Proceedings of the 2013 Uncertainty in Artificial Intelligence Application Workshops:

Part I: Big Data meet Complex Models

and

Part II: Spatial, Temporal, and Network Models

Workshops took place on July 15, 2013, in Bellevue, Washington, USA.

Edited by
Russell G. Almond, Florida State University
and
Ole J. Mengshoel, Carnegie Mellon University

CEUR Workshop Proceedings, Vol. 1024
http://ceur-ww.org/Vol-1024/

Preface

The *Uncertainty in Artificial Intelligence (UAI) Application Workshop* was conceived in 2002, and first held at UAI 2003 in Acapulco, Mexico; the first time a workshop was associated with the Conference. This year marks the Workshop's tenth anniversary, having been held in all years since but 2010. The goal of the workshops has been and will continue to be to look at the practical issues that arise in fielding applications based on the methods explored in the main conference.

Two half-day application Workshops where held on July 15, 2013 in Bellevue, Washington, USA, after the main UAI 2013 conference, along with two other workshops on specific topics. This volume collects the papers from both application workshops.

Application Workshop I: Big Data meet Complex Models

The theme of this workshop was large data sets containing many different kinds of data, and it especially emphasized the procedures that are used to combine data from various sources as part of the modeling process. There were 9 submissions. Each submission was reviewed by at least 1, and on the average 2.8 program committee members. The committee decided to accept 6 papers. Because one paper contained references to proprietary information, one set of authors published only the abstract in this volume.

Application Workshop II: Spatial, Temporal, and Network Models

The theme of this workshop was spatial, temporal, and network data. One may be interested in considering uncertainty in mobile data, generated by a GPS-enabled phone or a car. Another example is this: A scientist develops a probabilistic model in the form of a Bayesian network or a Markov random field. A computer scientist or computer engineer is then concerned about how to efficiently compile and execute the model in order to compute posterior distributions or estimate parameters on a multi-core CPU, a GPU, a Hadoop cluster, or a supercomputer. How well has this model worked, and what are current challenges and opportunities? There were 7 submissions. Each submission was reviewed by 3 program committee members. The committee decided to accept 5 papers, and all of these are being published in this volume.

Thanks to the program committees who put up with all of our nagging, and to all of the authors who came to us with a wide variety different applications, making the job of reading the papers much more interesting. Special thanks to John Mark Agosta of Toyota-ITC who as the UAI Workshop Chair (and past Applications Workshop Chair) helped us work out many details; and to Marina Meila of the University of Washington, who handled the local arrangements for us. Thanks also to EasyChair.org for helping with the submission and review process, to the volunteers who created the ceur-make facility for helping with the proceedings, and the people at CEUR for hosting our final papers.

Finally, thanks to the Association for Uncertainty in Artificial Intelligence (`http://auai.org/`) for hosting this workshop.

July 15, 2013
Bellevue, Washington, USA

Russell G. Almond and Ole J. Mengshoel

Program Committee

Part I: Big Data meet Complex Models

Russell Almond	Florida State University
Marek Druzdzel	University of Pittsburg
Julia Flores	Universidad de Castilla-La Mancha
Lionel Jouffe	Bayesia SAS
Kathryn Laskey	George Mason University
Suzanne Mahoney	Innovative Decisions
Thomas O'Neil	The American Board of Family Medicine
Linda van der Gaag	Utrecht University
Additional Reviewers	
Bermejo, Pablo	
Martínez, Ana María	

Part II: Models for Spatial, Temporal, and Network Data

Dennis Buede	Innovative Decisions
Asela Gunawardana	Microsoft
Jennifer Healey	Intel
Oscar Kipersztok	Boeing
Branislav Kveton	Technicolor
Helge Langseth	Norwegian University of Science and Technology
Ole Mengshoel	Carnegie Mellon University
Tomas Singliar	Boeing
Enrique Sucar	Instituto Nacional de Astrofisica Optica y Electronica, Mexico
Tom Walsh	Massachusetts Institute of Technology

Table of Contents

Debugging the Evidence Chain

Russell G. Almond* Florida State University
Yoon Jeon Kim Florida State University
Valerie J. Shute Florida State University
Matthew Ventura Florida State University

Abstract

In Education (as in many other fields) it is common to create complex systems to assess the state of latent properties of individuals — the knowledge, skills, and abilities of the students. Such systems usually consist of several processes including (1) a context determination process which identifies (or creates) *tasks*—contexts in which evidence can be gathered,—(2) an evidence capture process which records the work product produced by the student interacting with the task, (3) an evidence identification process which captures observable outcome variables believed to have evidentiary value, and (4) an evidence accumulation system which integrates evidence across multiple tasks (contexts), which often can be implemented using a Bayesian network. In such systems, flaws may be present in the conceptualization, identification of requirements or implementation of any one of the processes. In later stages of development, bugs are usually associated with a particular task. Tasks which have exceptionally high or unexpectedly low information associated with their observable variables may be problematic and merit further investigation. This paper identifies individuals with unexpectedly high or low scores and uses weight-of-evidence balance sheets to identify problematic tasks for follow-up. We illustrate these techniques with work on the game *Newton's Playground*: an educational game designed to assess a student's understanding of qualitative physics.

Key words: Bayesian Networks, Model Construction, Mutual Information, Weight of Information, Debugging

1 Introduction

The primary goal of educational assessment is to draw inferences about the unobservable pattern of student knowledge, skills and abilities from a pattern of observed behaviors in recognized contexts. The reasoning chain of an assessment system has several links: (1) It must recognize that the student has entered a context where evidence can be gathered (often, this is done by providing the student with a problem that provides the assessment context). We call such a context a *task*, as frequently it is the task of solving the problem which provides the required evidence. (2) The relevant parts of the student's performance on that task, the student's *work product*, must be captured. (3) The work product is then distilled into a series of *observable outcome* variables. (4) These observable outcome variables are used to update beliefs about the latent proficiency variables which are the targets of interest.

Bayesian networks are well suited for the fourth link in the evidentiary chain. Often the network can be designed to have a favorable topology, where observable variables from different contexts are conditionally independent given the latent proficiency variables. In such cases, the Bayesian network can be partitioned into a student proficiency model—containing only the latent proficiency variables—and a series of evidence models (one for each task)—capturing the relationships between the proficiency and evidence models for a particular task (Almond & Mislevy, 1999).

When the assessment system does not perform as expected, there is still a model with hundreds of variables that must be debugged. Furthermore, the problem may not lie just in the Bayesian network, the last link of the evidentiary chain, but anywhere along that chain. By using various information metrics, the prob-

*Paper submitted to Big Data Meets Complex Models, Application Workshop at Uncertainty in Artificial Intelligence Conference 2013, Seattle, WA.

lem can be traced to the parts of the evidentiary chain associated with a particular tasks. In particular, if the anomalous behavior can be associated with a particular individual attempting a particular task, this can focus troubleshooting effort to places where it is likely to provide the most value.

This paper explores the use of information metrics in troubleshooting the assessment system embedded in the game *Newton's Playground*(*NP*; Section 2). Section 3 describes a generic four process architecture for an assessment system. In *NP* tasks correspond to game levels; Section 4 describes some information metrics used to identify problematic game levels. Section 5 describes some of the problem identified so far, and our future development and model refinement plans.

2 Newton's Playground

Shute, Ventura, Bauer, and Zapata-Rivera (2009) explores the idea that if an assessment system can be embedded in an activity that students find pleasurable (e.g., a digital game), and that the activity requires them exercise a skill that educators care about (e.g., knowledge of Newton's laws of motion), then by observing performance in that activity, educators can make unobtrusive assessment of the students ability which can be used to guide future instruction. *Newtons Playground* (Shute & Ventura, 2013) is a two-dimensional physics game, inspired by the commercial game *Crayon Physics Deluxe*. It is also designed to be an assessment of three different aspects of proficiency: qualitative physics (Ploetzner & VanLehn, 1997), persistence, and creativity. This paper focuses on assessment of qualitative physics proficiency.

2.1 Gameplay

NP is divided into a series of levels, where each level consists of a qualitative physics problem to solve. In each game level, the player is presented with a drawing containing both fixed and movable objects. The goal of the level is to move the ball to a balloon (the target), by drawing additional objects on the screen. Most objects (both drawn and preexisting) are subject to the laws of gravity (with the exception of some fixed background objects) and Newton's laws of motion. (The open source Box 2D (Catto, 2011) physics engine provides the physics simulation.)

Figure 1 shows the initial configuration of a typical level called Spider's Web. Figure 2 shows one possible solution in which the player has used a springboard (attached to the ledge with two pins—small round circles) to provide energy to propel the ball up to the balloon. Deleting the weight will cause the ball to strike

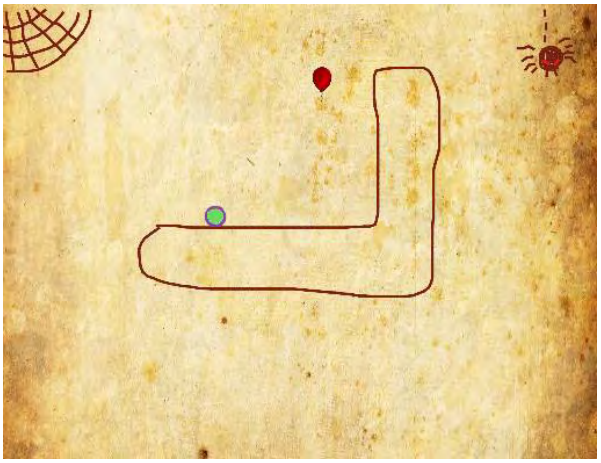


Figure 1: Starting Position for Spider Web Level

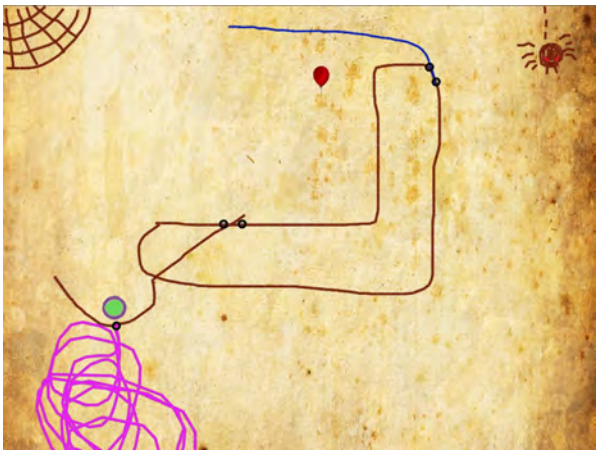


Figure 2: Spider Web Level with Springboard Solution.

ramp attached to the top of the wall which keeps the ball from flying over the target.

The focus of the current version has been on four *agents of motion* (simple machines): ramps, levers, springboards and pendulums. The game engine detects when one of those four agents was used as part of the solution. The game awards a trophy when the player solves a game level. Gold trophies are awarded if the solution is efficient (uses few drawn objects) and silver trophies are given as long as the goal is reached.

2.2 Proficiency and Evidence Models

The yellow nodes in Figure 3 show the student proficiency model for assessing a player's qualitative physics understanding. The highest level node, *Newton's Three Laws*, is the target of inference. It is divided into two components: one related to the application of those laws in linear motion, and one in angular

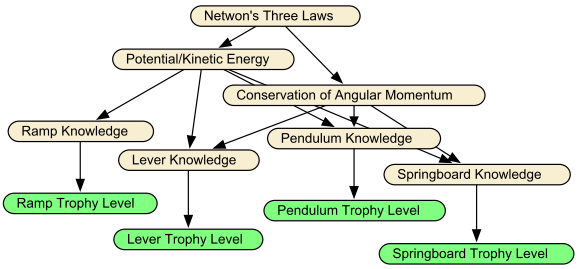


Figure 3: Physics Proficiency Model and Generic Observables

motion. The next layer has four nodes representing the four agents of motion. All of the nodes in the proficiency model had three levels: `High`, `Medium` and `Low`, and *expected a posterior* (EAP) scores could be calculated by assigning those levels a numerical value (3, 2, and 1, respectively) and taking the expectation.

The final layer of the model, shown in green represents the observable outcome variables from a generic level. These take on three possible values: `Gold`, `Silver` or `None`. The first two states are observed when the student solved using a particular agent. In that case, the observable for the correspond agent is set to the color of trophy received and the other observables are left unobserved. If the student attempts, but does not solve, the level the the observables corresponding to agents of motion the level designers thought would lead to solutions are set to `None`. The difficulty of a solution of each type, and the depth of physics understanding required, varies from level to level. So the green layer must be repeated for each game level. Version 1.0 (described in this paper) used 74 levels, so the complete Bayesian network had 303 nodes.

2.3 Field Test

In Fall 2012, a field trial was conducted using 169 8th grade students from a local middle school. The students were allowed to play the game for 4 45-minute class periods. The game engine kept complete logs of their game play. Students watched video demonstrations of how to create the four agents of motion in the game, and then were allowed to work through the game at their own pace. Game levels were grouped into playgrounds, with earlier playgrounds containing easier levels than the later playgrounds. Students were told that the player who got the most gold trophies would receive an extra reward.

