Improving Visual Cues for the Interactive Learning of Bayesian Networks

Lisa Li
UC Berkeley
lisa.qy.li@berkeley.edu

Omar Ramadan
UC Berkeley
omar.ramadan@berkeley.edu

Phoebe Schmidt
UC Berkeley
pbschmidt@berkeley.edu

INTRODUCTION

A Bayesian network is a visual representation of a set of variables and relationships that govern their conditional independence. Bayes nets (BNs) have been useful in solving many data analysis problems because they encode dependencies and causal relationships between variables without overfitting [5]. Learning the optimal structure of a Bayesian network, however, is an intractable problem [7]. There are two very different approaches for structure learning from heuristics. The constraint-based approach starts with a fully connected graph and iteratively removes edges if conditional independencies are measured in the data [8]. The more popular score-and-search approach on the other hand starts with an empty graph and searches the space of graphs for the highest scoring graph [8]. Such automated optimization-based approaches are prone to overfitting especially when the system is only partially observed. Structure learning is most effective when domain specific insights are leveraged. Interactive structure learning is where a domain expert uses heuristics from learning algorithms to guide the design process [3]. Some software programs implementing this type of learning have been developed, but are limited in their interactivity and the heuristics that they visualize.

We seek to improve existing visual tools that aid experts in learning an optimal Bayes net structure with information visualization techniques that will encode more dimensions of data with cleaner and more informative interaction. Specifically we augmented the score-and-search structure learning capability of an existing utility, OpenMarkov. We developed four interactions that we believe are integral to an interactive learning workflow: (i) *Evaluation View*: a visual analysis of how well the structure conforms to the learning data and an overview of proposed edits to the network (ii) *Edit View*: a visual exploration of positive edits to the network from a particular variable (iii) a *Look-ahead graph*: a preview of the edits applied by the learning algorithm after a number of steps for a back and forth interchange between the user and the automatic method (iv) *Objective Function View*: support for visualizing how the objective function changes with edits and tracking overfitting on a held-out dataset. In this paper we will first briefly review Bayes nets and conditional probability, discuss prior relevant work and the scope of OpenMarkov as it relates to BN structure learning, and present our visual tool and conclusions.

CONDITIONAL PROBABILITY AND BAYESIAN NETWORKS

Conditional independence is our most basic and robust form of knowledge about uncertain environments [1]. To provide a definition, A is conditionally independent of B given C if and only if knowledge of B does not affect the probability of A given C.

$$A \perp\!\!\!\perp B \mid C \iff P(A \mid B, C) = P(A \mid C)$$

A Bayes net visually encodes a set of variables and their conditional dependencies with a directed acyclic graph (DAG) of nodes and edges together with a conditional probability table for each node and its parents [1]. A BN implicitly encodes a joint distribution as a product of local conditional distributions; by performing the Chain Rule, or multiplying relevant conditionals together, an expert can determine the probability of a certain assignment of variables in the BN [1]. Bayes nets have proven useful in data analysis since the eighties, because they represent causal relationships and combine domain knowledge with data with decent protection from overfitting [5]. For instance, an employee at a sales company must decide whether they should increase exposure of a certain advertisement to increase sales. They could refer to a Bayes net to determine whether ad exposure causes increased sales, and by how much. Alternatively, an expert in the field of medicine is faced with a set of data about patients. The data includes various pieces of information about each patient, including recent travel history, family medical history, and symptoms. A complete Bayes net representing these variables and their relationships could be useful in diagnosis and studying trends for future patients. Bayes net representations are useful; the problem lies in constructing a complete Bayes net: an NP-hard problem. In the following section we will discuss existing research that aims to find Bayes net structures that best represent the joint distribution of a given set of data points.

RELATED WORK

There are two modes of learning in Bayesian networks: parameter learning and structure learning [3]. Parameter learning focuses on finding the probability distributions of a set of data while structure learning focuses on finding the the topology of the variables within the network. We focus our work solely on the latter, structure learning, and the improvements that can be brought to the process through more effective visualization and interactions.

The problem of finding the optimal BN structure for a set of related variables can represented as a state space search with the goal being to find the structure that best expresses the dataset. Unfortunately the number of DAGs as a function of the number of variables, G(n), grows at a super-exponential rate and is given by the following recurrence [8]:

$$G(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} G(n-k)$$

An exhaustive search of the space is therefore infeasible.

