Creating, Visualizing, and Exploring Knowledge Maps

Colorado Reed
U.C. Berkeley
colorado@berkeley.edu

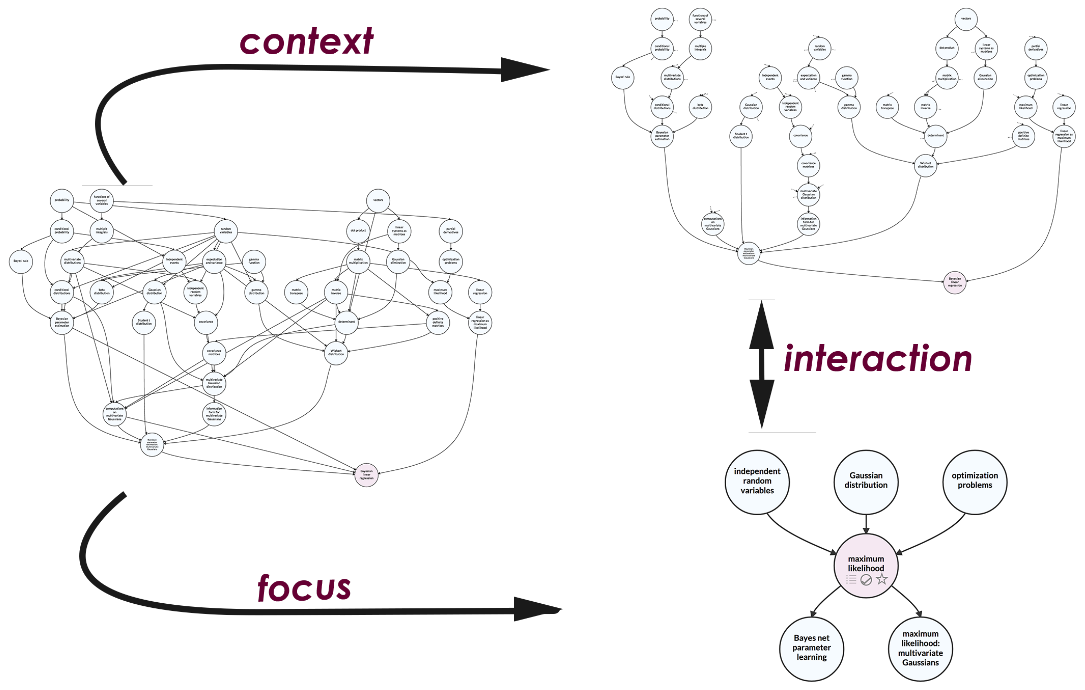

context
interaction
focus
independent random variables
Gaussian distribution
optimization problems
maximum likelihood
Bayes net parameter learning
maximum likelihood: multivariate Gaussians

Figure 1. The knowledge map visualization and exploration system described in this paper reduces the number of visual edges for large knowledge maps. In addition, it provides a number of interactive components to help users explore and understand these maps. The left figure shows a knowledge map (layered, Sugiyama-style, directed graph) of machine learning concepts. The large number of crisscrossing edges makes the graph difficult to visually parse. The top right figure shows the same graph produced by our system. Notice that the node locations are the same, but a large number of visible edges have been replaced with "wisp edges:" small, dashed protruding lines from the source and target nodes that become full edges when the user hovers on the wisp or the source/target node. The bottom right figure shows the focused mode that is displayed after clicking on a concept. The focused mode shows the inlinks and outlinks of the targeted node and the nodes associated with these edges.

ABSTRACT

Knowledge maps are cognitive aids designed to help users explore a body of knowledge in a nonlinear way. Specifically, a knowledge map is a graphical representation of knowledge chunks, e.g. single-digit addition, linked together by directed edges that express prerequisite relationships. In this paper we describe a novel knowledge map creation, visualization, and exploration system. We employ edge reduction and interactive techniques to aid users in exploring large knowledge maps. Furthermore, we describe a visual knowledge map editor that allows users to create new knowledge maps, associate related content with the map (e.g. textbook references of video lectures), and make local changes to large preexisting knowledge maps.

INTRODUCTION

A *knowledge map* is a graphical representation of concepts that reveals the relationships and dependencies among the concepts. In this work, a knowledge map is a directed graph in which the nodes are concepts and the edges specify prerequisite relationships between the concepts, see Figure 1. This graphical representation is a variant of the *concept maps* common in educational psychology, whereby the nodes are concepts and the edges are linking words that create a meaningful statement or proposition when combined with the nodes. Stated another way: a knowledge map primarily serves as a reference tool while a concept map serves as a learning tool. That is, a knowledge map typically has associated content with each concept, e.g. textbook references or video lecture, that a student would use to learn the material, while a concept map summarizes a body of knowledge and is oftentimes created by the student to help learn the material.