One behavior which was often observed was the drawing of a large number of objects on the screen (often just under the ball to lift it higher), without a systematic plan for how to solve the level. Such "stacking" solutions had been observed in early playtests, and an object limit had been put in place to prevent it, but these "gaming" solutions were still observed during the field trial. Such solutions could lead to a silver trophy, but not to a gold trophy.

In addition to playing the game, a nine-item qualitative physics pretest and a matched nine-item posttest where given to the players. The pretest and posttest were not very stable measures of qualitative physics. On six different pendulum items (three from the pretest, three from the posttest) the students performed only slightly better than the guessing probabilities. The reliabilities (Cronbach's α Kolen & Brennan, 2004/1995) of the resulting six item tests were 0.5, and 0.4 for Forms A and B respectively.[1] This is a problem as physics understanding as shown on the posttest was the criterion measure, and these numbers form an effective upper bound on the correlation expected between the Bayesian network scores and the posttest.

We trained the Bayesian network using data from the field trial, and scored the field trial students. The correlation between the EAP scores from the highest level node and the physics pretest and posttest was around 0.1, which is not significantly different from zero at this sample size. Clearly there were problems in the assessment system that needed to be identified and addressed.

3 Four Process in the Evidence Chain

Because the correlation of the within game measure of Physics is so low, there must be a bug somewhere within the assessment system. A high level architecture of the assessment system will help define possible places. Figure 4, adapted from (Almond, Steinberg, & Mislevy, 2002), provides a generic architecture onto which assessment systems can be matched. It describes an assessment system that consists of four processes: *context determination*, *evidence capture*,[2] *evidence identification*, and *evidence accumulation*. In a general system, these can be human or machine processes, and several processes may be combined into a single piece of software, but all of the steps are present. Throughout, we will assume that the goal is to make inferences about the state of certain latent variables, which we will call the targets of inference.

[1] Half the students received Form A as a pretest and half as a posttest. This counterbalancing allowed the scores on the two forms to be equated.

[2] This process is called *presentation* in Almond et al. (2002). It is renamed here because it is the role of capturing the work product of the task is more important than the

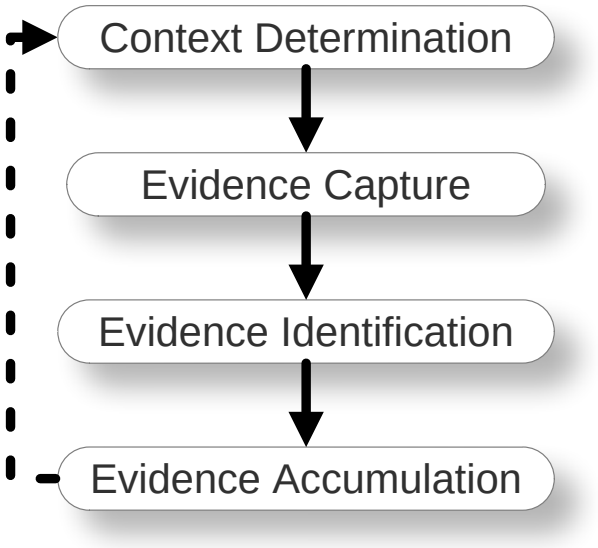


Figure 4: Four Process model of an Evidence Chain

Context determination is a process that identifies contexts in which evidence about the targets of interference can be gathered. In an educational assessment, these are often called *tasks*, as they represent problems a student must solve, or things a student must do. In a traditional assessment (like a college entrance exam), the test designers author tasks which are presented to the students forming the context for evaluating proficiency. When a student is engaged in free exploration with a simulator or game, the challenge in context determination is recognizing when current state of the simulator corresponds to a "task" that can be used to gather evidence (Mislevy, Behrens, DiCerbo, Frezzo, & West, 2012).

Other domains of application could use a mixture of engineered and natural contexts. For example, when trouble shooting a vehicle, the operators' reports of problems form natural contexts, while tests inside the garage are engineered contexts. Engineered contexts often provide stronger evidence than natural ones, because factors that might provide alternative explanations, and hence weaken the evidence for the targets of inference, can be controlled.

In *NP*, the contexts (tasks) are the game levels and they fall somewhere in between the natural and engineered range. Each of the game levels was designed by a member of the team, and each game level was designed to be solvable with a particular agent of motion (sometimes more than one). However, we had no control over which agent(s) the player would attempt to apply to the problem, and hence that part of the context was natural. Note that contexts are often described by variables (task model variables Almond, Kim, Velasquez, & Shute, 2012) that provide details about the context. In *NP*, the agents that the task designer thought provided reasonable solution paths (the `applicable agents`) and the task designer's estimate of difficulty were two such variables.

Evidence capture is a process that captures the raw data which will form the basis of the evidence. In educational assessment, we call that captured data the `work product` and note that this could come in a large variety of formats (e.g., video, audio, text, a log file of event traces). In *NP*, the evidence capture process was the game itself, and the work product consisted of a log file containing information about the player's interaction with the system (sufficient to replay the level), as well as additional information about the attempt (e.g., how long the player spent, how many objects were created and deleted, whether the player received a gold or silver trophy, etc.).

The *evidence identification* process takes the work product gathered by the evidence capture process and extracts certain key features: the *observable outcome variables*. One key difference of this process from the evidence accumulation process is that it always operates within a single context. The goal here is to reduce the complexity of the work product to a small, manageable number of variables. For example, a human rater (or natural language processing software) might rate an essay on several different traits. Those traits would be the observable outcome variables.

One design detail which is always tricky is figuring out how much processing of the work product to put into the evidence capture and how much is left for the evidence identification process. In *NP*, the evidence identification process was a collection of Perl scripts that extracted the observables from the log files. In some cases, it proved more convenient to implement the evidence identification rules in the game engine. In particular, it was important to identify if an object drawn by the player was a ramp, lever, pendulum or springboard. That was easier to do inside the game (i.e., evidence capture process) where the physics engine could be queried about the interactions of the objects. In other cases, it proved more convenient to filter the observables in the evidence accumulation process. For example, we did not want to penalize the player for failing to solve a level with a particular agent if the level was not designed to be solved with that agent. In this case, it turned out to be simpler to implement this on the Bayes net side (i.e., the evidence accumulation process), and the observable node corresponding to an agent would not be instantiated to `None` if the agent was not applicable for that level.

role of presenting the task.

The *Evidence accumulation* process is responsible for combining evidence about the targets of inference across multiple contexts. In *NP*, the evidence accumulation process consisted of a collection of Bayesian networks: a student proficiency model for each student, and a collection of evidence models for each game level. When it received a vector of observables for a particular student on a particular game level, it drew the appropriate evidence model from the library and attached it to that student's proficiency model. It then instantiated nodes in the evidence model corresponding to the observable values, and propagated the evidence into the proficiency model. The evidence model was then detached from the proficiency model which remained as a record of student proficiency. It could be queried at any time to provide a score for a student (Almond, Shute, Underwood, & Zapata-Rivera, 2009).

The dashed line in Figure 4 from the evidence accumulation process to the context determination process[3] is to indicate that in some situations the context determination might query the current beliefs about the targets of inference before selecting the next task (context). This produces a system that is adaptive (Shute, Hansen, & Almond, 2008). In *NP*, the player was free to choose the order for attempting the levels, hence this link was not used.

The four processes can be put together into a system that provides real-time inference or as a series of isolated steps. In version 1.0 of *NP*, only the evidence capture system (the game itself) was presented to the players in real-time. As the design of the other parts of the system was still undergoing refinement, it was simpler to implement them as separate post-processing steps. In a future version, these process will be integrated with the game so that players can get scores from the Bayes net as they are playing.

Developing each process requires three activities: *conceptualization*—identifying the key variables and work products and their relationships,—*requirement specifications*—writing down the rules by which values of the variables are determined,—and *implementation*—realizing those rules in code. A bug that causes the system to behave poorly can be related to a flaw in any one of those three activities, and can affect one or more of the four processes.

By the time the system was field tested, obvious bugs had been found and fixed. The remaining bugs only occur in particular particular game levels, and particular patterns of interaction with those levels. Once the levels in which bugs manifest and the patterns of usage which cause the bugs to manifest are identified, the problems can be addressed. This may entail adjust parameters for the Bayesian network fragment associated with that network, changing the level, replacing the level or making changes to the game engine, evidence identification scripts, or instructions to players.

[3]Almond et al. (2002) called this the activity selection process, to emphasize its adaptive nature.

4 Information Metrics as Debugging Tools

It is always the case that students interacting with an assessment system do so in ways that were unanticipated by the assessment designers. Information metrics provide a mechanism for flagging levels which behave in unexpected ways. In particular, we expect that a properly working game level will provide high information for the applicable agents (the ones that the designers targeted) and low information for the inapplicable agents. Extremely high information could also be an indication of overfitting the model to data.

Section 4.1 looks at the parameters of the conditional probability table as information metrics. Section 4.2 looks at the mutual information between the observable variable and its immediate parent in the model. Section 4.3 looks at tracing the score of specific individuals as they work through the game to identify problematic player/level combinations.

4.1 Parameters of the Conditional Probability Tables

Following Almond et al. (2001) and Almond (2010), we used models based on item response theory (IRT) to determine the values of the conditional probability tables. For each table, the effective ability parameter, $\tilde{\theta}$, is determined by the value of the parent variable (the values were selected based on equally spaced quantiles of a normal distribution: -0.97 for `Low`, 0 for `Medium`, and 0.97 for `High`). The model is based on estimates for two probabilities, the probability of receiving any trophy at all (using a specified agent), and the probability of receiving a gold trophy given that a trophy was received. These are expressed as logistic regressions on the effective theta value:

$$\begin{aligned}\Pr(\text{Any Trophy}|\text{Agent Ability}) \\ = \text{logit}^{-1}\, 1.7a_S(\tilde{\theta} - b_S), \quad (1)\\ \Pr(\text{Gold Trophy}|\text{Any Trophy}, \text{Agent Ability}) \\ = \text{logit}^{-1}\, 1.7a_G(\tilde{\theta} - b_G); \quad (2)\end{aligned}$$

where the 1.7 is a constant to match the logistic function to the normal probability curve. The two equations are combined to form the complete conditional probabilities using the generalized partial credit model (Muraki, 1992).

The silver and gold *discrimination* parameters, a_S and a_G, represent the slope of the IRT curve when $\theta = b$. They are measures of the strength of the association between the observable and the proficiency variable it measures. In high-stakes examinations, discriminations of around 1 are considered typical, and discriminations of less than 0.5 are considered low. We expect lower discriminations in game-based assessments as there may be other reasons (e.g., lack of persistence) that a player would fail to solve a game level. Still, when a game level is designed to target a player's understanding of a particular agent, very low discrimination is a sign that it is not working. High discriminations (above 2.0) are often a sign of difficulty in parameter estimation.

The silver and gold *difficulty* parameters, b_S and b_G, represent the ability level required to have a 50% chance of success. They have the opposite sign of a typical intercept parameter, and they should fall on a unit normal scale: tasks with difficulties below -3 should be solved by nearly all participants and those with difficulties above 3 should be solved by almost no participants.

The complete model had four parameters, two difficulty and two discrimination parameters, for each level/agent combination. One member of our level design team provided initial values for those parameters based on the design goals, applicable agents, and early pilot testing.

Correlations between the posttest scores and the Bayes net scores using the expert parameters were low, so we developed a method for estimating the parameters from the field trial data. First, the pretest and posttest were combined (as they were so short) and then separated into subscales based on agent of motion. As the scores were short, the augmented scoring procedure of Wainer et al. (2001) was used to shrink the estimates towards the average ability. Each subscale was split into `High`, `Medium` and `Low` categories with equal numbers of students in each. This provides a proxy for the unobservable agent abilities for each student.

We used the agent ability proxies and the observed trophies to calculate a table of trophies by ability for each level. The tables were rather sparse as many students did not attempt many levels, and and typically used only one agent for each level attempted. To overcome this sparseness, the conditional probability tables generated using the expert parameters were added to the observed data, and then a set of parameter (a_S, a_G, b_S, b_G) were found that maximized the likelihood of generated the combined prior + observed table using a gradient decent algorithm.

Looking for extremely high discrimination values immediately flagged some problems with this procedure. In particular, cases where only one of two students attempted a level with a particular agent, but were successful, could result in an extremely high discrimination. Increasing the weight placed on the prior when calculating the prior+observation table reduced the occurrences of this problem.

There were still some level/agent combinations with extremely high discrimination, but we noticed that they had extremely high difficulties as well. Looking at the conditional probability tables generated by these parameter values we noticed that they were nearly flat (in other words, the three points on the logistic curve corresponding to the possible parent levels were in one of the tails of the logistic distribution). Because the conditional probability table was flat, the high discrimination does not correspond to high information, so is not likely to overweight evidence from that game level. Consequently, flagging just high discrimination produced too many false positives, and additional screening was needed.

4.2 Mutual Information

The *mutual information* of two variables X and Y is defined as:

$$MI(X,Y) = \sum_{x,y} \Pr(x,y) \log \frac{\Pr(x,y)}{\Pr(x)\Pr(y)} \ . \quad (3)$$

Calculating the mutual information for all of the level/agent combinations yielded a maximum mutual information of 0.09, with most mutual information values below 0.01. Figure 5 shows the mutual information for both applicable agent/level combinations and inapplicable ones.

