entropy function ($\Theta(S)$ samples) was found. This algorithm is complex and has been left out due to its length, but can be found in the additional reference[3]. The prior work also does a comprehensive review of other estimators and their complexity.

Testing –small section

The paper goes on to test their developed entropy function with the Chow-Liu algorithm (in an algorithm they call "Modified Chow-Liu"). In tests of convergence on a small star graph. In order to test the effect of the number of samples on the algorithm, the number of samples was varied, and each Chow-Liu algorithm (original and the new modified) were run. A wrong edge ratio was measured as the difference between the outputted edges, and the expected true edges (with a score of 1 meaning the estimated tree is maximally different from the true tree). Results showed that when fewer than 3×10^3 samples were used, both algorithms returned ratios of 1 (maximally different tree). As the sample size exceeded 6×10^3 samples, the modified algorithm quickly dropped its ratio to 0, indicating a maximally correct reconstruction. The original algorithm continued to perform poorly until the sample size exceeded 47×10^3, showing an almost 8 times improvement between the modified and original algorithms.

The paper also suggests and tests and improvement to the Chow-Liu algorithm with respect to Bayesian Network classifiers. These involve conditioning the mutual information calculations typically used in the Chow-Liu algorithm over some Bayesian classifier C.

Comments on Paper-Extension

While this particular paper does not delve into the actual entropy models used, it gives a strong insight to an area of optimization for the Chow-Liu algorithm. Entropy approximations appear in many fields, and finding an adequate approximation can greatly boost the performance of learning trees, as shown in the experimental data. Using different entropy estimators can often show less generalizability, as many of the entropy estimations are complex algorithms requiring additional information[3]. For instance, the Bayesian Network Example tested in the paper requires conditional empirical mutual information, where the mutual information is conditioned on the class labels associated with the learning problem. It is still interesting to see whether there are use cases where replacing the entropy calculations with estimators can yield an adequate mutual information while also improving performance (as presented in the paper)

Paper 2: Improving Chow-Liu Performance by Mining Association Rules[4]

Problem statement

When constructing a Chow-Liu tree, there are situations that can arise where information is lost through the estimation process. This occurs at points where the underlying tree structure of a dataset is not best represented by the particular graph. An example is depicted in Figure 2a

Figure 2: Combination of Nodes in a Graph Example (originally Fig.2 in[4])

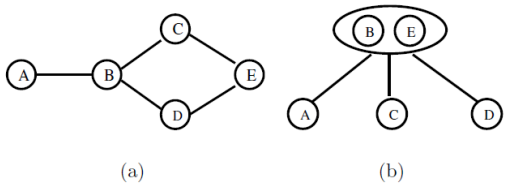


Fig. 2. (a): The underlying structure of a dataset (b): A large node tree structure we call "LNCLT"

It is noticed that the underlying structure is difficult to restore due to the cyclic nature of the graph. A suggested remedy for this issue is to combine nodes into a "large node" in order to properly represent the dataset as a tree. Figure 2b depicts this Large Note Chow-Liu Tree (LNCLT)

As a result this paper proposes to create a LNCLT, whereas combination rules are defined in order to construct a large node structure from the results of the traditional Chow-Liu algorithm. This allows the final structure to maintain a tree structure, giving it a strong resistance to overfitting problems.

Learning Large Node Chow-Liu Tree

This section will summarize the required definitions and proofs required for the Large Node Chow-Liu Tree.

Combination Transformation

Definition 1: A combination transformation is defined to be a transformation in a tree. structure T. This transformation combines several nodes into a large node and keeps the connection relationship of T[4].

A combination transformation is depicted in Figure 2b, where two nodes are combined.

Combination Rules

The rules, as defined in the paper are as follows[4]:

Rule 1 Sibling rule: The nodes to be combined satisfy that the set of these nodes are sibling relationship, i.e., there exists another node as their common parent.

Rule 2 Parent-child rule: The nodes to be combined satisfy that the set of these nodes can be sorted as a sequence based on a certain node as the root, in which each node is the parent node of its sequent node.

Rule 3 Association rule: The nodes to be combined satisfy that, under a given confidence level and a minimum support, the set of these nodes denoted by A forms an association rule , i.e., A→C, where C is the class label.