In recent years, several online education companies and schools have created knowledge maps to help students visualize and navigate their content in a nonlinear way, see e.g. Khan Academy [3], Duo Lingo [2], and Metacademy[4]. Khan Academy, a nonprofit online educational company with over six million unique monthly visitors, has a large (700+ node) knowledge map that founder Salman Khan discussed on p. 52 of [13]:

> [The knowledge map] became a core piece of the Khan Academy software platform. In stressing the connections among subjects and giving learners a visual picture of where they've been and where they're going, we hope to encourage students to follow their own path—to move actively up, down, and sideways, wherever their imaginations lead.

A problem with knowledge maps, however, is that the large number of relationships between the concepts produces a large number of intertwined and crisscrossing edges in the graph. In turn, it is difficult to interpret the conceptual relationships, see Figure 2 for examples from Khan Academy and Metacademy. These "hairball" graphs are common in network visualizations, and indeed, there is a deep field of research focused on visualizing networks, e.g. see [7]. The hierarchical placements of the nodes and edges in knowledge maps express the dependency structure within the body of knowledge, and for this reason, many network visualization techniques will not work with knowledge maps as they obscure or break these relationships.

In this work, we create a novel interactive system for creating, visualizing and exploring knowledge maps. On the creation side, we constructed a graphical user interface (GUI) for visually specifying concepts, their relationships, and associated content. On the visualization and exploration side, our system uses a layered (Sugiyama) directed graph placement algorithm [17] to initially place the nodes, a topological sort to determine an initial set of visible edges (the topological sort maintains the connectivity of the graph), an edge length threshold to determine which remaining edges should be shown, and edge "wisps" to indicate the absence of an edge. These techniques greatly reduce the number of visible edges in large knowledge graphs while maintaining the hierarchical and connectivity structure of the content.

To further accommodate the goals of knowledge map users, we incorporated several interactive components that aid the user in locating concepts of interest and understanding their relationships to other concepts. In particular, our system allows users to step through the dependency structure of a graph by sequentially visualizing contextual views of the concepts (the direct dependencies and outlinks). Furthermore, we include a number of hover and click operations that reveal the reason for the relationships and associated content present for a given concept.

RELATED WORK

In this section we discuss three areas of previous research and development relevant to this paper: online knowledge graph systems, visualizations of large directed graphs, and tools for creating knowledge maps.

Online Knowledge Graphs

The knowledge map creation, visualization, and exploration system described in this paper was designed for Metacademy—an open source platform for collaboratively creating, refining, and sharing knowledge maps and associated learning material [4]. Figure 2 (right) shows a Metacademy knowledge graph: one of the two main components of Metacademy. The other major component is a text-based view that provides the user with the following information for each concept: a description of the concept, goals for the learner, the prerequisites of the concept, learning resources for the concept, and related (non-dependency) concepts. Metacademy is typically used as a targeted learning resource, whereby a user searches for a particular concept and is then presented with a knowledge map with that concept as the sole leaf—most graphs have 5-50 nodes and less than 100 edges.

As mentioned in the introduction, Khan Academy [3]—a free video lecture and exercise repository—also provides a knowledge map of their material. Rather than showing target-based components of the graph, Khan Academy shows users their entire 700+ node knowledge map. This knowledge map is much larger than any of Metacademy's knowledge maps, but the techniques discussed in this work are applicable to their graph, and we will format their content for our system in future work.

Visualization and Exploration of Directed Graphs

Our knowledge map visualization and exploration system draws on a number contributions from the graph visualization community. In particular, the authors of [18] developed an interactive visualization system for large networks, e.g. three million nodes. The authors focused on providing context-specific visualizations of large directed graphs and stated that:

> Although providing a structural overview of the graph is a laudable goal, there are many cases where the user is simply not interested in a global view of the whole graph, but wants to solve a particular concrete task on the graph instead.

Rather than show all direct inlinks and outlinks as is done in several related systems [5, 6], the authors devised a degree of interest (DOI) term for each node that they use to determine which subset of contextual nodes to show the user. Since individual nodes in knowledge maps have a relatively small degree, it is typically possible to visualize the full context of each node rather than a subset, and we do not use the DOI technique in this work. The authors use wisp edges—truncated inlinks/outlinks to a node—to indicate the presence of an edge without cluttering the visualization with long, crisscrossing edges. We incorporate this technique into our visualization (see the next section).