Automatic structure learning

In practice we can use a variety of approximation algorithms. Constraint based approaches such as the IC algorithm [9] and the PC algorithm [10] begin with a fully connected graph and compute the conditional independencies between different sets of variables from the data to eliminate edges from the network. The problem with this approach is that the repeated independence tests lose statistical power.

Score and search methods on the other hand start with an empty graph and navigate the search space by adding edges and scoring the network to find a network that yields the maximal score. We can perform a local search such as greedy hill climbing or a global search such as Markov Chain Monte Carlo.

Furthermore, we can introduce restarts where we randomly perturb the network and restart the search to avoid local optima. Popular classes of scoring functions include the BDeu, the popular K2 score and the BIC (Bayesian Information Criterion) [8, 11, 12].

Interactive structure learning

Noisy data measurements coupled with the limitations of approximation models frequently yield models that are suboptimal. In most cases, experts will manually edit the output models from these algorithms. Existing tools for automatic structure learning with these algorithms are largely script based and dont provide functionality for users to interact with the generated graphs [8, 13, 14].

Typically, a user will generate learn the graph structure from the data and use a graph visualization tool such as GraphViz to evaluate the learned network, after which a domain expert will analyze the relationships to investigate their validity. The user can evaluate individual links and their conditional dependence using measures such as link and connection strength. The Link Strength package for BNT allows users to visualize such metrics using the edge thickness [15]. After analyzing and modifying the structure, the user may then pass the network through the learning algorithm again to identify additional edits.

OpenMarkov: Incorporating User Interaction

OpenMarkov, a project from the Research Centre for Intelligent Decision-Support Systems of the UNED in Madrid, Spain, is a comprehensive toolkit for probabilistic graphical models (PGMs), including but not limited to Bayesian networks [2]. It includes a module for interactive Bayes net structure learning that allows users to learn a network from a dataset using both PC and score-and-search algorithms with a variety of scoring metrics [2]. It includes automatic learning, which completes the BN structure with the selected algorithm and metric, or interactive learning, which allows the user to make manual edits or select edits from a list of suggestions [2]. All edits are undoable [2].OpenMarkov is the most advanced existing software tools we could found that is open to public. It comes with diverse capabilities and its interface is simple and clear. However, OpenMarkov does not dynamically visualize the performance of each actions user performed; for the remainder of this paper we will use the name OpenMarkov to refer specifically to the Bayes Net learning tool.

METHODS

We believe interactive structure learning of Bayes nets could be greatly improved with better visualization and user interaction. Given the time constraints of this project, we sought to leverage the advanced metrics and computations already implemented in OpenMarkov and augmented the graphical user interface to test whether these features can improve user experience. We added four main features: (i) *Evaluation View*, (ii) *Edit View*, (iii) a *Look-ahead graph*, and (iv) an *Objective Function View*. Users experience no conscious literal switching between various views as they are titled above; rather we conceptualize the features as different views because each one has unique features designed specifically to aid in different points of the network construction process.

Evaluation View

Evaluation view is the name for the default view of OpenMarkov plus our additional graphical tools. Before our changes, OpenMarkov presents the user with a circle of nodes, one for each variable. Depending on the learning algorithm selected, there are either zero edges present at the start (hill climbing), or all possible edges present (k-learning). All nodes are colored yellow and labeled with their names, and all edges are of equal thickness. Users can manually draw and remove edges or apply edits from a table of suggestions generated from a user-selected algorithm (x, y, or z). Every action is undo-able. Besides the animation of manually drawing an edge, the graph remains static.

Description

Our Evaluation View differs from OpenMarkovs default design in two main ways: the edge display and the node display.

In Evaluation View edge thickness scales with a value called **relative link strength**, which allows us to evaluate the relationships between variables.

Ebert-Uphof defines a similar quantity, link strength, as a measure of information gain defined by entropy [6]. The link strength of a directed edge from X to Y measures the information gain about the random variable Y given the value of X, conditioned on the parents Z of Y excluding X [6]. That is:

$$LS(X \to Y) = H(Y|Z) - H(Y|X,Z)$$

Unlike in [3], we choose to normalize this metric to ensure equivalent comparsions in the visualization. The link strength percentage in [6] regularizes the value by the initial entropy, i.e)

$$LS\%(X \to Y) = \frac{H(Y|Z) - H(Y|X,Z)}{H(Y|Z)}$$

This metric effectively tells us how much more we can learn about a variable given its parent. However, it can be misleading to use this metric for edge thickness because the normalization doesn't allow for visual comparison. Instead we define the normalized link strength to be normalized over the mutal information of all edges:

$$LS_{norm}(X \to Y) = \frac{LS(X \to Y)}{\sum_{(A,B)\in E} LS(A \to B)}$$

This quantity is recalculated with every edit to the graph, and the thickness dynamically changes appropriately. That is, when a user manually draws an edge or adds one from the edit table the dependence of the new edge and every other existing edge is recalculated, and each edge is repainted with a thickness corresponding to the new value [Figure 1].