Rule 4 Bound rule: The nodes to be combined satisfy that the number of these nodes is fewer than a given integer bound K.

Theoretical log likelihood

In order to prove that transformations following Rule 1 and Rule 2 are valid, the log likelihood of the new tree structure must be shown to be greater than or equal to the prior tree structure. This section will adapt the proof from the paper in order to show the validity of the proposed algorithm.

Log likelihood

For some dataset S and n variables, the log likelihood l_t can be written as

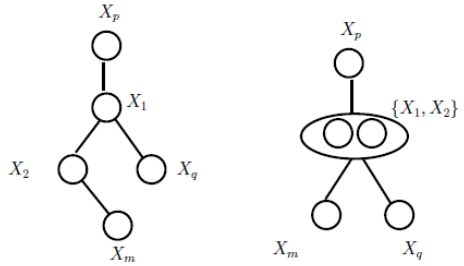
$$l_t(x^1, x^2, \dots, x^n) = \sum_{i=1}^{n} \sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right) \tag{3}$$

Where $x^1, x^2, \dots, x^n$ are n dimensional vectors such that x^k is the set $\{x_1^k, x_2^k, \dots, x_n^k, 1 \leq k \leq s\}$. $j(i)$ Represents parent node for some node i. This log likelihood applies when the dataset is fit in accordance to the Chow-Liu algorithm, (is maximized when the dataset is fit as a maximum weight spanning tree, with the weights being the mutual information between nodes)

Rule 2 Log Likelihood proof

For this proof we consider a tree structure as in Figure 3. We depict some parent structure X_p with child X_1. Node X_1 has child X_2, and some structure X_q. Node X_2 has some child structre X_m. As a first step, we rewrite the log likelihood formulation to stipulate more clearly this particular tree structure:

Figure 3: Rule 2 Log Likelihood General Combination (Originally Fig 4 in [4])



$$\begin{aligned} l_t(x^1, x^2, \dots, x^n) &= \sum_{i=1}^{n} \sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right) \\ l_t &= \sum_{i \neq X_1, X_2, X_m, X_q}^{n} \sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right) \\ &\quad + \sum_{k=1}^{s} \left[\log P\left(x_{X_2}^k | x_{X_1}^k\right) + \log P\left(x_{X_m}^k | x_{X_2}^k\right)\right. \\ &\quad \left. + \log P\left(x_{X_q}^k \middle| x_{X_1}^k\right) + \log P\left(x_{X_1}^k | x_{X_p}^k\right)\right] \end{aligned} \tag{4}$$

If we conduct a combination transformation as per Rule 2 (depicted in Figure 3) then we can further write the expanded formula as:

$$l_{t*} = \sum_{i \neq X_1, X_2, X_m, X_q}^{n} \sum_{k=1}^{s} \log P\left(x_i^k \middle| x_{j(i)}^k\right) + \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_1}^k x_{X_2}^k \middle| x_{X_p}^k\right)\right] \quad (5)$$

Now we can consider only the different terms between our two equations, (4) and (5). Using the notations in the article:

$$R(l_t) = \sum_{k=1}^{s} \left[\log P\left(x_{X_2}^k \middle| x_{X_1}^k\right) + \log P\left(x_{X_m}^k \middle| x_{X_2}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_1}^k\right) + \log P\left(x_{X_1}^k \middle| x_{X_p}^k\right)\right] \quad (6)$$

$$R(l_{t*}) = \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_1}^k x_{X_2}^k \middle| x_{X_p}^k\right)\right] \quad (7)$$

Simplifying (6) and using the standard definition of entropy, we obtain:

$$R(l_t) = \sum_{k=1}^{s} \left[\log P\left(x_{X_2}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_2}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_q}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_1}^k \middle| x_{X_p}^k\right)\right]$$

Using conditional probability on the last term:

$$R(l_t) = \sum_{k=1}^{s} \left[\log P\left(x_{X_2}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_2}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_q}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log \frac{P\left(x_{X_1}^k x_{X_p}^k\right)}{P\left(x_{X_p}^k\right)}\right]$$

$$R(l_t) = \sum_{k=1}^{s} \left[\log P\left(x_{X_2}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_2}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_q}^k \middle| x_{X_1}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_1}^k x_{X_p}^k\right)\right] - \sum_{k=1}^{s} \left[\log P\left(x_{X_p}^k\right)\right]$$

$$R(l_t) = -H(X_2|X_1) - H(X_m|X_2) - H(X_q|X_1) - H(X_1X_p) + H(X_p) \quad (8)$$