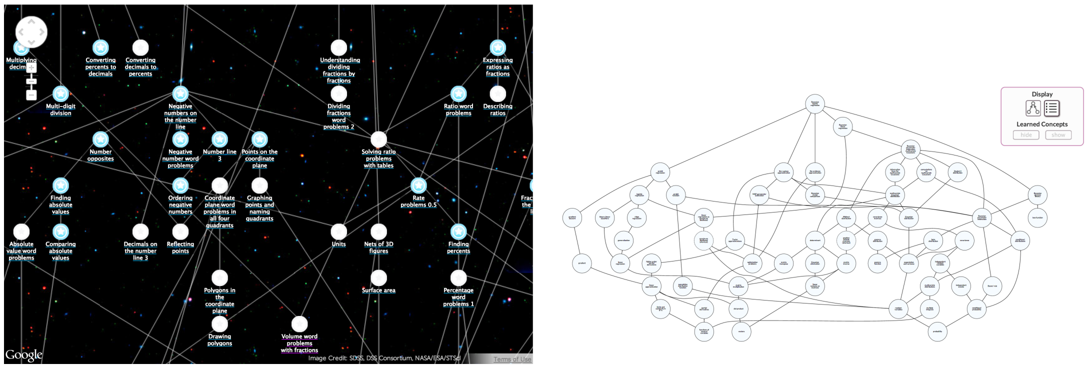

Multiplying decim
Converting percents to decimals
Converting decimals to percents
Understanding dividing fractions by fractions
Expressing ratios as fractions
Multi-digit division
Negative numbers on the number line
Dividing fractions word problems 2
Ratio word problems
Describing ratios
Number opposites
Negative number word problems
Number line 3
Points on the coordinate plane
Solving ratio problems with tables
Finding absolute values
Ordering negative numbers
Coordinate plane word problems in all four quadrants
Graphing points and naming quadrants
Rate problems 0.5
Absolute value word problems
Comparing absolute values
Decimals on the number line 3
Reflecting points
Units
Nets of 3D figures
Finding percents
Polygons in the coordinate plane
Surface area
Percentage word problems 1
Drawing polygons
Volume word problems with fractions
Google
Image Credit: SDSS, DSS Consortium, NASA/ESA/STScI
Display
Learned Concepts
hide
show

Figure 2. The left figure shows a section of the Khan Academy knowledge map; the right figure shows a Metacademy knowledge map.

There are a number of software packages and algorithms for visualizing (large) directed graphs that inspired our system:

- the dot placement algorithm, described in [12], is the underlying placement algorithm for the popular graphviz C++ graph visualization software. The dot algorithm is used in the dagre JavaScript graph placement library, which is used in this work (see the next item).
- dagre [1], a JavaScript library for directed graph placement is used to generate node and edge placements in our knowledge map system.
- Dynasty [11], a system for browsing large directed graphs where only a small portion of the graph is visible at any given time. Dynasty inspired the "focused mode" aspect of our system where a concept and its dependencies and outlinks are visible at a given time.
- the transition animations in Open Graph Drawing framework [10], a large C++ library of algorithms for graph drawing, inspired the transition animations, sequence, and timing used in this work.

Furthermore, Misue et al.'s mental map preservation properties for directed graphs [15] motivated our use of visualization modes rather than dynamic directed graphs.

Finally, many directed graph visualization systems transform the data for visualization. For instance, hive plots [14] produce a linear layout for grouping nodes by type and then arranges the nodes along a radial axis. Arranging knowledge map concepts along a radial axis does not work well for a visualization as the concepts commonly have relationships that span multiple levels, and as a result, the radial network becomes polluted with edges that span most of the circumference. Strictly hiding these types of dependencies masks the prerequisite complexity for various concepts and could deceive the user.

Knowledge Map Creation

As of the time of this writing, the author currently does not know of any tools that are explicitly used for creating knowledge maps as defined in this paper. Both Khan Academy and Metacademy use text-based systems, e.g. html forms, to specify the relationships in the graph. However, there is a large number of tools for creating similar structures such as diagrams, mind maps, and concept maps. We briefly discuss these tools and their relevance to our system.

Cmap tools [9] is a Java-based GUI for creating concept maps: directed graphs where the nodes are concepts and the edges are linking words so that tracing through a path is akin to reading a sentence. The Cmap tools GUI inspired our editor as it has simple drag and click operations to construct the graph. The Cmap tool editor, however, does not have associated content with the graph and is not browser-based. Furthermore, the Cmap tools editor requires the user to manually place the nodes and edges. Our editor operates in a similar fashion, but users have the ability to algorithmically arrange the nodes and edges using the dagre javascript library. Finally, our editor is browser-based while Cmap tools is a Java applet.