Nodes appear slightly altered in Evaluation View, as well. Nodes are still yellow and arranged in a circle by default, but we vary the alpha channel. For each node we consider every possible edit from that node: all edge additions, removals, or inversions. We then calculate the **motivation** for each of these possible edits. Motivation is a measure of the change in heuristic score for the whole network. At the start of the interactive learning process in OpenMarkov, the user specifies which metric or heuristic to use to evaluate the total network. Motivation of a particular edit is the amount the total heuristic score will change when that edit is applied to the network. The edit with the highest motivation is the best edit a user can make from that specific node. The transparency of the node directly corresponds to this maximum motivation; higher motivation values yield higher opacity and lower motivation leads to more transparency. We further added a label that redundantly encodes the value of the maximum motivation and the total number of possible edits from that node [Figure 2]. Since this is a relative measure, once there are no more possible edits all nodes return to the original yellow color.

In conclusion, we visually encode two additional dimensions: the strength of each link, or the relative link strength, and the heuristic of the best potential edit from each node.

Purpose

Evaluation View has two main goals. First, the dynamically changing edge thickness immediately provides the user with a notion of how well the current Bayesian network is performing. Secondly, the varied node transparency and labels inspires user action. The opaque nodes and the nodes with many potential edits will draw the user to click on them, switching them into the Edit View for further exploration and editing. The Edit View is described in detail in the following section.

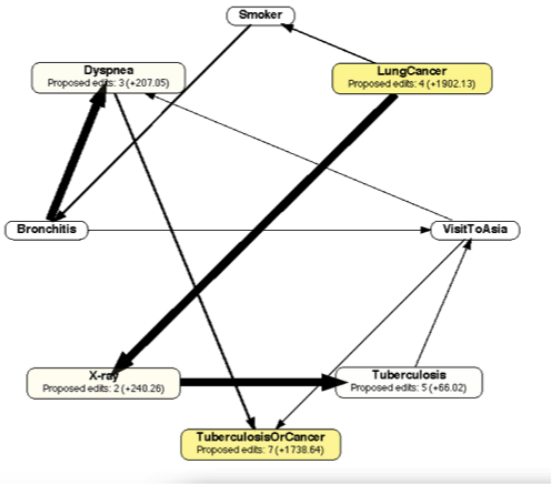

Figure 1. Edge thickness represents the relative link strength of each link, a measure of the correlation between the parent and child variables. It scales dynamically with each edit to the network.

Figure 2. Nodes have new labels. Proposed edits is the total number of edits from that node with a positive motivation. The value in parentheses is the motivation of the best edit.

The Evaluation View is meant to be used consistently throughout the construction of the network, between every change to the graph. As it is the default view, it requires no clicks to display, and the user can immediately evaluate each edit. We maintain the highly useful undo-able property of each edit, so the user may make an edit, evaluate the change, undo the change, and attempt a different edit instead.

Design Considerations

We decided to encode link strength with edge thickness for consistency with past implementations of interactive structural learning. Edge thickness has been used in prior works to encode strength of links [3,6].

We considered many different visual dimensions to encode best possible edit from each node before settling on transparency, such as node size and color. These methods, however, do not necessarily galvanize action on the part of the user. The relative areas of nodes may prompt users to click larger ones, however area is difficult for humans to perceive, especially with irregular shapes such as the elliptical nodes, and if the differences between node sizes are small. We considered using multiple color maps to prompt different levels of action, but such coloring is more suited to nominal, or categorical data where boundaries are more well defined. Varying transparency allows us to maintain consistency with the tool's design and also allows displaying small differences relatively well. To address the difficulty in comparing similar levels

of transparencies, we redundantly encode the max motivation with a node label [Figure 2].