Likewise, we can do the same for (7):

$$R(l_{t*}) = \sum_{k=1}^{s} \left[\log P\left(x_{X_m}^k \middle| x_{X_1}^k x_{X_2}^k\right) +\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right)\right] + \sum_{k=1}^{s} \left[\log P\left(x_{X_1}^k x_{X_2}^k \middle| x_{X_p}^k\right)\right]$$

Using a similar process, we reduce to

$$R(l_{t*}) = -H\left(X_2 \middle| X_1X_p\right) - H\left(X_m \middle| X_2X_2\right) - H\left(X_q \middle| X_1X_2\right) - H\left(X_1X_p\right) + H\left(X_p\right) \quad (9)$$

By comparing (8) and (9), and utilizing the trait of entropy, $H(X|Y) \geq H(X|YZ)$, we can eliminate the like the like terms, and compare the remain terms:

$$\begin{gathered} H(X_2|X_1) \geq H\left(X_2 \middle| X_1X_p\right) \\ H(X_m|X_1) \geq H(X_m|X_1X_2) \\ H\left(X_q \middle| X_2\right) \geq H\left(X_q \middle| X_1X_2\right) \end{gathered} \quad (10)$$

As a result we can reasonably deduce

$$R(l_t) \leq R(l_{t*})$$

And likewise

$$l_t \leq l_{t*}$$

Rule 1 Log Likelihood

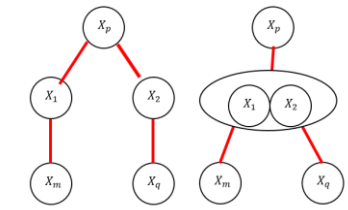


Figure 4: Rule 1 Log Likelihood General Combination

This proof was left out of the original paper. As a slight extension it will be done here. For this proof we consider a tree structure as in Figure 4

We depict some parent structure X_p with children X_1 *and* X_2 . Node X_1 has child structure X_m. Node X_2 has some child structre X_q. As a first step, we rewrite the log likelihood formulation to stipulate more clearly this particular tree structure:

$$l_t(x^1, x^2, \dots, x^n) = \sum_{i=1}^{n}\sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right)$$

$$l_t = \sum_{i \neq X_1, X_2, X_m, X_q}^{n} \sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right) + \sum_{k=1}^{s}\left[\log P\left(x_{X_1}^k | x_{X_p}^k\right) + \log P\left(x_{X_2}^k | x_{X_p}^k\right) + \log P\left(x_{X_m}^k | x_{X_1}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_2}^k\right)\right] \quad (11)$$

If we conduct a combination transformation as per Rule 1 (depicted in Figure 4) then we can further write the expanded formula as:

$$l_{t*} = \sum_{i \neq X_1, X_2, X_m, X_q}^{n} \sum_{k=1}^{s} \log P\left(x_i^k | x_{j(i)}^k\right) + \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_1}^k x_{X_2}^k | x_{X_p}^k\right)\right] \quad (12)$$

Now we can consider only the different terms between our two equations, (11) and (12). Using the notations in the article:

$$R(l_t) = \sum_{k=1}^{s}\left[\log P\left(x_{X_1}^k | x_{X_p}^k\right) + \log P\left(x_{X_2}^k | x_{X_p}^k\right) + \log P\left(x_{X_m}^k | x_{X_1}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_2}^k\right)\right] \quad (13)$$

$$R(l_{t*}) = \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right) + \log P\left(x_{X_1}^k x_{X_2}^k | x_{X_p}^k\right)\right] \quad (14)$$

Simplifying (13) and using the standard definition of entropy, we obtain:

$$R(l_t) = \sum_{k=1}^{s}\left[\log P\left(x_{X_1}^k | x_{X_p}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_2}^k | x_{X_p}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_q}^k \middle| x_{X_2}^k\right)\right]$$