Table 1 shows the conditional probability table parameters and mutual information for a few selected levels, looking at just the *Lever Trophy* observables. The particular levels were flagged because they had either high discrimination, high (in absolute value) difficulty or high mutual information. The game level "Stairs" is an example of a problem: it has an extremely high discrimination for silver trophies and an extremely high difficulty as well. Furthermore, the mutual information is toward the high end of the range. The level "Swamp People" is also a problem, it has a high gold discrimination as well as a high mutual information. Furthermore, lever was not thought to be a common way of solving the problem by the game designers.

It is important to use the mutual information as a screening criteria to eliminate false positives. The game level "Smiley" is an example of a false positive. Although the silver discrimination and difficulty are high, the mutual information is below 0.001, so

Table 1: Parameters and mutual information for selected lever observables.

	applicable	a_S	b_S	a_G	b_G	MI
Diving Board World	TRUE	0.897	5.036	0.024	1.974	0.000
Smiley	TRUE	3.368	7.255	0.002	1.479	0.000
St. Augustine	TRUE	0.897	5.036	0.024	1.974	0.000
Stairs	TRUE	11.084	10.756	0.000	0.774	0.064
Swamp People	FALSE	0.116	4.782	2.431	3.689	0.033
Ballistic Pendulum	FALSE	0.897	5.036	0.024	1.974	0.000

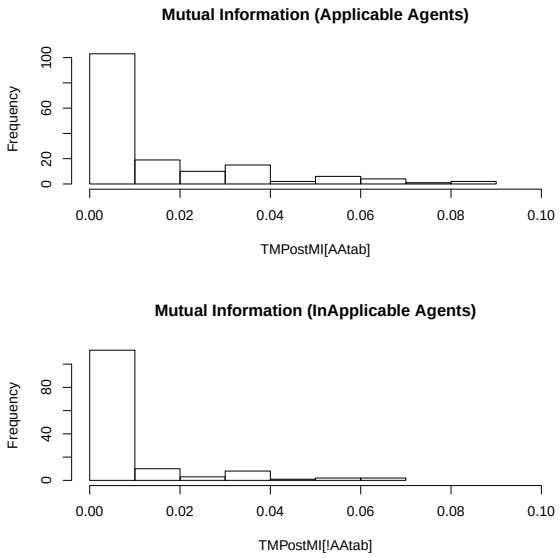


Figure 5: Histograms of Mutual Agent Distributions

the extreme parameter values are likely not causing a problem. This level could be a problem for a different reason: lever was judged to be an applicable solution agent, but the mutual information is low. The unexpectedly week evidentiary value of this level should be investigated.

4.3 Evidence Balance Sheets

The *weight of evidence*(Good, 1985) a piece of evidence E provides for a hypothesis H versus its negation $\overline{H}$ is:

$$W(H{:}E) = \log \frac{\Pr(E|H)}{\Pr(E|\overline{H})} = \log \frac{\Pr(H|E)}{\Pr(\overline{H}|E)} - \log \frac{\Pr(H)}{\Pr(\overline{H})} \ . \tag{4}$$

If the evidence arrives in multiple pieces, E_1 and E_2 (e.g., the evidence from each game level), the *conditional weight of evidence*:

$$W(H{:}E_2|E_1) = \log \frac{\Pr(E_2|H, E_1)}{\Pr(E_2|\overline{H}, E_1)} \ . \tag{5}$$

These sum in much the way that one would expect:

$$W(H{:}E_1, E_2) = W(H{:}E_1) + W(H{:}E_2|E_1) \ . \tag{6}$$

Madigan, Mosurski, and Almond (1997) suggest a *weight of evidence balance sheet*: simple graphical display for the conditional weights of evidence. Figure 7 shows an example. The leftmost column gives the game levels in the order that they were scored, as well as the agent and trophy that was received. The central column gives the conditional probability for the target node, *Newton's Three Laws* at various points in the scoring sequence. The third column gives the weight of evidence the most recent level provides for the hypothesis that the target node is at least at the level of `Medium`.

Constructing a balance sheet requires selecting a particular student. Interesting students can be identified by looking for outliers in the regression of the posttest (or pretest) scores on the Bayesian network EAP scores (Figure 6). Certain students were identified in this plot. Student S259 got no pretest items right (although that student got about 4 posttest items right, which was a good score), and had an EAP score of 2.3 (which is in the medium category for physics understanding).

Figure 7 shows the pattern of scores for this student. Early levels tend to have higher weights of evidence than later levels. Note that somewhere towards the middle of the sequence there are two huge spike in the weight of evidence. These correspond to the levels "jar of coins" and "Jurassic park"; both had weights of evidence of over 75. Table 2 presents the same information in a tabular fashion. Here the information is screened so that only levels with high weights of evidence are shown.

To systematically investigate what causes the spikes in the weight of evidence, we reviewed replay files of the identified students. For example, S259 mostly used solutions that "game the system"(e.g., crashing the system by drawing random large objects) and rarely tried to use applicable agents. Thus when he somehow managed to use an applicable agent and earned a trophy, the weight of evidence jumped.

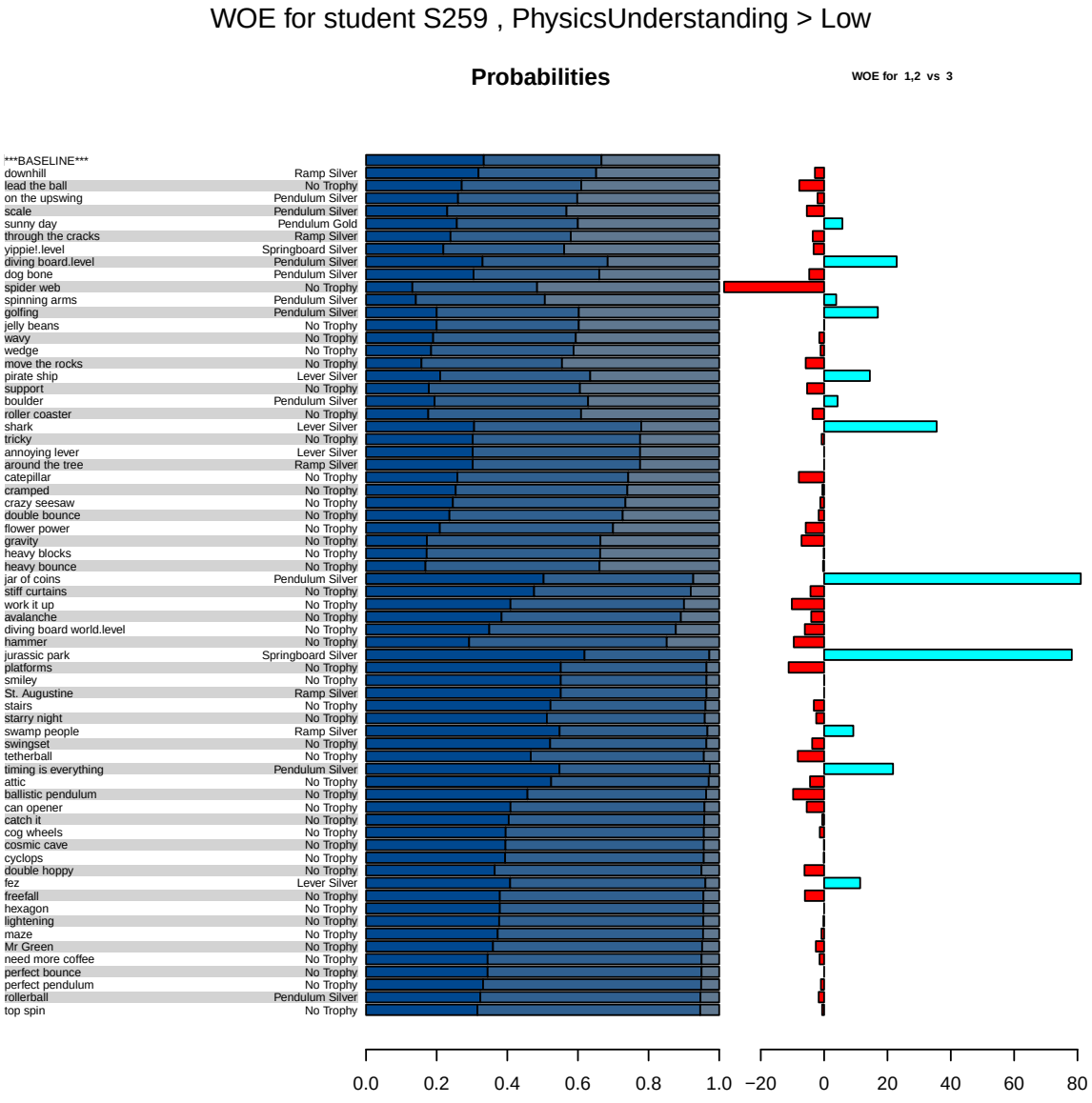


Figure 7: Weight of Evidence Balance Sheet for Student S259

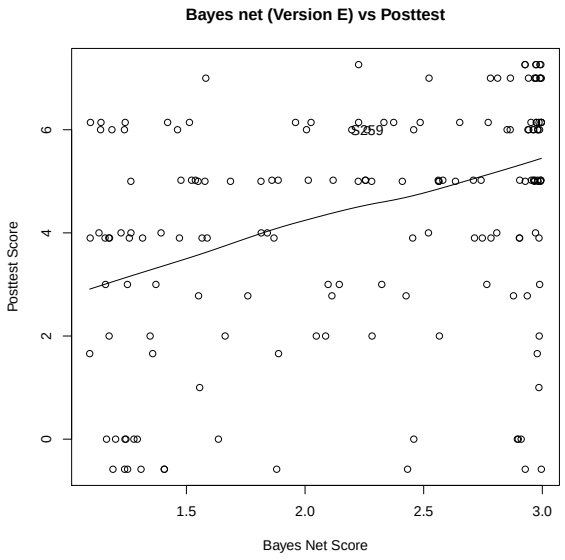


Figure 6: Scatterplot of Postttest versus Bayes net scores.

Table 2: Levels with high weights of evidence for Student S259

Level	WOE
lead the ball	-7.84
diving board	22.92
spider web	-32.59
golfing	16.97
pirate ship	14.45
shark	35.54
caterpillar	-7.99
jar of coins	80.04
work it up	-10.2
hammer	-9.58
Jurassic park	78.2
platforms	-11.21
swamp people	9.22
tether ball	-8.32
timing is everything	21.77
ballistic pendulum	-9.8
fez	11.38

For the case of "jar of coins", it is one of the levels that already has an applicable agent built in the level as an incomplete form (i.e., pendulum for this level), and all the player needs to do is to make the built-in agent work by completing it (e.g., add more mass to the pendulum bob). The review of his replay files revealed that he exploited the system again for jar of coin, but the system recognized his solution as an applicable due to the built-in agent. This finding should lead to one or more follow-up actions: (a) decrease discrimination for pendulum in the CPT of jar of coins, (b) revise the level to make it harder to "game" the system, and/or (c) replace the level with one that forces the player to directly draw the agent. We chose the third option for the next version of *Newton's Playground*.

5 Lessons Learned and Future Work

The work on constructing the assessment system for Newton's Playground is ongoing. Using these information metrics helped us identify problems in both the code and level design. For example, one case of unexpectedly low discrimination led to the discovery of a bug in the code that built the observed tables from the data (the labels of the `High` and `Low` categories were swapped and the observation table was built upside down). Unexpected high and low information also forced the designers to take a closer look at which agents students were actually using to solve the problems leading to a revision in the agent tables. Finally, viewing replays led us to identify places where the agent identification system misidentified the agent used to solve the problem. This led to improved values for the observable outcomes.

Correcting these problems lead to a definite improvement in the correlation between the Bayes net score and the pretest and posttest. With the revised networks and evidence identification code, the correlation with the pretest is 0.40 and with the posttest is 0.36, a definite improvement (and close to the limit of the accuracy available given the lack of reliability of the pretest and posttest).

We have also identified some conceptual errors that we are still working to address. In particular, a large number of the students (e.g., S259) engaged in off-track "gaming" behaviors, often earning silver trophies in the process. It is clear that the Bayesian network is lacking nodes related to that kind of behavior. Also, we need a better system for detecting that kind of behavior. These are being implemented in Version 2.0 of *Newton's Playground.*

References

Almond, R. G. (2010). 'I can name that Bayesian network in two matrixes'. *International Journal of Approximate Reasoning*, *51*, 167–178. Retrieved from `http://dx.doi.org/10.1016/j.ijar.2009.04.005`

Almond, R. G., DiBello, L., Jenkins, F., Mislevy, R. J., Senturk, D., Steinberg, L. S., et al. (2001). Models for conditional probability tables in educational assessment. In T. Jaakkola & T. Richardson (Eds.), *Artificial intelligence and statistics*

2001 (p. 137-143). Morgan Kaufmann.