Lastly, before our augmentation, the OpenMarkov tool focuses the users attention on a table of possible edits for the graph, creating an edge-driven construction process. We instead wanted the visualization to drive user action, intentionally inspiring a more natural variable driven exploratory process. The Evaluation View intrigues users to explore edit possibilities by selecting nodes. We believe this forces users to consider the variables themselves, and its possible relationships with the surrounding variables. In focusing on a single variable at a time, it can be more effective in learning networks with a larger number of variables where considering all of the combinations of edges at once could be overwhelming. Instead, the node-centric approach can help focus the user's edits to a smaller subset of potential graphs, all while the evaluation view promotes a holistic thinking about the structure of the Bayes net.

Edit View

Without our added GUI features, a user can edit a Bayes net in OpenMarkov in two ways: 1) manually drawing or removing edges using the toolboar, and 2) selecting an edit from the proposed edits table and clicking the apply button. Manual edits are created solely using expert knowledge. Suggested edits in the proposed edits table are generated as a part of the learning algorithm and are ordered by decreasing motivation; no visual cues exist. We want to encode information visually to guide the user in selecting edits.

Description

The user is in Edit View as soon as they select a node. All existing edges in the network that are unrelated to the selected node are displayed in light grey with a uniform thickness. All potential new edges are displayed in green. Existing edges touching the selected node are either red or blue to indicate a removal or a reversal of the edge, depending on which yields a higher motivation. All of the green, red, or blue edges are also displayed with a uniform thickness. A legend is displayed next to the network to map the colors, red, blue, and green, to the specific type of edit, remove, invert, and add [Figure 3].

Purpose

The Edit View is meant to be used when the user seeks to modify the topology of graph. Rather than using a list of suggested edits, the visualization helps guide a user to explore possible edits interactively by selecting and deselecting different nodes. The Edit View allows users to better combine their domain knowledge with algorithmic knowledge than in other structure learning methods. Firstly, users see indicators of graph performance before entering the edit view allowing them to make more informed changes. Furthermore, the node-centric interaction allows experts to focus on a smaller more manageable subset of the graph, with a maximum of $|V|$ relationships at any given time as opposed to $|V|^2$. The Edit View helps makes the interactive structure learning process more user-driven rather than algorithm-driven.

Design Considerations

This view required us to make the most design decisions. First, we decided existing edges in the network unrelated to the selected node must remain on the graph to keep the user oriented. We also chose to shrink them all from varying thickness in the Evaluation View to a uniform thickness and we change stroke color from black in normal state to gray. This decision was mostly to reduce graph clutter because of the many dimensions of the new potential edges we want to user to emphasize on.

We determined color is the best attribute to display whether each new potential edit was an add, remove, or invert. The tradeoff of requiring a small legend is worthwhile if the colors provide clear, recognizable distinction between the new edges that appear in the Edit View. We provide a legend in lower right of our editor panel, which annotates color for different types of edit proposals.

Look-ahead Graph

Unlike previous interactive structure learning tools, we implement a look-ahead preview functionality. To view a batch of changes introduced by the learning algorithm in OpenMarkov, the user clicks the Complete Phase button to automatically complete the Bayes net construction. However, such functionality doesn't allow the user to keep track of what changes are introduced into the network which can be cumbersome when dealing with larger networks.

Description

We created functionality that allows the user to preview the next set of changes introduced by the structure learning algorithm. Along the top toolbar we added an open input field and three buttons. The open field can take any integer as an input to specify the number of steps to look-ahead. The three buttons are labeled Look Ahead, Reset, and Apply Edits [Figure 4]. If the user fills in the open field with an integer, k greater than 0, and clicks the Look Ahead button, the software will preview the next k positive edits that can be applied to the network. To discrimenate these proposed changes from edges within the network, we display them with a uniform thickness and with a different color. The new graph, including the grey overlaid edges, represents the network after applying the k next most optimal positive steps, according to the algorithm [Figure 5]. In the event that there are fewer then k, positive edits, the look-ahead will only display those changes that would yield improvement to the objective function.

After clicking Look Ahead, the user can choose either Undo or Apply Edits. Undo removes the newly overlaid grey edges. Apply Edits turns the newly overlaid grey edges to black, and the edges re-scale according to their independence, becoming permanent edges in the network. The view then switches back to the Evaluation view where the user can analyze the strength of the newly added connections.