Using conditional probability rule:

$$R(l_t) = \sum_{k=1}^{s}\left[\log P \frac{P\left(x_{X_1}^k x_{X_p}^k\right)}{P\left(x_{X_p}^k\right)}\right] + \sum_{k=1}^{s}\left[\log P \frac{P\left(x_{X_2}^k x_{X_p}^k\right)}{P\left(x_{X_p}^k\right)}\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_q}^k \middle| x_{X_2}^k\right)\right]$$

$$R(l_t) = \sum_{k=1}^{s}\left[\log P\left(x_{X_1}^k x_{X_p}^k\right)\right] - \sum_{k=1}^{s}\left[\log P\left(x_{X_p}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_2}^k x_{X_p}^k\right)\right] - \sum_{k=1}^{s}\left[\log P\left(x_{X_p}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_q}^k \middle| x_{X_2}^k\right)\right]$$

$$R(l_t) = -H(X_m | X_1) - H(X_q | X_2) - H(X_1 X_p) - H(X_2 X_p) + 2H(X_p) \quad (15)$$

Likewise, we can do the same for (14)

$$R(l_{t*}) = \sum_{k=1}^{s}\left[\log P\left(x_{X_m}^k | x_{X_1}^k x_{X_2}^k\right) +\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_q}^k \middle| x_{X_1}^k x_{X_2}^k\right)\right] + \sum_{k=1}^{s}\left[\log P\left(x_{X_1}^k x_{X_2}^k | x_{X_p}^k\right)\right]$$

Using a similar process:

$$R(l_{t*}) = -H(X_m|X_1X_2) - H(X_q|X_1X_2) - H(X_1X_p) - H(X_2X_p) + 2H(X_p) \quad (16)$$

By comparing (14) and (15), and utilizing the trait of entropy $H(X|Y) \geq H(X|YZ)$, we can eliminate the like the like terms, and compare the following:

$$\begin{aligned} H(X_m|X_1) &\geq H(X_m|X_1X_2) \\ H(X_q|X_2) &\geq H(X_q|X_1X_2) \end{aligned} \quad (17)$$

As a result we can reasonably deduce again,

$$R(l_t) \leq R(l_{t*})$$

And likewise

$$l_t \leq l_{t*}$$

Rule 3 Comments

Rule 3 pertains to nodes that likely share attributes. As described in the paper, since the attributes share an association rule pointing to the class label C, they are likely to be more dependent on one another than other attributes. This means they are likely to perform well as a single node, and thus should be grouped together with a high priority.

Rule 4 Comments

Due to space constraints, the full proof for this section will not be included. Rule 4 describes situation of merging multiple nodes. Considering a large K number of combinations, many nodes will be combined together, essentially creating one large node in the limiting case. This would likely be a poor estimator of the problem. As a result there needs to be some minimum support measure that limits the amount of nodes combined.

The derived minimum support was as follows:

$$s_m \geq \frac{2}{(1 - p_{cf})^2} \quad (18)$$

Where N is the total number of cases in the dataset, and p_{cf} is the confidence level specified by the user.

Paper Results

The full algorithm from the paper appears in Appendix 2, as a snapshot of the paper it came from[4]. The paper applied the algorithm on the MNIST dataset. As expected from the proofs of Rule 1 and Rule 2, the LNCLT was found to return a larger log likelihood than the traditional CLT. More generally, the LNCLT performed better with stronger result on the classification of the datasets (with better results on all 10 datasets). One of the limitations presented in the paper is the time-analysis of the algorithm. Combining many of the nodes might have been more accurate, but likely will also take more computational resources. This was an interesting takeaway that would be interesting to explore, especially in conjunction with some of the efficient methods presented in the first paper.

Exploration of Findings

For this part of the paper, I investigate some of the suggested improvements for the Chow-Liu Algorithm. Specifically, a small-scale, proof of concept baseline calculation was created to see the feasibility of utilizing both methods to improve the Chow-Liu algorithm.

PLEASE NOTE: **Figures and tables for this section appear in Appendix 1 since space was limited**

Timing Improvements Using Entropy Estimates

The first paper served as an inspiration for finding an entropy estimator. An attempt was made to use the linear entropy estimator suggested in the paper. However, this proved to be far too difficult, as there were many hurdles in the programming language of choice (Python). Specifically, certain expected libraries stopped functioning, making visualization and debugging difficult.