There are countless diagramming tools such as gliffy[1], draw.io[2], and dia [3]. Many of these of tools are polished, intuitive, and downright beautiful diagramming tools, but they are more general purpose than what is needed for knowledge map creation. Letting users choose the color and shape of the nodes, the thickness of the edges, etc, detracts from the main purpose of specifying the concept relationships and providing associated content.

METHODOLOGY

In this section we describe the visualization and interaction methods employed by our knowledge map system. In particular, our system builds upon the Metacademy knowledge map system, and we have recently acquired the data for Khan Academy's knowledge map and plan to extend our system for this data in the near future.

Knowledge Map Creation and Exploration

From an informal usability study conducted with seven U.C. Berkeley students using the think-aloud protocol[16] and the Metacademy knowledge map system, we identified the following common objectives the students had when interacting with the knowledge map:[4]

[1]`http://www.gliffy.com/index-g.php`

[2]`http://www.draw.io`

[3]`http://sourceforge.net/projects/dia-installer/`

[4]Specifically, we observed the students interacting with relatively large (30-50 nodes) and small (circa 10 nodes) knowledge maps

1. visualize the overall complexity of the targeted concept, i.e. the depth and breadth of the graph and the number of edges and nodes
2. locate specific concepts within the knowledge map
3. identify the prerequisites and outlinks for a concept, the prerequisites of its prerequisites, outlinks of its outlinks, etc
4. focus on a particular concept
5. explore related concepts not shown on the current graph (inclusion of new data)

Metacademy's primary form of interaction is allowing users to mark concepts they have learned and subsequently remove them from the graph. When removing the concepts, the Metacademy system naively recalculates the graph layout without taking into account the previous layout. The system then instantaneously flashes to the new knowledge graph without showing the user the corresponding transitions, see Figure 3. This dynamic form of interaction does not directly address any of the objectives expressed by the students in our usability test. Furthermore, according to Misue et al.'s theory of mental map preservation for dynamic directed graphs [15], this transition destroys the user's mental map of the system as it violates the following three mental-map-preservation properties:

1. orthogonal ordering: maintaining the relative directions between the nodes (these updates can shift the relative directions see e.g. the "random variables" and "expectation and variance" nodes in Figure 3)
2. proximity model: maintaining the relative distance of the nodes (these updates can change the relative distance of the nodes, see e.g. the "random variables" and "expectation and variance" nodes in Figure 3)
3. topology model: graphical objects within a relative region should stay within that region (these updates can change the regional placement of nodes: this is not present in the supplied example, but it tends to occur with larger graphs).

While letting users remove nodes and edges associated with learned concepts helps reduce the number of objects in the graph, this form of interaction does not address the identified objectives of the users and comes at the cost of nullifying the user's previous mental map. From these observations, we did not include this functionality within our system.

To address the visualization and exploration objectives above, we created two modes within our system: (1) an *overview* mode in which the entire graph is shown and (2) a *focused* mode in which a particular concept is shown along with its immediate dependencies and outlinks. The goal of the overview mode is to allow users to visualize the overall complexity of the targeted concept as well as quickly trace the paths of various concepts (objectives 1 and 3), as well as quickly find specific concepts within the knowledge map and explore related concepts by loading them into the current graph (objectives 2 and 5). The focused mode, inspired by the focused graph view from [11], address objective 4. In the following paragraphs we describe these modes in greater detail.

Both Metacademy and Khan Academy knowledge maps have a large number of long, crisscrossing edges that make it difficult to trace the paths within the knowledge map. We addressed this problem in our system by employing the following algorithm to reduce the number of visible long, crisscrossing edges in our graph:

```
1. remove all transitive edges from the graph
2. compute the graph layout using dagre
3. set visible-outlinks ← $\emptyset$
4. for each node $n$ in the graph:
     set visible-outlinks ← visible-outlinks
         $\cup$ {outlinks($n$) with path length $< L$}
         $\cup$ shortest-outlink($n$)
5. for all edges $e$ not in visible-outlinks: display an edge wisp
   protruding from the source and target of $e$
```

This algorithm is used to help accurately convey the relationships between the concepts while reducing the edge saturation within the graph. The motivating philosophy behind this technique is to show users edges when they are interested in seeing them, while still providing enough edges to maintain the proper structure of the graph.

In step 1, we remove the transitive edges in the graph because they do not affect the relative prerequisite structure of the graph. That is, if we have edges (A,B), (B,C), and (A,C), then removing (A,C) from the graph will not affect the A,B,C ordering, and from a pedagogical perspective, it is a redundant edge. In step 2 we compute the edge and node placements using dagre, though any graph placement algorithm can be used during this step.