Figure 4. The Look-ahead buttons added to the OpenMarkov toolbar

Purpose

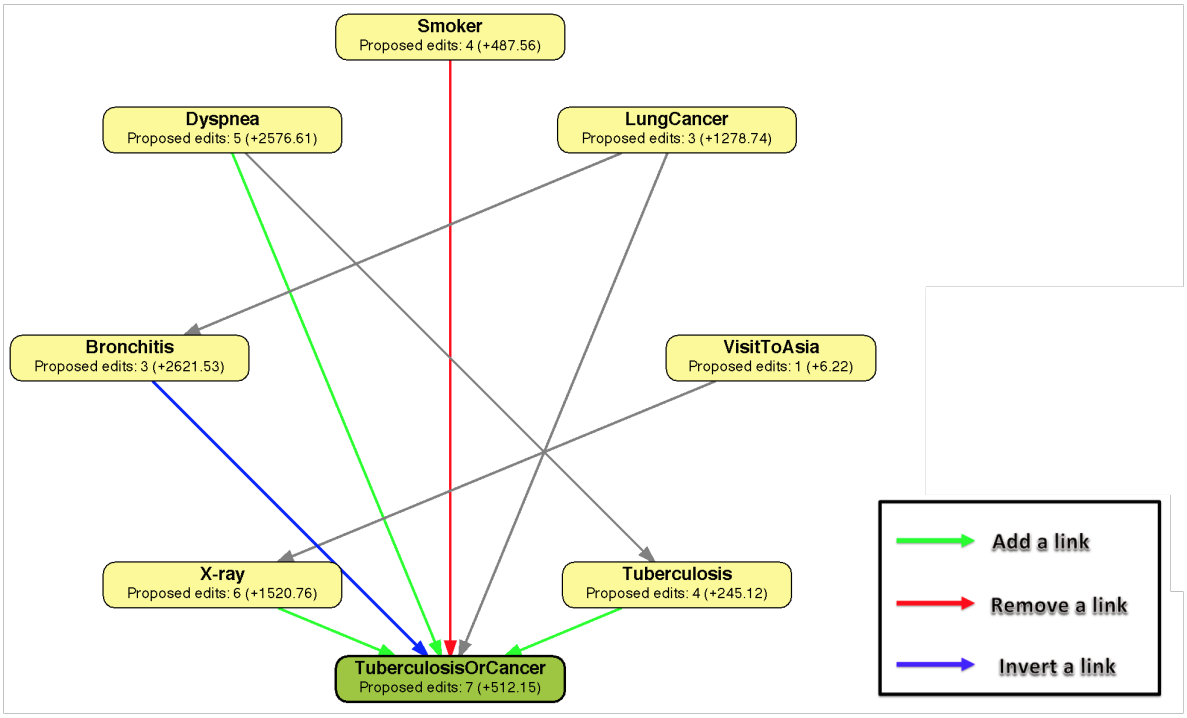

Figure 3. The Edit View: Here a user has selected "TuberculosisorCancer" and the potential edits in green, blue, and red corresponding respectively to adding, inverting, or removing the edge.

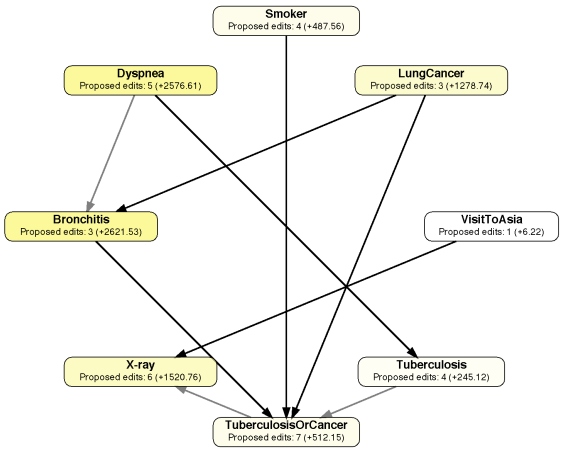

Figure 5. Look-ahead Graph: Users select a number of steps to look-ahead, and changes appear on the graph in grey. These edges become a permanent part of the graph and turn to black when the user selects the "Apply Edits" button in the toolbar.

The Look-ahead graph is for users to preview the algorithms k next proposed edits without all of the extra data provided in the Edit View. Throughout the Bayes net construction process the user will continually select nodes, inspect the next potential edits, apply edits, explore how the strength of edge changes, undo steps, and redo steps.