Instead, it can be noticed that a simple change to the entropy calculation should still have observable effects, albeit not as drastic as those presented in the paper.

Specifically we can consider the standard formula for entropy (the Shannon entropy, as used in papers and class), $H(P) = -\sum_{i=1}^{n} p_i \log p_i$

Where the sum of all p_i for $i = 0,1 \ldots n$ is equal to 1. In a computer program (such as Python) some functions such as the logarithm take slightly more time than other methods. Taking the logarithm repeatedly can add up to a large efficiency loss, especially for high values of n. Similar to how the paper presented found an entropy estimator to reduce the sample space required, we can similarly find a simple entropy estimator to save us time calculating logarithms. Consider the Rényi entropy: $H_\alpha(P) = \frac{1}{1-\alpha} \log \sum_{i=1}^{n} p_i^\alpha$

With $\alpha \geq 0$. This formula approximates $H(P)$, and becomes a stronger approximation the closer α is to 1 (as $\alpha \to 1, H_\alpha \to H$). Something we can immediately notice is the reduction in logarithm calculations, since it now appears outside the summation. While not necessary, reducing the entire runtime complexity as the presented paper did, we can still achieve some mild optimizations.

In order to test this, simulations were made. Arrays of random p_i values for various sizes of n were constructed to simulate doing repeated entropy calculations. Values, although random, were normalized to maintain the sum of all p_i for $i = 0,1 \ldots n$ is equal to 1. The results of the simulations are as follows

Hyperparameter Selection

Table 1 in Appendix 1 shows results of measuring the error (absolute difference between the entropy calculations) for various α and n values. This was to assess how accurate the estimator is. As expected, for α significantly close to 1 (in other words $1 - \alpha \leq 0.000001$) the estimator is accurate to over 6 decimal places, regardless of the size of n. This is mildly surprising that the size of the sets had little effect on the error measures.

Timing Analysis

Figure 5 in Appendix 1 plots the time taken for the script to execute, given variations in n. α was fixed to 0.999999, as this was found to be a threshold for a highly accurate value in the previous section. As expected, the change in estimator did not yield graphs of different complexities. Both appear linear to n. However, the rate (slope) of the original entropy calculation is indeed faster (slope is steeper). This causes much more time to be taken for large n. Without compromising accuracy, using the entropy estimator yields a small improvement to the performance.

Although small, with large n, programs can easily become very cumbersome. The Actual Chow-Liu algorithm requires repeated entropy calculations, as the mutual information formula:

$$I(X;Y) = H(X) + H(Y) - H(XY) = H(X) - H(X|Y)$$

can require two to three entropy calculations.

Feasibility of the Chow-Liu Algorithm with Node combinations.

The second small simulation was to create the Chow-Liu algorithm, and run it for a particular model. The idea was to see how long some Node combinations would take, as per the algorithm presented in the second discussed paper. This was meant to work in conjunction with the prior entropy estimates.

Mutual information measures were simulated using the prior framework. This was approximated once again using random numbers to calculate multiple entropies for each edge on the model graph. The timings for the calculation of the mutual information appeared in Figure 6, and appear as scaled versions of the graphs in Figure 5. Figure 7 shows the sample node graph with the Chow-Liu Tee edges in red. Figure 8 shows the node combination (BC) on the same graph, as per Rule 2[4]. Overall, the steps to combine nodes took a marginal and statistically insignificant amount of time, especially compared to the mutual information calculations. As a result, even when trying to scale the simulations, it is inconclusive how much extra the node combinations take.

Conclusion and Next Steps

Overall this paper's goal was to introduce complementary methods to improve the traditional Chow-Liu algorithm. Namely, two literary sources were presented, with the details of the papers explored, and some proofs shown. Furthermore a small simulation was done to start testing the feasibility of implementing the algorithms in conjunction with one another.