Almond, R. G., Kim, Y. J., Velasquez, G., & Shute, V. J. (2012, July). *How task features impact evidence from assessments embedded in simulations and games.* Lincoln, NE. (Paper presented at the International Meeting of the Psychometric Society (IMPS))

Almond, R. G., & Mislevy, R. J. (1999). Graphical models and computerized adaptive testing. *Applied Psychological Measurement*, *23*, 223-238.

Almond, R. G., Shute, V. J., Underwood, J. S., & Zapata-Rivera, J.-D. (2009). Bayesian networks: A teacher's view. *International Journal of Approximate Reasoning*, *50*, 450–460.

Almond, R. G., Steinberg, L. S., & Mislevy, R. J. (2002). Enhancing the design and delivery of assessment systems: A four-process architecture. *Journal of Technology , Learning, and Assessment*, *1*, (online). Retrieved from `http://www.jtla.org/`

Catto, E. (2011). Box2D v2.2.0 user manual [Computer software manual]. Retrieved from `http://box2d.org/` (Downloaded July 25, 2012 from)

Good, I. J. (1985). Weight of evidence: A brief survey. In J. Bernardo, M. DeGroot, D. Lindley, & A. Smith (Eds.), *Bayesian statistics 2* (p. 249-269). North Holland.

Kolen, M. J., & Brennan, R. L. (2004/1995). *Test equating, scaling, and linking: Methods and practices* (2nd ed.). Springer-Verlag.

Madigan, D., Mosurski, K., & Almond, R. G. (1997). Graphical explanation in belief networks. *Journal of Computational Graphics and Statistics*, *6*(2), 160-181. Retrieved from `http://www.amstat.org/publications/jcgs/index.cfm?fuseaction=madiganjun`

Mislevy, R. J., Behrens, J. T., DiCerbo, K. E., Frezzo, D. C., & West, P. (2012). Three things game designers need to know about assessment. In D. Ifenthaler, D. Eseryel, & X. Ge (Eds.), *Assessment in game-based learning: Foundations, innovations, and perspectives* (pp. 59–81). Springer.

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). On the structure of educational assessment (with discussion). *Measurement: Interdisciplinary Research and Perspective*, *1*(1), 3-62.

Muraki, E. (1992). A generalized partial credit model: Application of an em algorithm. *Applied Psychological Measurement*, *16*, 159–176.

Ploetzner, R., & VanLehn, K. (1997). The acquisition of informal physics knowledge during formal physics training. *Cognition and Instruction*, *15*(2), 169–205.

Shute, V. J., Hansen, E. G., & Almond, R. G. (2008). You can't fatten a hog by weighing it - or can you? Evaluating an assessment for learning system called ACED. *International Journal of Artificial Intelligence in Education*, *18*(4), 289–316. Retrieved from `http://www.ijaied.org/iaied/ijaied/abstract/Vol_18/Shute08.html`

Shute, V. J., & Ventura, M. (2013). *Stealth assessment in digital games.* MIT series.

Shute, V. J., Ventura, M., Bauer, M. I., & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow. In U. Ritterfeld, M. J. Cody, & P. Vorderer (Eds.), *Serious games: Mechanisms and effects* (pp. 295–321). Routledge, Taylor and Francis.

Wainer, H., Veva, J. L., Camacho, F., Reeve III, B. B., Rosa, K., Nelson, L., et al. (2001). Augmented scores — "borrowing strength" to compute scores based on a small number of items. In D. Thissen & H. Wainer (Eds.), *Test scoring* (pp. 343–388). Lawrence Erlbaum Associates.

Acknowledgments

Many aspects of the *Newton's Playground* examples are based on work of the *Newton's Playground* team, Val Shute, P.I. In addition to the authors, the team includes Matthew Small, Don Franceschetti, Lubin Wang, and Weinan Zhao. Pete Stafford assisted with the data analysis. Work on *Newton's Playground* and this paper was supported by the Bill & Melinda Gates Foundation U.S. Programs Grant Number #0PP1035331, *Games as Learning/Assessment: Stealth Assessment.* Any opinions expressed are solely those of the authors.

Bayesian Supervised Dictionary learning

B. Babagholami-Mohamadabadi
CE Dept.
Sharif University
Tehran, Iran

A. Jourabloo
CE Dept.
Sharif University
Tehran, Iran

M. Zolfaghari
CE Dept.
Sharif University
Tehran, Iran

M.T. Manzuri-Shalmani
CE Dept.
Sharif University
Tehran, Iran

Abstract

This paper proposes a novel Bayesian method for the dictionary learning (DL) based classification using Beta-Bernoulli process. We utilize this non-parametric Bayesian technique to learn jointly the sparse codes, the dictionary, and the classifier together. Existing DL based classification approaches only offer point estimation of the dictionary, the sparse codes, and the classifier and can therefore be unreliable when the number of training examples is small. This paper presents a Bayesian framework for DL based classification that estimates a posterior distribution for the sparse codes, the dictionary, and the classifier from labeled training data. We also develop a Variational Bayes (VB) algorithm to compute the posterior distribution of the parameters which allows the proposed model to be applicable to large scale datasets. Experiments in classification demonstrate that the proposed framework achieves higher classification accuracy than state-of-the-art DL based classification algorithms.

1 Introduction

Sparse signal representation (Wright et al., 2010), has recently gained much interest in computer vision and pattern recognition. Sparse codes can efficiently represent signals using linear combination of basis elements which are called atoms. A collection of atoms is referred to as a dictionary. In sparse representation framework, dictionaries are usually learned from data rather than specified apriori (i.e wavelet).
It has been demonstrated that using learned dictionaries from data usually leads to more accurate representation and hence can improve performance of signal reconstruction and classification tasks (Wright et al., 2010). Several algorithms have been proposed for the task of dictionary learning (DL), among which the K-SVD algorithm (Aharon et al., 2006), and the Method of Optimal Directions (MOD) (Engan et al., 1999), are the most well-known algorithms. The goal of these methods is to find the dictionary $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K]$, and the matrix of the sparse codes $A = [a_1, a_2, ..., a_N]$, which minimize the following objective function

$$[\hat{A}, \hat{D}] = argmin_{A,D} \|X - DA\|_F^2, \quad s.t. \quad \|x_i\|_0 \leq T \ \ \forall i, \tag{1}$$

where $X = [x_1, x_2, ..., x_N]$ is the matrix of N input signals, K is the number of the dictionary atoms, $\|.\|_F$ denotes the Frobenius norm, and $\|x\|_0$ denotes the l_0 norm which counts the number of non-zero elements in the vector x.

2 Related Work

Classical DL methods try to find a dictionary, such that the reconstructed signals are fairly close to the original signals, therefore they do not work well for classification tasks. To overcome this problem, several methods have been proposed to learn a dictionary based on the label information of the input signals.
Wright (Wright et al., 2009), used training data as the atoms of the dictionary for Face Recognition (FR) tasks. This method determines the class of each query face image by evaluating which class leads to the minimal reconstruction error. Although the result of this method on face databases are promising, it is not appropriate for noisy training data. Being unable to utilize the discriminative information of the training data is another weakness of this method.
Yang (Yang et al., 2010), learned a dictionary for each class and obtained better FR results than Wright method. Yang (Yang et al., 2011), utilized Fisher Discriminant Analysis (FDA) to learn a sub-dictionary for each class in order to make the sparse coefficients more discriminative. Ramirez (Ramirez et al., 2010), added a structured incoherence penalty term to the

objective function of the class specific sub-dictionary learning problem to make the sub-dictionaries incoherent. Mairal (Mairal et al., 2009), introduced a supervised DL method by embedding a logistic loss function to learn a single dictionary and a classifier simultaneously. Given a limited number of labeled examples, most DL based classification methods suffer from the following problem: since these algorithms only provide point estimation of the dictionary, the sparse codes, and the classifier which could be sensitive to the choice of training examples, they tend to be unreliable when the number of training examples is small. In order to address the above problem, this paper presents a Bayesian framework for supervised dictionary learning, termed **Bayesian Supervised Dictionary Learning**, that targets tasks where the number of training examples is limited. Using the full Bayesian treatment, the proposed framework for dictionary learning is better suited to dealing with a small number of training examples than the non-Bayesian approach.

Dictionary learning based on the Bayesian non-parametric models was originally proposed by Zhou (Zhou et al., 2009), in which a prior distribution is put on the sparse codes (each sparse code is modeled as an element-wise multiplication of a binary vector and a weight vector) which satisfies the sparsity constraint. Although the results of this method can compete with the state of the art results in denoising, inpainting, and compressed sensing applications, it does not work well for classification tasks due to its incapability of utilizing the class information of the training data.

To address the above problem, we extend the Bayesian non-parametric models for classification tasks by learning the dictionary, the sparse codes, and the classifier simultaneously. The contributions of this paper are summarized as follows:

- The noise variance of the sparse codes (the sparsity level of the sparse codes) and the dictionary is learn based on the Beta-Bernoulli process (Paisley et al., 2009) which allows us to learn the number of the dictionary atoms as well as the dictionary elements.
- A logistic regression classifier (multinomial logistic regression (Bohning, 1992), classifier for multiclass classification) is incorporated into the probabilistic dictionary learning model and is learned jointly with the dictionary and the sparse codes which improves the discriminative power of the model.
- The posterior distributions of the dictionary, the sparse codes, and the classifier is efficiently computed via the VB algorithm which allows the proposed model to be applicable to large-scale datasets.
- The Bayesian prediction rule is used to classify a test instance and therefore the proposed model is less prone to overfitting, specially when the size of the training data is small. Precisely speaking, test instances are classified by weighted average of the parameters (the dictionary, the sparse codes of the test instances , and the classifier), weighted by the posterior probability of each parameter value given the training data.
- Using the Beta-Bernoulli process model, many components of the learned sparse codes are exactly zero, which is different from the widely used Laplace prior, in which many coefficients of the sparse codes are small but not exactly zero.

The remainder of this paper is organized as follows: Section 3 briefly reviews the Beta-Bernoulli process. The proposed method is introduced in Section 4. Experimental results are presented in Section 5. We conclude and discuss future work in Section 6.

3 Beta-Bernoulli Process

The Beta process $B \sim BP(c, B_0)$ is an example of a Lévy process which was originally proposed by Hjort for survival analysis (Hjort, 1990), and can be defined as a distribution on positive random measures over a measurable space $(\Omega, \mathcal{F})$.

B_0 is the base measure defined over Ω and $c(\omega)$ is a positive function over Ω which is assumed constant for simplicity. The Lévy measure of $B \sim BP(c, B_0)$ is defined as

$$\nu(d\pi, d\omega) = c\pi^{-1}(1-\pi)^{c-1} d\pi B_0(d\omega). \qquad (2)$$

In order to draw samples from $B \sim BP(c, B_0)$, Kingman (Kingman, 1993), proposed a procedure based on the Poisson process which goes as follows.

First, a non-homogeneous Poisson process is defined on $\Omega \times \mathcal{R}^+$ with intensity function ν. Then, $Poisson(\lambda)$ number of points $(\pi_k, \omega_k) \in [0,1] \times \Omega$ are drawn from the Poisson process ($\lambda = \int_{[0,1]} \int_{\Omega} \nu(d\omega, d\pi) = \infty$). Finally, a draw from $B \sim BP(c, B_0)$ is constructed as

$$B_\omega = \sum_{k=1}^{\infty} \pi_k \delta_{\omega_k}, \qquad (3)$$

where δ_{ω_k} is a unit point measure at ω_k (δ_{ω_k} equals one if $\omega = \omega_k$ and is zero otherwise) . It can be seen from equation 3, that B_ω is a discrete measure (with probability one), for which $B_\omega(A) = \sum_{k:\omega_k \in A} \pi_k$, for any set $A \subset \mathcal{F}$.

If we define $Z = Be(B)$ as a Bernoulli process with B_ω defined as 3, a sample from this process can be drawn as

$$Z = \sum_{k=1}^{\infty} z_k \delta_{\omega_k}, \tag{4}$$

where z_k is generated by

$$z_k \sim Bernoulli(\pi_k). \tag{5}$$

If we draw N samples $(Z_1, ..., Z_N)$ from the Bernoulli process $Be(B)$ and arrange them in matrix form, $\mathbf{Z} = [Z_1, ..., Z_N]$, then the combination of the Beta process and the Bernoulli process can be considered as a prior over infinite binary matrices, with each column Z_i in the matrix $\mathbf{Z}$ corresponding to a location, δ_{ω_i}. By marginalizing out the measure B, the joint probability distribution $P(Z_1, ..., Z_N)$ corresponds to the Indian Buffet Process (Thibaux et al., 2007).

4 Proposed Method

All previous classification based sparse coding and dictionary learning methods have three shortcomings. First, the noise variance or the sparsity level of the sparse codes must be specified apriori in order to define the stopping criteria for estimating the sparse codes. Second, the number of the dictionary atoms must be set in advance (or determined via the cross-validation technique). Third, a point estimate of the dictionary and the sparse codes are used to predict the class label of the test data points which can result in overfitting. To circumvent these shortcomings, we propose a Bayesian probabilistic model in terms of the Beta-Bernoulli process which can infer both the dictionary size and the noise variance of the sparse codes from data. Furthermore, our approach integrates a logistic regression classifier (multinomial logistic regression classifier for multiclass classification) into the proposed probabilistic model to learn the dictionary and the classifier simultaneously while most of the algorithms learn the dictionary and the classifier separately.