This is a user driven process and differs vastly from traditional structural learning approaches in which the user and algorithm take turns applying changes to the network. The look-ahead functionality thus allows this type of interchange between the user and the algorithm to persist, while giving more control over the changes introduced by the learning algorithm. Nonetheless, we expect that users will utilize the Look-ahead funcionality less frequently than the Evaluation and Edit Modes.

The grey edges that appear after clicking Look Ahead are meant to seem temporary, like overlays. Only when the user selects Apply Edits, and the new edges turn black are they truly part of the network.

This feature is meant to better integrate human interaction with machine generated suggestions. OpenMarkov previously only displayed look-ahead steps as text entries in a table. Users could not preview graphs without applying edits. The look-ahead graph provides new flexibility.

Design Considerations

Our first design decision for this view was to overlay the look-ahead graph on top of the current network. We considered a side-by-side view of the current Bayes net and the look-ahead graph, or a pop up window, but we wanted to avoid limiting or occluding the available workspace. A side-by-side view cuts a user's workspace in half, and a large pop-up would cover

the current workspace and would be disconnected from the other interactions.

We also discussed at length the use of an open input box and three buttons. In order to maintain the feeling that the look-ahead graph is a temporary overlay until intentionally applying the edits, it is important to maintain two separate Look Ahead and Apply Edits buttons. We briefly considered removing the Look Ahead and Undo buttons completely, and instead utilizing only an input box and an Apply Edits button. In this scenario a user would type an integer and click the single button; to remove the look-ahead edits, a 0 would be entered. All these options are unfit for our GUI because we want to emphasize that the look-ahead edges are temporary. We also considered using a slider bar instead of an input box for user to enter look ahead steps. We finally decided on using input box because number of steps look ahead could vary greatly depending on number of nodes in the graph and current graph state.

Lastly we considered keeping different edge thickness for all existing edges. Finally, we decide on thinning all existing edges, such that existing edges and look ahead preview edges in the network have a uniform thickness. The main reason for keeping a uniform thickness for all edges in look ahead graph is we want to provide user a general overview of the network k steps from current graph state. After testing both scenarios, we find varying edge thicknesses make the graph look cluttered and provides misleading information to draw user's attention to a subset of the grpah instead of the whole picture; it helps enforce the distinction between the look ahead preview and the existing network edges, and contrary to our hypotheses was not visually overwhelming.

Objective Function View

One of the many problems with approximate Bayesain structure learning is the overfitting of the network to the dataset. When working with a system is not fully observed, the choice of metric will strongly influence the resulting structure that is learned. Current structural learning tools limit the user to work with a single metric on which the network structure is optimized. This can lead to overfitting of the training dataset. The objective function view allows users to compare the metric used in training with the classification rate on a held out set to monitor overfitting.

Description

OpenMarkov allows users to load in datasets for performing inference under *File -¿ Load Evidence*. We leverage this functionality during training to allow users to benchmark the network's classification accuracy on a held-out set while training. After loading in this testing data, the user can select which variable to classify over in the same window as the table of suggested edits, by clicking on a tab at the top labeled [Figure 6].

With each change to the network, the software performs inference on the dataset. For each piece of evidence in the dataset, it computes the probability that the test variable matches the entry. Only samples for which the test variables is know is the test performed. If the probability exceeds the user specified threshold, it considers it as a correct classification.

Each change to the network yields a different classification rate as well as a differnt change to the network score used by hte learning algorithm. These values are plotted in the window. To be able to display both measurements simulataneously, we use mutliple axes of differnet scales and a legend to differentiate between the two plots.

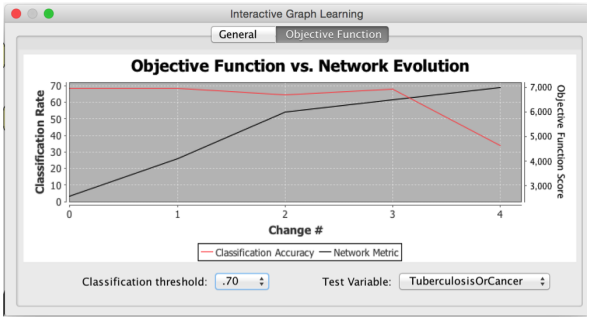

Figure 6. The Objective Function view. Users can select the test variable, the classification threshold, and examine the results with each edit to the network.