Since the trials were very small, and the graphs learned were not overly complex, it is quite difficult from the simulations to draw many meaningful conclusions. The original intention with the project was to learn a Bayesian Network, akin to how the second literary source applied their algorithm on the MNIST dataset. Furthermore, the combinations of a more efficient entropy estimate with the increase in complexity form the node combination methods would have been an interesting case study (to test whether the two additions will cancel out to limit the change in complexity of the modified approach). This approach was attempted, but difficulties in programming caused issues. Namely, one of the more frequented Python libraries (Pomegranate) for solving Bayesian graphs stopped working on newer machine, yielding difficulties in debugging and visualizations. This in conjunction with time constraints limited the external analysis, causing the entire report to focus more on summarizing the articles. Furthermore, scaling down the problem also rendered many of the entropy estimator formulas difficult to apply, as there was a series of simulated data to work with.

Future steps would be to revive a larger testing scenario, where the timing and efficiency of the algorithms could be looked at in more depth. Specifically, finding new tools to learn a Bayesian Network could be used, along with a modified Chow-Liu algorithm that takes advantage of entropy estimators and node combinations.

References

[1]A. Khisti, *ECE1504 Course notes - week1*. Toronto: University of Toronto, 2020.

[2]J. Jiao, Y. Han and T. Weissman, "Beyond maximum likelihood: Boosting the Chow-Liu algorithm for large alphabets", *2016 50th Asilomar Conference on Signals, Systems and Computers*, 2016. Available: 10.1109/acssc.2016.7869051 [Accessed 18 December 2020].

[3]Jiantao Jiao, K. Venkat, Yanjun Han and T. Weissman, "Minimax Estimation of Functionals of Discrete Distributions", *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2835-2885, 2015. Available: 10.1109/tit.2015.2412945 [Accessed 19 December 2020].

[4]K. Huang, I. King, M. Lyu and H. Yang, "Improving Chow-Liu Tree Performance by Mining Association Rules", *Neural Information Processing: Research and Development*, pp. 94-112, 2004. Available: 10.1007/978-3-540-39935-3_6 [Accessed 19 December 2020].

Appendix 1: Simulation Figures and Tables

Table 1: Error, abs(H(P)-H_a(P)), For various n and α selections (rounded to 3 significant figures)

		Number of samples (n)							
		5	10	50	100	500	1000	5000	10000
α	0.1	0.150	0.285	0.134	0.165	0.160	0.153	0.164	0.160
	0.5	0.0568	0.0750	0.0730	0.0752	0.0738	0.0676	0.0740	0.0744
	0.9	0.00496	0.0119	0.0136	0.0108	0.0119	0.0134	0.0131	0.0127
	0.99	0.000550	0.00200	0.00127	0.00107	0.00116	0.001274	0.00124	0.00128
	0.999	2.66E-05	0.000128	0.000113	0.000135	0.000130	0.000121	0.000127	0.000123
	0.9999	8.50E-06	6.55E-06	1.32E-05	1.10E-05	1.25E-05	1.25E-05	1.30E-05	1.24E-05
	0.99999	8.76E-07	9.71E-07	8.56E-07	1.74E-06	1.30E-06	1.26E-06	1.26E-06	1.28E-06
	0.999999	1.91E-07	6.83E-08	1.10E-07	1.31E-07	1.27E-07	1.23E-07	1.25E-07	1.21E-07