In step 4 we remove all edges exceeding a length threshold L while ensuring that each node has at least one outlink (the shortest outlink). By including the shortest outlink for each node, we maintain the visual connectivity of the graph after hiding other long edges. Naively truncating all edges could induce artificially separated clusters of concepts that would be determined by the graph placement algorithm rather than the content. We note that with a small enough threshold L, each node would have one outlink and the graph would become a tree. Therefore, if we simply did not include the non-visible edges in the graph placement algorithm, it would be possible to obtain a tree layout with zero edge crossings. This tree layout, however, could visually invalidate the relative prerequisite structure of the graph. Using the previous example with nodes A, B, and C, we could have a tree with visible edges (B,C) and (A,C) and invisible edge (B,C), which would result in visually placing nodes A and B on the same level.

and associated content for 30-60 minutes while verbalizing their thoughts. These maps showed the full prerequisite structure for a given concept in machine learning or probabilistic artificial intelligence.

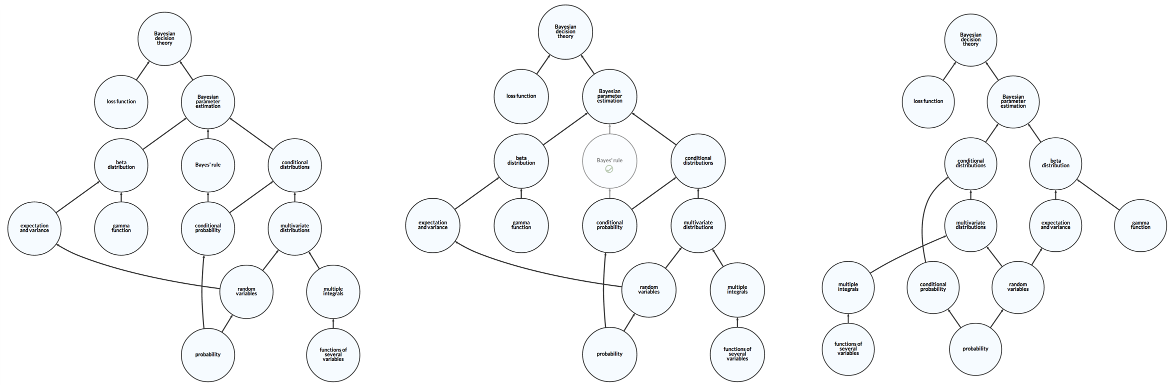


Figure 3. One of Metacademy's primary forms of interaction is allowing users to remove concepts they know: (left) the original graph, (middle) the dimmed node is marked as known, (right) the node is removed from the graph. In removing the node, the Metacademy system naively redetermines the location of the edges and nodes without taking into account their previous locations. In the text, we argue that this form of interaction nullifies the user's mental map, and as a result, we have not included this feature in our system.

The edge wisps from step 4 give a simple visible indication of the number of prerequisites and outlinks for a given concept and are used in the interactive components discussed below. Figure 4 shows a medium-sized Metacademy knowledge map of 39 nodes rendered directly from dagre (74 visible edges with 61 visible edge crossings) and the same knowledge map obtained using the above algorithm (38 visible edges with 0 visible edge crossings).

We employ the following interactive tools and techniques to help the user explore and understand the knowledge map generated using the visible-edge-reduction-algorithm:

- hovering over nodes highlights the inlinks/outlinks from that node as well as the source (target) of the inlink (outlink) and shows a set of clickable icons that the user can use to e.g. show the content associated with the node
- clicking on a node triggers a series of animations that transitions to the focused mode for that node (see below)
- clicking on an edge shows a pop-up justification for the edge
- a contractible side menu gives users the ability to search for a concept in the graph

The "focused mode" shows the user a selected concept as well its dependencies and outlinks. We use a series of animations to contract and expand the graph when a user enters the focused mode: the node is highlighted, then centered on the screen, then the dependencies and outlinks are translated so that they are adjacent to the focused node while the other nodes and edges fade out. This sequence is based on similar transitions in the Open Graph Drawing framework [10]. The animations and interactive tools were implemented using the d3 JavaScript library [8].

In the focused mode, a small information box appears at the bottom of the screen and indicates the number of hidden nodes and edges. The entire animation sequence takes slightly longer than one second. Clicking on an adjacent node within focused mode moves that node into focus, while clicking on the focused node runs the sequence of animations in reverse and returns the user to the full graph. Figure 5 shows a visualization of focused mode and the associated transition.