Purpose

This feature is meant for users to monitor the performance of the network on a sparse test dataset. The time-series graph shows whether the performance increases or decreases. While the learning algorithm metric will continue to yield positive edits, such edits may be overfittign to the dataset. This view allows you to monitor this by using a different dataset and a classification analysis.

Design Considerations

In designing a tool for interactive structure learning, we knew that overfitting was an issue that we would need to address. Our original design featured a dashboard of metrics depicting the current score of the network. However, we felt that being able to track the changes would be useful to understand where overfitting occured and perform the necessary undoes to address the issue.

After building the time series chart, the original plan was to add a dropdown that would allow the user to display which metric to plot. However, we found it more effective to visualize the metrics simultaneously with mutliple axes to allow for easier comparison.

RESULTS

Results of our research project are based on feedback from a limited number of human subjects. We asked a small group of students, staff, and faculty try use our tools with OpenMarkov and give us informal oral feedback. All subjects had a basic understanding of Bayesian networks and related concepts of conditional probability. For each subject, we first gave a review of related concepts of Bayesian networks and then we gave a quick demo of the basic functionalities of our software. We chose to test all subjects with a Asia10k dataset, which contains 10 variables of common diseases and their potential causes. We choose this dataset for two reasons: 1) The

search space of the set of graphs is adequately large enough to justify learning. 2) Users are able to have some insights about the network based on common knowledge. We observe users' interaction with our software and ask user to compare their experiences of using the original OpenMarkov framework versus the our tool.

Evaluation View

We received overwhelming positive feedback about the evaluation view. Users reported it is a necessary improvement to the existing OpenMarkov framework, and that the edge thickness is an intuitive visual encoding of edge strength. Most other feedback regarded a misunderstanding of the true meaning of link-strength and how to compare the different edge thicknesses. Perhaps this should be better communicated through the OpenMarkov GUI.

Edit View

Almost all of our users commented with dissatisfaction about the colorful potential new edits. With the help of the legend users easily understood the meaning of the color-coded edges, however they expressed desire for an obvious visual dimension displaying the motivation of each potential edit. A few subjects suggested we re-use edge thickness. Although edge thickness already encodes a related but different value in the Evaluation View, the edge strength or dependence, edge thickness here would represent motivation of a new edit, or the relative performance improvement of the entire network. Despite this discrepancy our user subjects believed it would not be misleading to encode both of these values with the same visual dimension.

Other feedback for the Edit View simply commented on the quantity of incoming edges to each selected node, and the lack of outgoing edges. Although this may be a bug in our implementation, it is more likely a result of certain edits resulting in negative motivation scores; that is, they would decrease the overall performance of the network. Future work could include a way to display these negative edits, or making it clear to users that negative edits do not appear in the Edit View.

Furthermore, we could improve the edit view by allowing users to directly click edges to apply changes rather than having the user exit the Edit view and use the tool bar.

Look-ahead Graph

Subjects generally found the look-ahead graph intuitive and informative. We specifically asked opinions on the text box and three buttons, and no users reported that any of the three buttons was superfluous. We received mixed feedback about the edge thicknesses. Some users preferred that we keep them varied with dependence values, our current implementation. Some users, however, reported that they would prefer the look-ahead graph to be a cleaner preview of a finished product, with all edges having uniform thickness.

Objective Function View

Users responded most enthusiastically to the Objective Function View. It is the most innovative and computationally advanced of our features. With some explanation subjects understood the purpose of the graph, and could easily use it to identify the point of overfitting in the process. Most other feedback focused on a desire to better understand the behind-the-scenes computation.

DISCUSSION

Based on our feedback we determined that dynamic visualization of metrics, or evaluation of each edge performance, is the most important visual encoding we created for OpenMarkov. Not only was our Evaluation View well received, it was intuitive for users and required little to no explanation. This feature is extremely helpful during network construction process because it provides user immediate visual feedbacks of current and future actions based on current graph state. The next most useful feature for this domain is providing tools to detect overfitting. Although our Objective Function view cannot prevent this problem, it offers consistent monitoring and immediate detection, helping to avoid the most significant problem with automatic computation of Bayes net structures.

More research needs to be completed to evaluate whether the Edit View truly inspires a node-drive approach to building Bayes nets, and whether this re-orientation of the process helps improve users comprehensive understanding of the networks. More feedback is also required to determine the utility of a preview function, such as the look-ahead graph.