4.1 Problem Formulation

Consider we are given a training set of N labeled signals $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in R^{M \times N}$, each of them may belong to any of the c different classes. We first consider the case of $c = 2$ classes and later discuss the multiclass extension. Each signal is associated with a label $(y_i \in \{-1, 1\}, i = 1, ..., N)$. We model each signal $\boldsymbol{x}_i$, as a sparse combination of atoms of a dictionary $D \in R^{M \times K}$, with an additive noise ϵ_i. The Matrix form of the model can be formulated as

$$X = DA + E, \tag{6}$$

where $X_{M \times N}$ is the set of the input signals, $A_{K \times N}$ is the set of the K dimensional sparse codes, and $E \sim \mathcal{N}(0, \gamma_x^{-1} I_M)$ is the zero-mean Gaussian noise with precision value γ_x (I_M is an $M \times M$ Identity matrix). Following (Zhou, 2009), we model the matrix of the sparse codes (A) as an element-wise multiplication of a binary matrix (Z) and a weight matrix (S). Hence, the model of equation 6 can be reformulated as

$$X = D(Z \odot S) + E, \tag{7}$$

where $\odot$ is the element-wise multiplication operator. We put a prior distribution on the binary matrix Z using the extension of the Beta-Bernoulli process which takes two scalar parameters a and b and was originally proposed by (Paisley, 2009). A sample from the extended Beta process $B \sim BP(a, b, B_0)$ with base measure B_0 may be represented as

$$B_\omega = \sum_{k=1}^{K} \pi_k \delta_{\omega_k}, \tag{8}$$

where,

$$\pi_k \sim Beta(a/K, b(K-1)/K), \quad \omega_k \sim B_0. \tag{9}$$

This sample will be a valid sample from the extended Beta process, if $K \to \infty$. B_ω can be considered as a vector of K probabilities that each probability π_k corresponds to the atom ω_k. In our framework, we consider each atom ω_k as the k-th atom of the dictionary (d_k) and we set the base measure B_0 to a multivariate zero-mean Gaussian distribution $\mathcal{N}(0, \gamma_d^{-1} I_K)$ (with precision value γ_d) for simplicity. So, by letting $K \to \infty$, the number of the dictionary atoms can be learned from the training data. To model the weights $(\boldsymbol{s}_i)_{i=1}^N$, we use a zero-mean Gaussian distribution with precision value γ_s.

In order to make the dictionary discriminative for the classification purpose, we incorporate a logistic regression classifier to our probabilistic model. More precisely, if $\boldsymbol{\alpha}_t = \boldsymbol{z}_t \odot \boldsymbol{s}_t$ be the sparse code of a test instance x_t, the probability of $y_t = +1$ can be computed using the logistic sigmoid acting on a linear function of $\boldsymbol{\alpha}_t$ so that

$$P(y_t = +1 \mid \boldsymbol{z}_t, \boldsymbol{s}_t, \boldsymbol{w}, w_0) = \sigma(\boldsymbol{w}^T(\boldsymbol{z}_t \odot \boldsymbol{s}_t) + w_0), \tag{10}$$

where $\sigma(x)$ is the logistic function which is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{11}$$

As the probability of the two classes must sum to 1, we have $P(y_t = -1 \mid \boldsymbol{z}_t, \boldsymbol{s}_t, \boldsymbol{w}, w_0) = 1 - P(y_t = +1 \mid \boldsymbol{z}_t, \boldsymbol{s}_t, \boldsymbol{w}, w_0)$. Since the logistic function has the property that $\sigma(-x) = 1 - \sigma(x)$, we can write the class conditional probability more concisely as

$$P(y_t \mid \boldsymbol{z}_t, \boldsymbol{s}_t, \boldsymbol{w}, w_0) = \sigma\big(y_t[\boldsymbol{w}^T(\boldsymbol{z}_t \odot \boldsymbol{s}_t) + w_0]\big), \tag{12}$$

where $\boldsymbol{w} \in R^K$ and $w_0 \in R$ are the parameters of the classifier which are drawn from $\mathcal{N}(0, \gamma_w^{-1} I_K)$ and $\mathcal{N}(0, \gamma_w^{-1})$ respectively. We typically place non-informative Gamma hyper-priors on $\gamma_x, \gamma_d, \gamma_s$ and γ_w. The proposed hierarchical probabilistic model for the binary classification given the training data $(X, Y) = (\boldsymbol{x}_i, y_i)_{i=1}^N$, can be expressed as

$$P(X \mid D, Z, S, \gamma_x) \sim \prod_{j=1}^{N} \mathcal{N}(\boldsymbol{x}_j; D(\boldsymbol{z}_j \odot \boldsymbol{s}_j), \gamma_x^{-1} I_M), \tag{13}$$

$$P(\gamma_x \mid a_x, b_x) \sim Gamma(\gamma_x; a_x, b_x), \tag{14}$$

$$P(Z \mid \Pi) \sim \prod_{i=1}^{N} \prod_{k=1}^{K} Bernoulli(z_{ki}; \pi_k), \tag{15}$$

$$P(\Pi \mid a_\pi, b_\pi, K) \sim \prod_{k=1}^{K} Beta(\pi_k; a_\pi/K, b_\pi(K-1)/K), \tag{16}$$

$$P(S \mid \gamma_s) \sim \prod_{i=1}^{N} \prod_{k=1}^{K} \mathcal{N}(s_{ki}; 0, \gamma_s^{-1}), \tag{17}$$

$$P(\gamma_s \mid a_s, b_s) \sim Gamma(\gamma_s; a_s, b_s), \tag{18}$$

$$P(D \mid \gamma_d) \sim \prod_{i=1}^{M} \prod_{k=1}^{K} \mathcal{N}(d_{ik}; 0, \gamma_d^{-1}), \tag{19}$$

$$P(\gamma_d \mid a_d, b_d) \sim Gamma(\gamma_d; a_d, b_d), \tag{20}$$

$$P(Y \mid Z, S, \boldsymbol{w}, w_0) \sim \prod_{j=1}^{N} \sigma(y_j[\boldsymbol{w}^T(\boldsymbol{z}_j \odot \boldsymbol{s}_j) + w_0]), \tag{21}$$

$$P(\boldsymbol{w} \mid \gamma_w) \sim \mathcal{N}(\boldsymbol{w}; 0, \gamma_w^{-1} I_K), \tag{22}$$

$$P(\gamma_w \mid a_w, b_w) \sim Gamma(\gamma_w; a_w, b_w), \tag{23}$$

$$P(w_0 \mid \gamma_{w_0}) \sim \mathcal{N}(w_0; 0, \gamma_w^{-1}), \tag{24}$$

where $\Pi = [\pi_1, \pi_2, ..., \pi_K]$.
$\Phi = \{a_\pi, b_\pi, a_x, b_x, a_d, b_d, a_s, b_s, a_w, b_w, K\}$ are the hyper-parameters of the proposed model. The graphical representation of the probabilistic proposed model is shown in Fig. 1. For multiclass extension, we use the multinomial logistic regression classifier which is a model of the form

$$P(y_t = c \mid \boldsymbol{z}_t, \boldsymbol{s}_t, \Xi) = \frac{exp(\boldsymbol{w}_c^T(\boldsymbol{z}_t \odot \boldsymbol{s}_t))}{\sum_{c'=1}^{C} exp(\boldsymbol{w}_{c'}^T(\boldsymbol{z}_t \odot \boldsymbol{s}_t))}, \tag{25}$$

where C is the number of classes and $\Xi = [\boldsymbol{w}_1, ..., \boldsymbol{w}_C]$ are the parameters of the classifier which are drawn from multivariate zero-mean Gaussian distribution with precision value γ_w ($\boldsymbol{w}_c \sim \mathcal{N}(0, \gamma_w^{-1} I_K)$). The

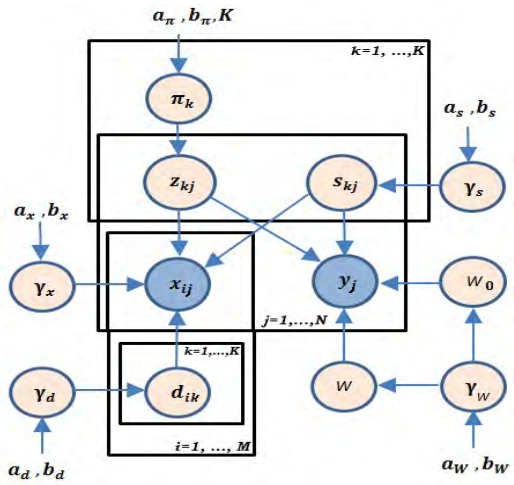


Figure 1: The graphical representation of the proposed binary classification model (blue shadings indicate observations).

hierarchical probabilistic model for the multiclass classification is the same as the model for the binary classification, except for the equations 21-24 which are replaced by

$$P(Y \mid Z, S, \Xi) \sim \prod_{j=1}^{N} \frac{exp\big(\boldsymbol{w}_{y_j}^T(\boldsymbol{z}_j \odot \boldsymbol{s}_j)\big)}{\sum_{c=1}^{C} exp(\boldsymbol{w}_c^T(\boldsymbol{z}_j \odot \boldsymbol{s}_j))}, \tag{26}$$

$$P(\Xi \mid \gamma_w) \sim \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{w}_c; 0, \gamma_w^{-1} I_K), \tag{27}$$

$$P(\gamma_w \mid a_w, b_w) \sim Gamma(\gamma_w; a_w, b_w). \tag{28}$$

4.2 Posterior Inference

Due to the intractability of computing the exact posterior distribution of the hidden variables, in this section, we derive a Variational Bayesian algorithm (Beal, 2003), to approximate the posterior distribution over the hidden variables of the proposed probabilistic model given the training data.
The goal of the variational inference is to approximate the true posterior distribution over the hidden variables with a variational distribution which is closest in KL divergence to the true posterior distribution. A brief review of the VB algorithm for the exponential family distributions provided in the Supplementary Material[1] (see appendix A).
In our variational inference framework, we use the finite Beta-Bernoulli approximation, in which the number of the dictionary atoms (K) is truncated and set

[1]The supplementary Material can be downloaded from http://ce.sharif.edu/~jourabloo/papers/SM.pdf

to a finite but large number. If K is large enough, the analyzed data using this number of dictionary atoms, will reveal less than K components.
In the following two sections, we derive the variational update equations for the binary and the multiclass classification models.

4.2.1 Variational Inference For the Binary Classification

In the proposed binary classification model, the hidden variables are

$$W = \Big\{ \Pi = [\pi_1, \pi_2, ..., \pi_K], Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N], \\ S = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_N], D = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K], \boldsymbol{w}, w_0, \\ \gamma_s, \gamma_w, \gamma_x, \gamma_d \Big\}.$$

We use a fully factorized variational distribution which is as follows

$$q(\Pi, Z, S, D, \boldsymbol{w}, w_0, \gamma_s, \gamma_w, \gamma_x, \gamma_d) = \\ \prod_{k=1}^{K} \prod_{i=1}^{M} q_{\pi_k}(\pi_k) q_{d_{ik}}(d_{ik}) \prod_{j=1}^{N} \prod_{k=1}^{K} q_{z_{kj}}(z_{kj}) q_{s_{kj}}(s_{kj}) \times \\ q_{\boldsymbol{w}}(\boldsymbol{w}) q_{w_0}(w_0) q_{\gamma_s}(\gamma_s) q_{\gamma_w}(\gamma_w) q_{\gamma_x}(\gamma_x) q_{\gamma_d}(\gamma_d).$$

It's worth noting that instead of using the parameterized variational distribution, we use the factorized form of the variational inference which is called Mean Field method (Beal, 2003). More precisely, we derive the form of the distribution $q(x)$ by optimizing the KL divergence over all possible distributions.
Based on the graphical model of Fig. 1, the joint probability distribution of the observations (training data) and the hidden variables can be expressed as

$$P(X, Y, W \mid \Phi) = \\ \prod_{j=1}^{N} \Big(P(x_j \mid \boldsymbol{z}_j, \boldsymbol{s}_j, D, \gamma_x) P(y_j \mid \boldsymbol{z}_j, \boldsymbol{s}_j, \boldsymbol{w}, w_0) \Big) \times \\ \prod_{k=1}^{K} \Big(P(\pi_k \mid a_\pi, b_\pi) \prod_{j=1}^{N} P(z_{kj} \mid \pi_k) P(s_{kj} \mid \gamma_s) \times \\ \prod_{i=1}^{M} P(d_{ik} \mid \gamma_d) \Big) P(\boldsymbol{w} \mid \gamma_w) P(w_0 \mid \gamma_w) P(\gamma_s \mid a_s, b_s) \times \\ P(\gamma_x \mid a_x, b_x) P(\gamma_w \mid a_w, b_w) P(\gamma_d \mid a_d, b_d). \quad (29)$$

In the binary classification model, all of the distributions are in the conjugate exponential family except for the logistic function. Due to the non-conjugacy between the logistic function and Guassian distribution, deriving the VB update equations in closed-form is intractable. To overcome this problem, we use the local lower bound to the sigmoid function proposed by (Jaakkola et al., 2000), which states that for any $x \in R$ and $\xi \in [0, +\infty]$

$$\frac{1}{1 + exp(-x)} \geq \sigma(\xi) exp\Big((x - \xi)/2 - \lambda(\xi)(x^2 - \xi^2) \Big), \quad (30)$$

where,

$$\lambda(\xi) = \frac{-1}{2\xi} \big(\frac{1}{1 + exp(-\xi)} - \frac{1}{2} \big). \quad (31)$$

ξ is the free variational parameter which is optimized to get the tightest possible bound. Hence, we replace each sigmoid factor in the joint probability distribution (equation 29) with the above lower bound (equation 30), then we use the EM algorithm to optimize the factorized variational distribution and the free parameters ($\boldsymbol{\xi} = \{\xi_1, \xi_2, ..., \xi_N\}$) which computes the variational posterior distribution in the E-step and maximizes the free parameters in the M-step. All update equations are available in the Supplementary Material (see appendix B).