Knowledge Map Creation

In this subsection we describe our visual knowledge map creator. While this tool is independent of a particular platform, its current implementation is aligned for the Metacademy system. We designed this visual editor using the d3 JavaScript library [8] and only included features that benefitted the following operations: add/remove concepts, add/remove relationships between concepts (edges), add/remove associated content, and preview the knowledge map using the techniques discussed in the previous subsection.

Figure 6 shows the knowledge map creator. The main view consists of a large graph editing surface and a small toolbox. The user can perform the following editing operations:

- add node: shift + click on the editing surface
- remove node: click on a node and press delete or backspace
- move node: click + drag node
- change node title: shift + click on the node
- edit content associated with node: click on the small circle that appears when hovering the node
- expand/contract a node's dependencies (outlinks): click the plus/minus icon at the top (bottom) of the node
- add directed edge: shift + click on the source node and drag to the target node
- remove edge: click on a edge and press delete or backspace
- scale graph: scroll
- translate graph: click + drag on the editing surface

In addition, the toolbox shown in Figure 7 provides the user with the ability to optimize the graph location using dagre, clear all nodes and edges from the graph, preview the graph using the knowledge map exploration system described in this

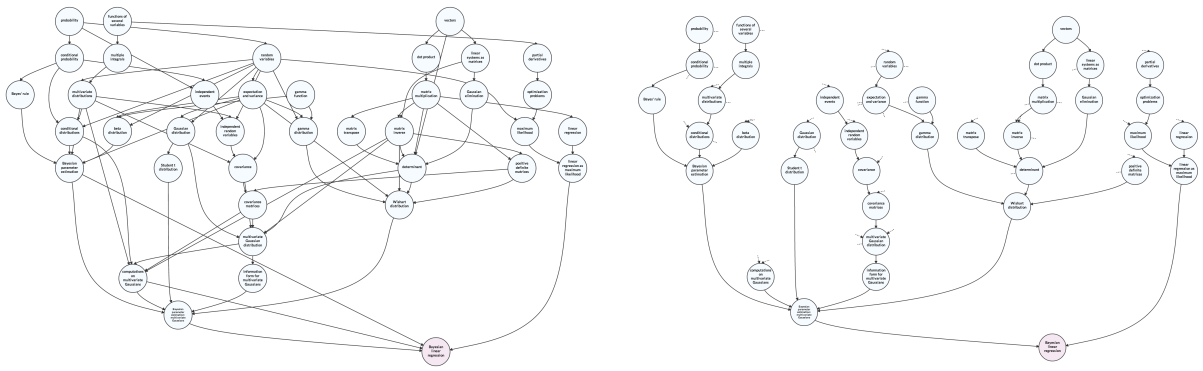
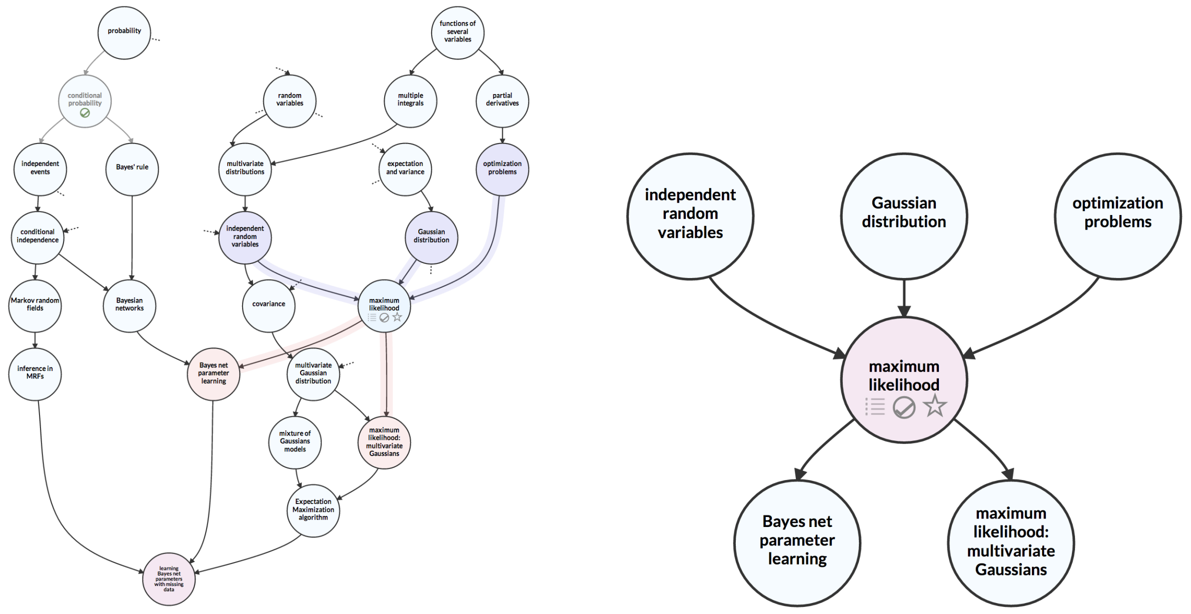