FUTURE WORK

There is still much to be completed in the field of improving visualization in interactive Bayes net software systems. There are a number of worthwhile improvements and extensions of our current project, as well as implications for the greater field of probabilistic graphical models (PGMs).

Our current implementation only utilizes one construction algorithm, hill-climbing, and a limited number of objective functions to evaluate the performance of the network. Future work would extend functionality to other algorithms and heuristics. More formal user studies need to be completed to better evaluate the UI of our tool, and compare it with OpenMarkov without our additions as well as other interactive structure learning software. These tests should include gathering feedback from subjects who routinely utilize Bayesian networks in their field of work.

From the feedback we recieved from our user study, we note that an area of future work includes expanding up the visualizations and interactions of the Edit View. Firstly, we suggest methods to to help differentiate between the motivations of proposed edits. We reviewed several domains in which we could encode such information. We advise against reusing edge thickness, because we use that dimension in the Evaluation View to encode link strength; and it could be misleading to use the same attribute to encode motivation, or difference in performance of the network as a whole in tightly coupled views. We also considered using transparency, however it was

hard to detect with the thin shape of the links. We suggest exploring text based tooltips or other visualizations to encode the proposed edits' motivation.

Further work on this problem could be extended to the construction process of other graphical models, such as Hidden Markov Models. We believe the success of our OpenMarkov modifications implies that visualization and interaction should play important roles in all fields that necessarily combine expert domain knowledge and computational tools. Quality interaction and visualization will be key to integrating these two as computers and technology are now defining parts data analysis and problem solving.

REFERENCES

[1] D. Klein. CS 188. Class Lecture, Topic: Bayes Nets: Representation. Department of Electrical Engineering and Computer Science, Universtify of California Berkeley. Berkeley, CA, November, 2013.

[2] CISIAD. OpenMarkov Tutorial. Internet: `http://www.openmarkov.org/docs/tutorial/tutorial.html`, Sept. 2, 2013 [Nov. 1, 2041].

[3] L. E. Sucar and M. Martinez-Arroyo. Interactive Structural Learning of Bayesian Networks. Expert Systems With Applications, vol 15, pp. 325-332, 1998.

[4] S. Acid, et al. A Comparison of learning algorithms for Bayesian networks: a case study based on data from an emergency medical service. Artificial Intelligence in Medicine, vol. 30, pp. 215-232, 2004.

[5] D. Heckerman. A Tutorial on Learning with Bayesian Networks. Microsoft Research Advanced Technology Division, Microsoft Corp, Redmond, WA. Rep. MSR-TR-95-06, Nov. 1996.

[6] I. Ebert-Uphoff. Measuring Connection Strengths and Link Strengths in Bayesian Networks. Robotics and Intelligent Machines Center, Georgia Ins. Tech., Atlanta, GA. Rep. GT-IIC-07-01, Jan. 29, 2007.

[7] M. Koivisto and K. Sood. Exact Bayesian Structure Discovery in Bayesian Networks. Journal of Machine Learning Research, vol. 5, pp. 549-573, 2004.

[8] Murphy. How to use the Bayes Net Toolbox. Internet: `http://bnt.googlecode.com/svn/trunk/docs/usage.html`, Oct. 29, 2007 [Nov. 15, 2014].

[9] J. Pearl and T.S. Verma. A Theory of inferred causality. Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, pp. 441-452, 1991.

[10] Spirtes, et al. The PC Algorithm, in Causation, Prediction, and Search, Cambridge, MA: The MIT Press, 2001, ch. 5, sec. 4.2, pp. 116-124.

[11] W. Buntine, Learning classification trees, Stat Comput, vol. 2, no. 2, pp. 6373, Jun. 1992.

[12] G. F. Cooper and E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach Learn, vol. 9, no. 4, pp. 309347, Oct. 1992.

[13] A. J. Hartemink. Banjo: Bayesian Network Inference with Java Objects. Internet: `http://www.cs.duke.edu/~amink/software/banjo/`, Nov. 2, 2010 [Dec. 5, 2014].

[14] A. Moore and W. Wong. AutonLab: Bayes Net Learner. Internet: `http://www.autonlab.org/autonweb/10530`, 2010 [Dec. 5, 2014].

[15] I. Ebert-Uphoff. Linkstrength Packet for BNT. Internet: `http://www.dataonstage.com/BNT/PACKAGES/LinkStrength/`, Nov. 2011 [Nov. 3, 2014].