4.2.2 Variational Inference For the Multiclass Classification

In the proposed multiclass classification model, because of non-conjugacy between the multinomial logistic regression function (equation 25) and the Gaussian distribution, deriving the VB update equations in closed-form is intractable. To tackle this non-conjugacy problem, we utilize the following simple inequality which was originally proposed by (Bouchard, 2007), which states that for every $\{\beta_c\}_{c=1}^{C} \in R$ and $\alpha \in R$,

$$log \big(\sum_{c=1}^{C} e^{\beta_c} \big) \leq \alpha + \sum_{c=1}^{C} log \big(1 + e^{\beta_c - \alpha} \big). \quad (32)$$

If we replace x with $\alpha - \beta_c$ in the equation 30, and take the logarithm of the both sides of that equation, we have

$$\log(1 + e^{\beta_c - \alpha}) \leq \lambda(\xi) \big((\beta_c - \alpha)^2 - \xi^2 \big) - \\ \log \sigma(\xi) + \big((\beta_c - \alpha) + \xi \big)/2. \quad (33)$$

Then, by replacing each term in the summation of the right hand side of the equation 32 with the upper bound of the equation 33, we have

$$\log \big(\sum_{c=1}^{C} e^{\beta_c} \big) \leq \alpha + \sum_{c=1}^{C} \log \big(1 + e^{\beta_c - \alpha} \big) \\ \leq \sum_{c=1}^{C} \Big(\lambda(\xi_c) \big((\beta_c - \alpha)^2 - \xi_c^2 \big) - \log \sigma(\xi_c) \Big) + \\ \alpha + \frac{1}{2} \sum_{c=1}^{C} (\beta_c - \alpha + \xi_c). \quad (34)$$

We utilize the above inequality for approximating the denominator of the right hand side of the equation 26. So, for the proposed multiclass classification model, the free parameters are $\{\{\alpha_i\}_{i=1}^{N}, \{\xi_{ij}\}_{i=1,j=1}^{N,C}\}$.
We derive an EM algorithm that computes the variational posterior distribution in the E-step and maximizes the free parameters in the M-step. Details of the update equations are available in the Supplementary Material (see appendix C).

4.3 Class Label Prediction

After computing the posterior distribution, in order to determine the target class-label y_t of a given test instance x_t, we first compute the predictive distribution of the target class label given the test instance by integrating out the hidden variables ($\{D, \gamma_x, \boldsymbol{z}_t, \boldsymbol{s}_t, \boldsymbol{w}, w_0\}$ for binary classification model, and $\{D, \gamma_x, \boldsymbol{z}_t, \boldsymbol{s}_t, [\boldsymbol{w}_c]_{c=1}^{C}\}$ for multiclass classification model), then we pick the label with the maximum probability value. For binary classification, this procedure can be formulated as

$$\hat{y}_t = argmax_{y_t \in \{-1,1\}} P(y_t \mid x_t, T), \tag{35}$$

where $T = (\boldsymbol{x}_j, y_j)_{j=1}^{N}$ is the training data. $P(y_t \mid x_t, T)$ can be computed as

$$\begin{aligned}
&P(y_t \mid x_t, T) = \\
&\sum_{\boldsymbol{z}_t} \iiint P(y_t, \boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{w}, w_0 \mid x_t, T) d\boldsymbol{s}_t \, d\boldsymbol{w} \, dw_0 \\
&= \sum_{\boldsymbol{z}_t} \iiint P(y_t \mid \boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{w}, w_0, x_t, T) \times \\
&\qquad P(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{w}, w_0 \mid x_t, T) d\boldsymbol{s}_t \, d\boldsymbol{w} \, dw_0 \\
&= \sum_{\boldsymbol{z}_t} \iiint \sigma(y_t[\boldsymbol{w}^T(\boldsymbol{s}_t \odot \boldsymbol{z}_t) + w_0]) P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T) \times \\
&\qquad P(\boldsymbol{w} \mid T) P(w_0 \mid T) d\boldsymbol{s}_t \, d\boldsymbol{w} \, dw_0 \\
&\approx \sum_{\boldsymbol{z}_t} \iiint \sigma(y_t[\boldsymbol{w}^T(\boldsymbol{s}_t \odot \boldsymbol{z}_t) + w_0]) P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T) \times \\
&\qquad q^*(\boldsymbol{w}) q^*(w_0) d\boldsymbol{s}_t \, d\boldsymbol{w} \, dw_0,
\end{aligned} \tag{36}$$

where we replaced $P(\boldsymbol{w} \mid T)$ and $P(w_0 \mid T)$ with the approximate posterior distributions $q^*(\boldsymbol{w})$ and $q^*(w_0)$ respectively.
Since the above expression cannot be computed in closed form, we resort to Monte Carlo sampling to approximate that expression. In other words, we approximate the distribution $P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, X, Y) q^*(\boldsymbol{w}) q^*(w_0)$ with l samples, then we compute $P(y_t \mid x_t, T)$ as

$$P(y_t \mid x_t, T) \approx \frac{1}{l} \sum_{l} \sigma(y_t[(\boldsymbol{w}^l)^T(\boldsymbol{s}_t^l \odot \boldsymbol{z}_t^l) + w_0^l]), \tag{37}$$

where r^l is the l-th sample of the hidden variable r.
Since the approximate posterior distributions $q^*(\boldsymbol{w})$ and $q^*(w_0)$ are Gaussian (see the appendix B of the Supplementary Material), sampling from these distributions is straightforward. $P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T)$ can be computed as

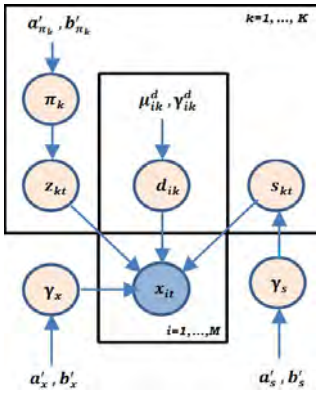


Figure 2: The graphical model of the Gibbs sampling method.

$$P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T) = \frac{P(x_t \mid \boldsymbol{s}_t, \boldsymbol{z}_t, T) P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid T)}{P(x_t \mid T)}, \tag{38}$$

which cannot be directly sampled from. Therefore, to sample from $P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T)$, we sample from $P(\boldsymbol{s}_t, \boldsymbol{z}_t, D, \Pi, \gamma_s, \gamma_x \mid x_t, T)$ based on the Gibbs sampling method (Robert et al., 2004), then simply ignore the values for $D, \Pi, \gamma_s, \gamma_x$ in each sample. The graphical model in Fig. 2 shows all the relevant parameters and conditional dependence relationships, by which the Gibbs sampling equations are derived. The detailes of the Gibbs sampling equations are avalable in the Supplementary Material (see appendix D). It should be noted that the parameters of the variables in Fig. 2 are the updated parameters of the variational posterior distribution which were computed using VB algorithm (see appendix B of the Supplementary Material). For multiclass extension, the posterior distribution over the class label of a test instance x_t can be approximated as

$$P(y_t = c \mid x_t, T) \approx \frac{1}{l} \sum_{l} \frac{exp\big((\boldsymbol{w}_c^l)^T(\boldsymbol{z}_t^l \odot \boldsymbol{s}_t^l)\big)}{\sum_{c'=1}^{C} exp((\boldsymbol{w}_{c'}^l)^T(\boldsymbol{z}_t^l \odot \boldsymbol{s}_t^l))}, \tag{39}$$

where $\{\boldsymbol{w}_c^l\}_{c=1}^{C}$ are the l-th samples of the approximate posterior distributions $\{q^*(\boldsymbol{w}_c)\}_{c=1}^{C}$, and $(\boldsymbol{z}_t^l, \boldsymbol{s}_t^l)$ is the l-th sample of the posterior distribution $P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T)$.
Sampling from $\{q^*(\boldsymbol{w}_c)\}_{c=1}^{C}$ is straightforward. Sampling from the distribution $P(\boldsymbol{s}_t, \boldsymbol{z}_t \mid x_t, T)$ for multiclass classification model is the same as sampling from that distribution for the binary classification model (see appendix D).

5 Experimental Results

In this section, we verify the performance of the proposed method on various applications such as digit recognition, face recognition, and spoken letter recognition. For applications which include more than two classes, we use one versus all binary classification (one classifier for each class) based on the proposed binary classification model (PM_b) as well as the proposed multiclass classification model (PM_m).
All of the experimental results are averaged over several runs of randomly generated splits of the data. Moreover, in all experiments, all Gamma priors are set as Gamma $(10^{-6}, 10^{-6})$ to make the prior distributions uninformative. The parameters a_π, b_π of the Beta distribution are set with $a_\pi = K$ and $b_\pi = K/2$ (many other settings of a_π and b_π yield similar results). For the Gibbs sampling inference, we discard the initial 500 samples (burn-in period), and collect the next 1000 samples to present the posterior distribution over the sparse code of a test instance.

5.1 Digit Recognition

We apply the proposed method on two handwritten digit recognition datasets MNIST (LeCun et al., 1998), and USPS (Hull, 1994). The MNIST dataset consists of 70000 28×28 images, and the USPS dataset composes of 9298 16×16 images. We reduced the dimensionality of both datasets by retaining only the first 100 principal components to speed up training. Details of the experiments for the digit databases are summarized in Table 1.
We compare the proposed models (PM_b, PM_m) with state of the art methods such as the Sparse Representation for Signal Classification (denoted by SRSC) (Huang et al., 2006), the supervised DL method with generative training and discriminative training (denoted by SDL-G and SDL-D) (Mairal, 2009), and the Fisher Discriminant Dictionary learning (denoted by FDDL) (Yang et al., 2011). Furthermore, the results of two classical classification methods, K-nearest neighbor (K=3) and linear SVM are also reported. The average recognition accuracies (over 10 runs) together with the standard deviation is shown in Table 2, from which we can see that the proposed methods outperform the other methods approximately by 3.5%.
The improvement in performance compared to other methods is due to the fact that the number of the training data points are small. Precisely speaking, the methods SRSC, SDL-G, SDL-D and FDDL are optimization based learners (MAP learners from probabilistic point of view) which can overfit small-size training data. In contrast, the proposed method does weighted averaging over the dictionary, the sparse codes, and the classifier, weighted by their posterior distributions and hence is relatively immune to overfitting. From Table 2, We also observe that the one versus all binary classifier (PM_b) has slightly better performance than the multiclass classifier (PM_m), but has more computational complexity than the multiclass classifier. Moreover, because of small number of the training data, generative SDL (SDL-G) has better performance than discriminative SDL (SDL-D).
In order to demonstrate the ability of the proposed method to learn the number of the dictionary atoms as well as the dictionary elements, we plot the sorted values of $\langle \Pi \rangle$ For the MNIST dataset, inferred by the algorithm (Fig. 3). As can be seen, the algorithm inferred a sparse set of factors, fewer than the 250 initially provided.
To further analyze the performance of the proposed method on various number of training data points, we illustrate the change in the classification accuracy on the MNIST digit dataset over successive iterations, for which we add more labeled samples at each iteration. Fig. 5 plots the recognition rates of different methods versus different number of training data points, from which we can see that improvement in the accuracy of the optimization based methods (FDDL, SDL-G, SDL-D) is larger than the proposed multi class classification method. This is due to the fact that when the number of the training data grows, the likelihood of overfitting the training data is reduced.
We also plot the sorted values of $\langle \Pi \rangle$ For the MNIST dataset for 1000 training data points, inferred by the algorithm (Fig. 4). As can be seen, when the number of the training data points increases, we need more dictionary atoms to capture the complexity of the data points.

Table 1: Properties of the digit datasets and experimental parameters

	MNIST	USPS
examples (train)	250	250
examples (test)	1000	1000
classes	10	10
input dimensions	784	256
features after PCA	100	100
runs	10	10
K (number of dictionary atoms)	250	250

5.2 Face Recognition

We then perform the face recognition task on the widely used extended Yale B (Lee et al., 2005), and AR (Martinez et al., 1998), face databases. The extended Yale B database consists of $2,414$ frontal-face

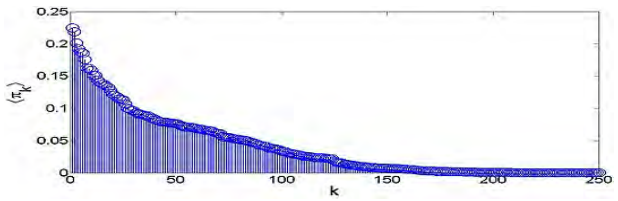


Figure 3: Inferred $\langle \Pi \rangle$ for the MNIST dataset (250 training samples).