Figure 4. The left figure shows the knowledge map obtained directly using the dagre output; the right figure shows the knowledge map obtained using our visible-edge-reduction algorithm.



probability
functions of several variables
conditional probability
random variables
multiple integrals
partial derivatives
independent events
Bayes' rule
multivariate distributions
expectation and variance
optimization problems
conditional independence
independent random variables
Gaussian distribution
Markov random fields
Bayesian networks
covariance
maximum likelihood
inference in MRFs
Bayes net parameter learning
multivariate Gaussian distribution
mixture of Gaussians models
maximum likelihood: multivariate Gaussians
Expectation Maximization algorithm
learning Bayes net parameters with missing data
independent random variables
Gaussian distribution
optimization problems
maximum likelihood
Bayes net parameter learning
maximum likelihood: multivariate Gaussians

Figure 5. The left figure shows the highlighted edges and nodes when hovering over a specific node. The right figure shows the "focused mode" that occurs after clicking the node. Clicking the focused node in the focused mode returns to the full-graph display, while clicking the adjacent nodes places the focus on them.

paper, and download/upload a graph to/from a JSON[5] representation.

The search bar at the bottom of the toolbox gives users the ability to add preexisting concepts to graph. When a user adds a preexisting concept to the graph, the inlinks/outlinks of that node are contracted by default so that e.g. 20-50 prerequisite nodes are not immediately placed into the graph. When expanding/contracting nodes, a small information box at the bottom of the screen displays the number of hidden nodes and edges.

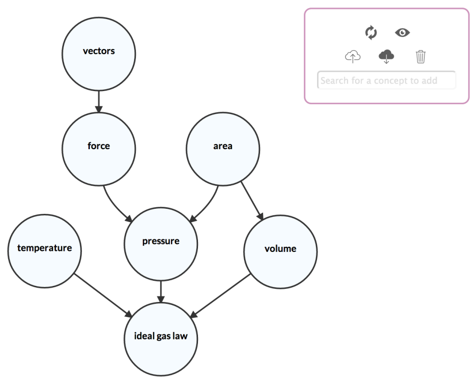


Figure 6. This figure displays the graph editor described in the text.

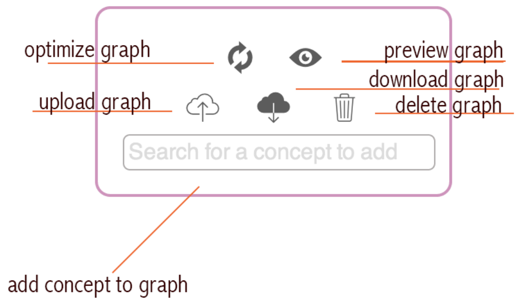
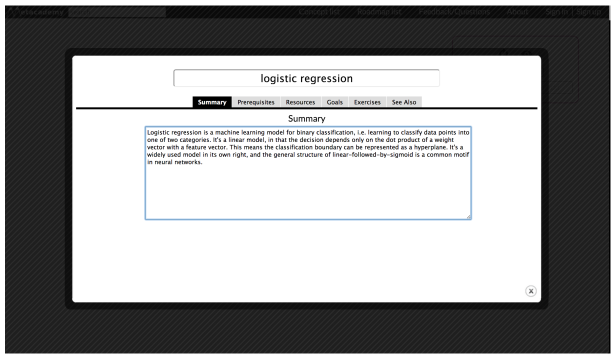


Figure 7. This figure shows the annotated toolbox for the graph editor.

When hovering over a node, a small circle appears on the outer radius of the node. If the user clicks the small circle then a content editor overlays the graphs with roughly a 0.8 alpha value, Figure 8. This content editor lets the user provide a summary for the concept, as well as provide links to learning resources and justifications for the prerequisites. Presently, this form is hard-coded html, but in the near future, we would like to create a simple interface for specifying the types data associated with the knowledge map. In this way, we hope to make this system usable outside of Metacademy.