Figure 5: Simulated time Taken per increase in Sample for H

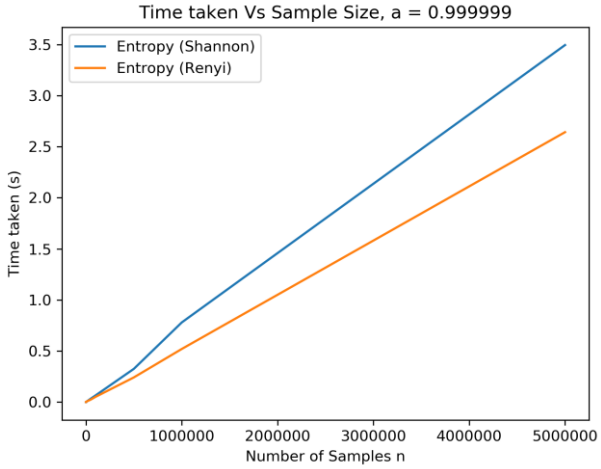


Figure 6: Simulated time Taken per increase in Sample For I

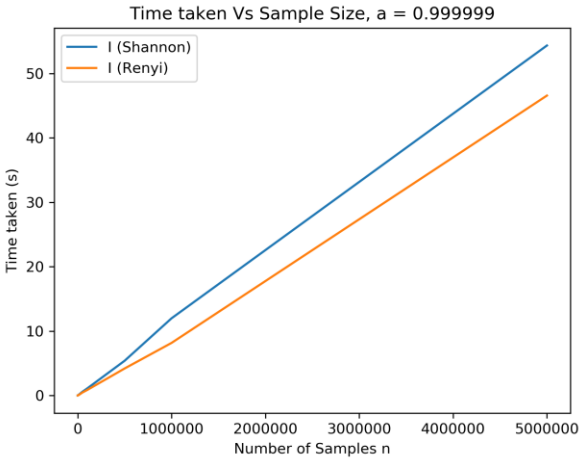


Figure 7: Sample graph with weights and Chow-Liu tree in red

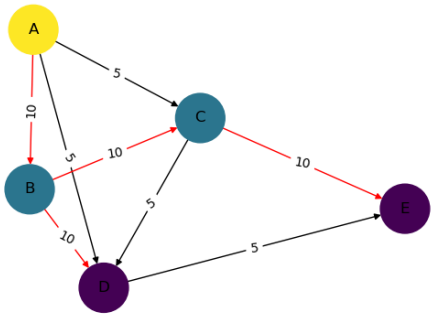
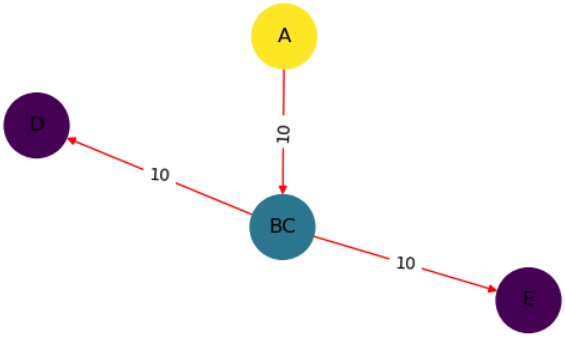


Figure 8: Combined node BC in Chow-Liu tree from Figure 7



Appendix 2: Definitions

Definition

As per [4], the following notation sets up the LNCLT problem.

- V denotes a set of n random and discrete varaibles, and A is some subset of V
- T denotes a graph (V, E) with V verticies as defined prior, and E undirected edges
- T is a tree if T is connected and acyclic, and it is a spanning tree if the number of edges $|E|$ is equal to one less the number of vertices, $|V| - 1$
- Let V^* be a set that is a subset of V, with conditions satisfied:
    - $\cup U_i \in V^* U_i = V$
    - $U_i \cap U_j = \emptyset$ with $U_i, U_j \in V^*, i \neq j$
- A large node tree $T^*(V^*, E^*)$ is a tree where V^* is the vertex set that matches the above conditions, and E^* are the corresponding edges. It is important to note that the vertices of T^* are a subset of V with no overlapping variables

Node Combination Algorithm

The following is a direct copy from [4] showing the algorithm

Phase 1: Detecting all the association rules $X \rightarrow Y$, where Y is specified by the class variable C and X is a subset of attributes set, with the cardinality fewer than a bound K.

(1) Determine a good value of the minimum support, based on (17). Call the Apriori procedure to generate the association rules, whose X's have the cardinality fewer than K.

(2) Record all the association rules together with their supports into list L.

Phase 2(3): Drafting Chow-Liu Tree [3].

Phase 3: Adapting the tree structure based on combination transformation

(4) According to tree T, filter out association rules from L whose X's do not satisfy combination conditions, i.e., Rule 1 or Rule 2 from L. We get the new L'.

(5) Sort L' in descending order based on the supports of the association rules.

(6) Do until L' is $NULL$.

(a) Do the combination transformation based on the first itemset l_1 of L'.

(b) Delete l_1 and any other association rules l_i in L' which satisfy the following condition:

$$l_1.X \cap l_i.X \neq \varnothing,$$

where $l_1.X$ and $l_i.X$ refers to the X part of l_1 and l_i, respectively.

(c) Examine whether the newly generated items satisfy the combination rules. If yes, insert them into L' and sort L'.

(d) Go to (a).