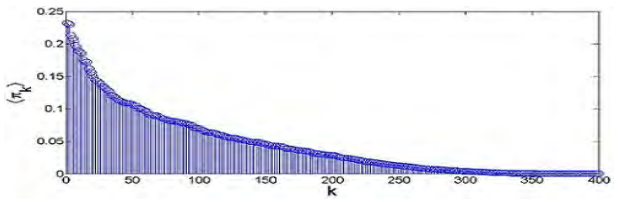


Figure 4: Inferred $\langle \Pi \rangle$ for the MNIST dataset (1000 training samples).

images from 38 individuals (about 64 images per individual), and the AR database consists of over 4,000 frontal images from 126 individuals which was generated in two sessions, each of them consists of 14 images per individual. The extended Yale B and AR images are normalized to 54×48 and 60×40 respectively. We use the Eigenface (Turk et al., 1991), with dimension 300 for both extended Yale B and AR datasets. For the extended Yale B database, each training set comprised of 20 images per individual, and the remaining images were used to test. For AR dataset, seven images from the first session are used for training, the remaining seven images from the second session are used for testing. Details of the experiments for the face databases are summarized in Table 3.

To illustrate the superiority of the proposed models, we compare our methods with the best result of discriminative KSVD (denoted DKSVD) (Zhang et al., 2010), dictionary learning with structure incoherence (denoted DLSI) (Ramirez et al., 2010), FDDL, K-NN, and SVM. The results of these experiments on the face databases are listed in Table 4. Again, due to the lack of enough number of the training data, our methods have better performance than the other methods.

5.3 Spoken Letter Recognition

Finally, we apply our method on the Isolet database (Blake et al., 1998), from UCI Machine Learning Repository which consists of 6238 examples and 26 classes corresponding to letters of the alphabet. We reduced the input dimensionality (originally at 617) by projecting the data onto its leading 100 principal components. We use 250 samples for training and 1000 samples for testing. The truncation level K for this experiment is set to 400. We also use only a subset of 10 classes of the Isolet dataset. The average recognition accuracies (over 10 runs) is shown in Table 5, from which we can see that the proposed methods outperform the other methods approximately by 3%.

Table 2: Classification accuracy of different methods on Digit datasets.

	MNIST	USPS
SVM	79.3± 2.0	80.7± 1.5
3-NN	80.4± 1.4	81.4± 2.1
SDL-D	80.2± 2.1	83.5± 1.9
SRSC	78.9± 1.2	80.2± 1.2
SDL-G	81.3± 1.4	84.0± 1.3
FDDL	81.1± 1.8	83.8± 1.7
PM_m	**84.9** ± 1.3	**86.6** ± 1.0
PM_b	**85.8** ± 1.1	**87.4** ± 0.9

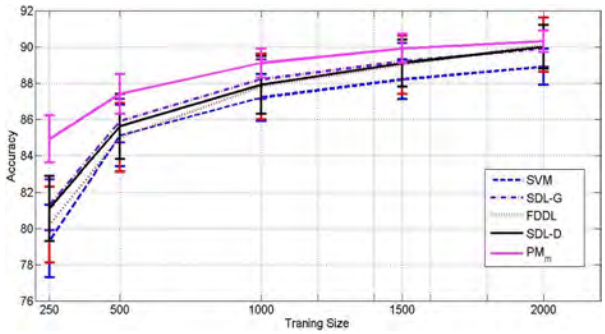


Figure 5: The recognition rate of different methods versus the number of training data for MNIST dataset.

6 Conclusion

We developed new models for the dictionary learning based pattern classification tasks based on the Beta-Bernoulli process, and a new algorithm based on the variational inference which allows our method scales to large data sets. We also used Bayesian prediction rule to determine the label of the unknown samples which

Table 3: Properties of data sets and experimental parameters.

	E-Yale B	AR
examples (train)	760	700
examples (test)	1654	700
classes	38	100
input dimensions	2592	2400
features after PCA	300	300
runs	10	10
K	500	600

Table 4: Classification accuracy of different methods on Face datasets.

	E-Yale B	AR
SVM	88.8± 1.2	87.1± 1.3
3-NN	65.9± 1.8	73.5± 2.1
DLSI	85.0± 1.6	73.7± 1.4
DKSVD	75.3± 1.4	85.4± 1.2
FDDL	91.9± 1.0	92.0± 1.3
PM_m	**94.7**± 1.3	**94.2**± 1.2
PM_b	**95.1**± 1.1	**94.9**± 1.0

Table 5: Classification accuracy of different methods on Isolet dataset.

Method	SVM	DLSI	FDDL	PM_b	PM_m
Accuracy	90.9	88.6	90.5	**93.3**	**92.9**

makes our method be suitable for small size training data. The experimental results on digit recognition, face recognition and spoken letter classification clearly demonstrated the superiority of the proposed model to many state-of-the-art dictionary learning based classification methods. For the future work, we will apply our method on the semi-supervised classification tasks.

References

M. Aharon, M. Elad, and A. Bruckstein (2006). k -svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 4311-4322.

M. Beal (2003). Variational algorithms for approximate bayesian inference. *Doctoral dissertation, University College London*.

C.L. Blake, and C.J. Merz (1998). Uci repository of machine learning databases. *University of California, Department of Information and Computer Science*.

D. Bohning (1992). Multinomial logistic regression algorithm. *Annals Inst. Stat. Math*.

G. Bouchard (2007). Efficient bounds for the softmax function.*In NIPS*.

N.L. Hjort (1990). Nonparametric bayes estimators based on beta processes in models for life history data. *Annals of Statistics*, 1259-1294.

K. Huang, and S. Aviyente (2006). Sparse representation for signal classification. *In NIPS*.

J. J. Hull (1994). A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell*, 550-554.

T. Jaakkola, and M. I. Jordan (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing*, 25-37.

J. F. C. Kingman (1993). Poisson Processes. *Oxford University Press*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE*.

K. Lee, J. Ho, and D. Kriegman (2005). Acquiring linear subspaces for face recognition under variable lighting. *In IEEE TPAMI*, 27(5): 684-698.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman (2009). Supervised dictionary learning. *In NIPS*.

A. Martinez, and R. benavente (1998). The AR face database. *CVC Tech. Report*.

J. Paisley, and L. Carin (2009). Nonparametric factor analysis with beta process priors. *In Proc. International Conference on Machine Learning*.

I. Ramirez, P. Sprechmann, and G. Sapiro (2010). Classification and clustering via dictionary learning with structured incoherence and shared features. *In CVPR*.

C.P. Robert, and G. Casella (2004). Monte carlo statistical methods. *Springer Verlag*.

R. Thibaux, and M. I. Jordan (2007). Hierarchical beta processes and the Indian buffet process. *In AISTATS*.

M. Turk, and A. Pentland (1991). Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71-86.

J. Wright, Y. Ma, J. Mairal, G. Sapiro, T.S. Huang, and S. Yan (2010). Sparse representation for computer vision and pattern recognition, *Proceedings of the IEEE*.

J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma (2009). Robust Face Recognition via Sparse Representation. *IEEE TPAMI*, 210-227.

M. Yang, L. Zhang, X. Feng, and D. Zhang (2011). Fisher discrimination dictionary learning for sparse representation. *In ICCV*.

M. Yang, L. Zhang, J. Yang, and D. Zhang (2010). Metaface learning for sparse representation based face recognition. *In ICIP*.

Q. Zhang, and B.X. Li (2010). Discriminative K-SVD for dictionary learning in face recognition. *In CVPR*.

M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin (2009). Non-Parametric Bayesian Dictionary Learning for Sparse Image Representations. *In NIPS*.

Identifying Learning Trajectories in an Educational Video Game

Deirdre Kerr
UCLA/CRESST
Peter V. Ueberroth Building (PVUB)
10945 Le Conte Ave., Suite 1400
Los Angeles, CA 90095-7150
dkerr@gseis.ucla.edu

Gregory K.W.K. Chung
UCLA/CRESST
Peter V. Ueberroth Building (PVUB)
10945 Le Conte Ave., Suite 1355
Los Angeles, CA 90095-7150
greg@ucla.edu

Abstract

Educational video games and simulations hold great potential as measurement tools to assess student levels of understanding, identify effective instructional techniques, and pinpoint moments of learning because they record all actions taken in the course of solving each problem rather than just the answers given. However, extracting meaningful information from the log data produced by educational video games and simulations is notoriously difficult. We extract meaningful information from the log data by first utilizing a logging technique that results in a far more easily analyzed dataset. We then identify different learning trajectories from the log data, determine the varying effects of the trajectories on learning, and outline an approach to automating the process.

1. INTRODUCTION

Computer games and simulations hold great potential as measurement tools because they can measure knowledge that is difficult to assess using paper-and-pencil tests or hands-on tasks (Quellmalz & Pellegrino, 2009). These measures can then be used to support diagnostic claims about students' learning processes (Leighton & Gierl, 2007), provide detailed measures of the extent to which players have mastered specific learning goals (National Science and Technology Council, 2011), and generate information that can be used to improve classroom instruction (Merceron & Yacef, 2004).

Log files from games can store complete student answers to the problems (Merceron & Yacef, 2004), allowing the researcher to record unobtrusively (Kim, Gunn, Schuh, Phillips, Pagulayan, & Wixon, 2008; Mostow, Beck, Cuneao, Gouvea, Heiner, & Juarez, 2011) the exact learning behavior of students (Romero & Ventura, 2007) that is not always captured in written or verbal explanations of their thought processes (Bejar, 1984).

Though log data is more comprehensive and more detailed than most other forms of assessment data, analyzing such data presents a number of problems because the log files typically include thousands of pieces of information for each student (Romero, Gonzalez, Ventura, del Jesus, & Herrera, 2009) with no known theory to help identify which information is salient (National Research Council, 2011). Additionally, the specific information stored in the log files is not always easy to interpret (Romero & Ventura, 2007) as the responses of individual students are highly context dependent (Rupp, Gusta, Mislevy, & Shaffer, 2010) and it can be very difficult to picture how student knowledge, learning, or misconceptions manifest themselves at the level of a specific action taken by the student in the course of the game. Due to these difficulties, there is currently no systematic approach to extracting relevant data from log files (Muehlenbrock, 2005). The interpretation of the rich stream of complex data that results from the tracking of in-game actions is one of the most serious bottlenecks facing researchers examining educational video games and simulations today (Mislevy, Almond, & Lukas, 2004).

1.1 RELATED WORK

Due to the difficulty involved in analyzing log data of students' in-game performance, educational researchers occasionally analyze student in-game performance by hand, despite the size of the data. Trained human raters have been used to extract purposeful sets of actions from game logs (Avouris, Komis, Fiotakis, Margaritis, &

Voyiatzaki, 2005) and logs of eye-tracking data (Conati & Merten, 2007). One study hand-identified student errors in log files from an introductory programming environment (Vee, Meyer, & Mannock, 2006) and another examined behavior patterns in an exploratory learning environment by hand to categorize students into learning types (Amershi & Conati, 2011). Another had the teacher play the role of a game character to score student responses and provide live feedback to the students (Hickey, Ingram-Goble, & Jameson, 2009).

Other studies avoided hand-coding log data by using easily extracted in-game measures such as percent completion or time spent on task to measure performance. The number of activities completed in the online learning environments *Moodle* (Romero, Gonzalez, Ventura, del Jesus, & Herrera, 2009) and *ActiveMath* (Scheuer, Muhlenbrock, & Melis, 2007) have been used to predict student grades. The time spent in each activity in an online learning environment has been used to detect unusual learning behavior (Ueno & Nagaoka, 2002). Combinations of the total time spent in the online environment and the number of activities successfully completed have been used to predict student success (Muhlenbrock, 2005) and report student progress (Rahkila & Karjalainen, 1999).

1.2 OUR CONTRIBUTION

In this study, we identify learning trajectories from information stored in log data generated by an educational video game. We do this by extracting the number of attempts required to solve each level (rather than the time spent or the number of levels completed) and then hand clustering the individual learning trajectories that result from plotting the attempts over time. We show that this process results in the identification of substantively different types of learning trajectories that differ on a variety of measures. We also discuss the benefits of our logging, preprocessing, and exploratory analysis techniques in regards to ease of interpretation and potential use in data mining techniques.

1.3 SAMPLE

This study uses data from 859 students who played an educational video game about identifying fractions called *Save Patch* in their classrooms for four days as part of a larger study. These students were given a paper-and-pencil pretest to measure their prior knowledge of fractions. After they played the game, students were given both an immediate posttest and a delayed posttest. The immediate posttest was computerized and was given on the last day of game play. The delayed posttest was a paper-and-pencil test that was given a few weeks later. All three tests consisted of both a set of content items and a set of survey items. In addition, the game generated log data consisting of each action taken by each student in the course of game play. The resulting dataset consisted of 1,288,103 total actions, 17,685 of which were unique.