DISCUSSION

The Metacademy system was designed so that individuals in variegated disciplines could both contribute content and explore preexisting content. Until now, this process involved



Logistic regression is a machine learning model for binary classification, i.e. learning to classify data points into one of two categories. It's a linear model, in that the decision depends only on the dot product of a weight vector with a feature vector. This means the classification boundary can be represented as a hyperplane. It's a widely used model in its own right, and the general structure of linear-followed-by-sigmoid is a common motif in neural networks.

Figure 8. This figure shows the content-editing form that overlays the knowledge map creator. This form allows users to associate content with the graph.

exploring large static graphs and editing text files. The contributions from this paper will help users interactively explore the knowledge maps and associated content. Furthermore, users can now contribute content via a visual knowledge map creator.

From watching users interact with our system during a research-poster session, we found that several users had had comments along the line of "visualizing the edges isn't necessary until you've chosen a reference node and you want to see it's relationships." While this certainly is not a formal usability study, we found it encouraging the users organically agreed with one of the driving forces for this work. We may pursue a formal usability study in the future.

This work and its inevitable progressions and extensions can be found at **`http://www.metacademy.org`**. The exact system described in this paper can be found at **`http://github.com/metacademy/metacademy|application/tree/graph|editor`**.

FUTURE WORK

The Khan Academy knowledge map inspired this paper, and in the coming weeks, we plan to make our system compatible with their data. As mentioned in the previous section, a formal user study would help validate or nullify our conjecture that our knowledge map exploration and creation tools help users efficiently explore and create educational material.

Beyond the realm of visualization, we would like to explore how users interact with these graphs in an online educational setting. Some interesting questions to answer include: what advantages and disadvantages does a knowledge map system have over traditional content presentations such as a textbook or wiki page, do users that create knowledge maps perform better in the class (not a concept map, as has been studied countless times in educational psychology), and to what extent are large knowledge maps useful, i.e. how big is too big?

ACKNOWLEDGEMENTS

Colorado Reed would like to thank Kristin Stephens for her fantastic feedback throughout this project and Dan Allan and Zach MacHardy for testing early versions of the knowledge map creator.

[5]**`http://www.json.org`**

REFERENCES

1. dagre javascript library. `https://github.com/cpettitt/dagre`. Accessed: 2013-12-01.
2. Duolingo. `http://www.duolingo.com`. Accessed: 2013-12-01.
3. Khan academy. `http://www.khanacademy.org`. Accessed: 2013-12-01.
4. Metacademy. `http://www.metacademy.org`. Accessed: 2013-12-01.
5. Palantir. `http://www.palantirtech.com`. Accessed: 2013-12-01.
6. Touchgraph. `http://www.touchgraph.com/navigator`. Accessed: 2013-12-01.
7. Batagelj, V., and Mrvar, A. *Pajekanalysis and visualization of large networks*. Springer, 2004.
8. Bostock, M., Ogievetsky, V., and Heer, J. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).
9. Cañas, A. J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gómez, G., Arroyo, M., and Carvajal, R. Cmaptools: A knowledge modeling and sharing environment. In *Concept maps: Theory, methodology, technology. Proceedings of the first international conference on concept mapping*, vol. 1 (2004), 125–133.
10. Chimani, M., and Gutwenger, C. The open graph drawing framework (ogdf). `http://www.ogdf.net/`.
11. Eisner, J., Kornbluh, M., Woodhull, G., Buse, R., Huang, S., Michael, C., and Shafer, G. Visual navigation through large directed graphs and hypergraphs. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'06), Poster/Demo Session* (Baltimore, Oct. 2006), 116–117.
12. Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K.-P. A technique for drawing directed graphs. *Software Engineering, IEEE Transactions on 19*, 3 (1993), 214–230.
13. Khan, S. *The one world schoolhouse: Education reimagined*. Hachette Digital, Inc., 2012.
14. Krzywinski, M., Birol, I., Jones, S. J., and Marra, M. A. Hive plotsrational approach to visualizing networks. *Briefings in Bioinformatics 13*, 5 (2012), 627–644.
15. Misue, K., Eades, P., Lai, W., and Sugiyama, K. Layout adjustment and the mental map. *Journal of visual languages and computing 6*, 2 (1995), 183–210.
16. Nielsen, J. Evaluating the thinking-aloud technique for use by computer scientists. In *Advances in human-computer interaction (vol. 3)*, Ablex Publishing Corp. (1993), 69–82.
17. Sugiyama, K., Tagawa, S., and Toda, M. Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics, IEEE Transactions on 11*, 2 (1981), 109–125.
18. Van Ham, F., and Perer, A. Search, show context, expand on demand: Supporting large graph exploration with degree-of-interest. *Visualization and Computer Graphics, IEEE Transactions on 15*, 6 (2009), 953–960.