2. DATA PREPARATION

The *Data Preprocessing and Intelligent Data Analysis* article (Famili, Shen, Weber, & Simoudis, 1997) lists eleven problems with real-world data that should be addressed in preprocessing. Our data comes from a single source, so we do not have to worry about merging data from multiple sources or combining incompatible data. The nine remaining problems and how they are applicable to our data are shown in Table 1.

Table 1: Potential Problems with *Save Patch* Data

PROBLEM	DESCRIPTION
Corruption and noise	Interruptions during data recording can lead to missing actions
Feature extraction	Important events must be identified from sets of individual actions
Irrelevant data	Not all actions taken in the game are meaningful
Volume of data	Hundreds or thousands of actions are recorded for each student
Missing attributes	Logs can fail to capture all relevant attributes
Missing attribute values	Logs can fail to record all values for all captured attributes
Numeric and symbolic data	Data for each action contains both numeric and symbolic components
Small data at a given level	We only have data for 859 students
Multiple levels	Data are recorded at multiple levels of granularity for each action

Our approach to minimizing the impact of these problems is explained in the following sections. Missing attributes are addressed in Section 2.1 (Game Design) and Section 2.2 (Logging). Corruption and noise, missing attribute values, numeric and symbolic data, and multiple levels are addressed in Section 2.2 (Logging). Feature extraction is addressed in Section 2.3 (Preprocessing), irrelevant data is addressed in Section 2.3.1 (Data Cleaning), and volume of data and small data at a given level are addressed in Section 3.1 (Exploratory Analysis).

2.1 GAME DESIGN

The educational video game used in this study is *Save Patch*. The development of *Save Patch* was driven by the findings that fluency with fractions is critical to performance in algebra (U.S. Department of Education, 2008), and that the understanding of fractions is one of the most difficult mathematical concepts students learn before algebra (Carpenter, Fennema, Franke, Levi, & Empson, 2000; McNeil & Alibali, 2005; National Council of Teachers of Mathematics, 2000; Siebert & Gaskin, 2006).

Once fractions concepts were identified as the subject area for the game, the most important concepts involved in fractions knowledge were analyzed and distilled into a set of knowledge specifications delineating precisely what students were expected to learn in the game (Vendlinski, Delacruz, Buschang, Chung, & Baker, 2010). These knowledge specifications, in turn, drove game design.

Because the game was designed specifically to measure student understanding of a predetermined set of knowledge specifications, both game mechanics and level design reflected those knowledge specifications and helped assure that all important attributes were measured in the game and recorded in the log files.

Figure 1: Example Level from *Save Patch*

In *Save Patch*, students must identify the fractional distances represented in each level, break ropes into pieces representing that distance, and place the correct number of rope pieces on each sign on the game grid to guide the puppet to the cage containing the prize. Units are represented by dirt paths and large gray posts, and small red posts break the units into fractional pieces. The level in Figure 1 is two units wide and one unit tall, and each unit is broken into thirds. To solve the level correctly, students must place four thirds on the first sign, one third on the second sign, and change the direction on the second sign so that it points down.

Save Patch is broken into stages based on content. All levels in a given stage represent the same fractions content. The game starts with whole number representations so that students can learn how to play, and then advances to unit fractions, whole numbers and unit fractions, proper fractions, and mixed numbers. After the fractions content stages, the game contains a test stage that was intended to be an in-game measure of learning and a series of challenge levels. The test stage includes an exact replica of one level from each of the previous stages and the challenge levels provide complicated combinations of the earlier material. This study focuses on the mixed numbers stage, because it contains the most complex representation of fractions in the game.

2.2 LOGGING

The data from *Save Patch* was generated by the logging technique outlined in Chung and Kerr (2012). As opposed to most log data from educational video games that consists of only summary information about student performance, such as the number of correct solutions or a probability that the content is known, the log data from this system consists of each action taken by each student in the course of game play.

However, such actions are not fully interpretable without relevant game context information indicating the precise circumstances under which the action was taken (Koedinger, Baker, Cunningham, Skogsholm, Leber, & Stamper, 2011). For this reason, each click that represented a deliberate action was logged in a row in the log file that included valuable context information such as the game level in which the action occurred and the time at which it occurred, as well as both general and specific information about the action itself.

As shown in Table 2, general information is stored in the form of a Data Code that is unique to each type of action (e.g., Data Code 3000 = selecting a rope piece from the Path Options). Each Data Code has a unique Description, for human readers and for documentation purposes, that identifies the action type and lists the interpretation of the following three columns. Data_01, Data_02, and Data_03 contain specific information about each action in the form of values that correspond to the bracketed information in the Description. For example, the third row in the table indicates that a rope was added (Data Code 3010) to the first sign (1/0 in Data_01), that the rope was a 1/3 piece (1/3 in Data_02), and that the resulting value on the sign was 1/3 (1/3 in Data_03). Additionally, the Gamestate records the values already placed on all signs in the level at the time of each action.

Logging the data in this manner allows for the easy interpretation of numeric and symbolic data because all comparable data is stored in the same format (e.g., 1/3 rather than .33) and because different representations of the same values have different interpretations in the game (e.g., 1/3 differs from 2/6). Additionally, the redundancy of carrying down each level of granularity (e.g., storing student ID and Level Number in each action) allows data to be recorded and analyzed at multiple levels without having to combine different datasets. This also reduces the negative effects of corruption and noise stemming from interruptions during data recording, because each action can be interpreted independently. Even if a given action is corrupted, all other actions in the level are still recorded correctly and each action contains all the information necessary for interpretation. While data corruption may result in missing attribute values in many

Table 2: Example Log Data from *Save Patch*

ID	Level	Game Time	Data Code	Description	Data_01	Data_02	Data_03	Gamestate
1115	14	3044.927	2050	Scrolled rope from [initial value] to [resulting value]	1/1	3/3		0/0_on_Sign1
1115	14	3051.117	3000	selected coil of [coil value]	1/3			0/0_on_Sign1
1115	14	3054.667	3010	added fraction at [position]: added [value] to yield [resulting value]	1/0	1/3	1/3	0/0_on_Sign1
1115	14	3058.443	3000	selected coil of [coil value]	1/3			1/3_on_Sign1
1115	14	3064.924	3010	added fraction at [position]: added [value] to yield [resulting value]	1/0	1/3	2/3	1/3_on_Sign1
1115	14	3088.886	3020	Submitted answer: clicked Go on [stage] – [level]	2	3		2/3_on_Sign1
1115	14	3097.562	3021	Moved: [direction] from [position] length [value]	Right	1/0	2/3	2/3_on_Sign1
1115	14	3106.224	4020	Received feedback: [type] consisting of [text]	Success	Congrat ulations!		2/3_on_Sign1
1115	14	3108.491	5000	Advanced to next level: [stage] – [level]	2	4		2/3_on_Sign1

other logging techniques, this is rarely the case with data logged in this manner because attribute values are recorded at the action level rather than calculated over time.

2.3 PREPROCESSING

The game design and logging techniques addressed a number of potential issues with the data, but it was still necessary to extract relevant features from the data.

In this study we were interested in examining student performance over time. In order to create these learning trajectories, we needed to identify a measure of performance in each level of the mixed numbers stage. Simply calculating whether students had correctly solved the level was insufficient, because students could replay a level as many times as was necessary and students could not advance to the next level without solving the current one. Therefore, we determined that the number of attempts it took a student to solve each level was the best measure of performance.

Attempts were not an existing feature of the log data, so each new attempt had to be calculated from existing information. We defined an attempt as all actions from the start of a level to either a reset of that level or advancing to the next level. The start of each attempt was identified using the following SPSS code, wherein Data Code 4010 indicates a reset:

If $casenum = 1 attempt = 1.
If id < > *lag*(id, 1) attempt = 1.
If curr_level < > *lag*(curr_level, 1) attempt = 1.
If *lag*(data_code, 1) = 4010 attempt = 1.

The first action in each attempt was then numbered consecutively using the following SPSS code:

Sort Cases By attempt(D) id curr_level uber_sn.
If id = *lag*(id,1) *and* attempt = 1
and curr_level = *lag*(curr_level, 1)
attempt = *lag*(attempt, 1) + 1.

Finally, the following SPSS code propagated the attempt number to all subsequent actions in that attempt:

Sort Cases By id curr_level uber_sn.
If attempt = 0 attempt = *lag*(attempt, 1).

2.3.1 Data Cleaning

Given the game design, logging technique, and preprocessing, little additional data cleaning was required after the attempts were calculated. However, irrelevant data still needed to be identified.

Irrelevant data in this analysis were defined as *invalid attempts*, which were attempts wherein students made no meaningful actions. In *Save Patch*, invalid attempts occurred largely because the student clicked reset twice in a row (either accidentally or due to impatience with the speed of the avatar) or because the student accidentally clicked "Go" immediately after a new level loaded (due to the initial location of the cursor directly above the "Go" button). If left in the dataset, these invalid attempts would

artificially inflate the number of attempts those students required to solve each level and thereby indicate a greater level of difficulty than was actually the case.

Invalid attempts were identified and dropped using the following SPSS code, wherein Code_3000 was a count of the number of times a rope was selected in that attempt:

Calculate DropAttempt = 0.
If Code_3000 = 0 DropAttempt = 1.
Select If DropAttempt = 0.

Remaining attempts were renumbered after all invalid attempts were dropped.

Additionally, a small number of students had not reached the portion of the game being analyzed. Approximately five percent of the students were dropped from the analysis because they had not reached the mixed numbers levels and therefore their learning trajectories for this content area could not be calculated.

3. EXPLORATORY ANALYSIS

Extracting the number of attempts each student required to solve each level reduced the dataset from over a million rows to only 21,713 rows of data (2,316 of which belonged to the subsample of students in the first 10% of the dataset, 413 of which occurred in the levels of interest). While this is too large of a volume of data for standard educational statistics, the data is also too small at this level for unsupervised, exploratory data mining techniques. Therefore, we decided to run some exploratory analyses to give us the information we would need to run a supervised data mining analysis.

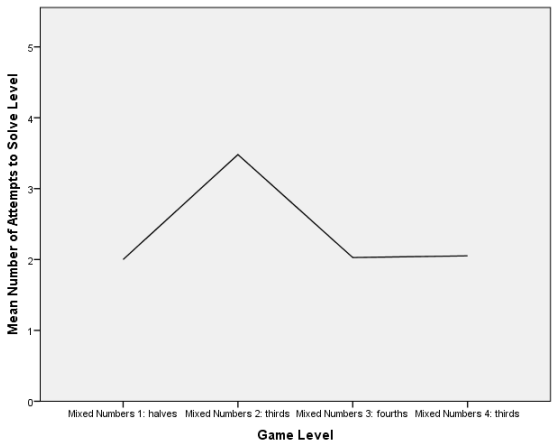


Figure 2: Mean Number of Attempts Per Level

An initial plot of the mean number of attempts students required to solve each of the mixed numbers levels is shown in Figure 2. This graph seems to indicate that the second level is more difficult than the other three levels, but does not otherwise seem to indicate any change in student performance as they move through the stage. Even given that the first level in the stage was designed as a training level and was intended to be much easier than other levels in the stage, it is difficult to make any claims about increased performance over time that might indicate student learning occurred. However, when examining performance curves over time, examining only mean values can hide more meaningful differences in learning trajectories between individuals (Gallistel, Fairhurst, & Balsam, 2004). Therefore, we decided to examine the individual learning trajectories of each of the students in our subset by hand.

3.1 IDENTIFYING LEARNING TRAJECTORIES

Only the first 10% of students in the sample was selected for the hand clustering dataset. The remaining 90% of the data was retained for subsequent data mining techniques. The individual learning trajectories for each of these 78 students were printed out. Similar to a hierarchical agglomerative clustering approach, we started with the first student's trajectory in a single cluster. Each subsequent student's trajectory was added to an existing cluster if it appeared substantively similar, or placed in a separate pile forming a new cluster if it appeared substantively different.

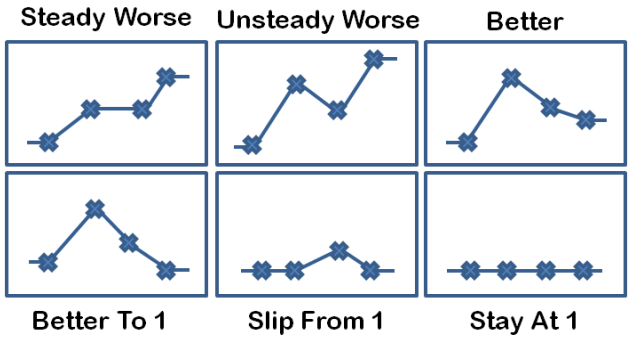


Figure 3: Identified Types of Learning Trajectories

The hand clustering resulted in six different groups of students, corresponding to six different types of learning trajectories (see Figure 3). The first type of learning trajectory demonstrated increasingly worse performance throughout the stage. In each consecutive level, these students (Steady Worse) took as many or more attempts to solve the level than they had required to solve the previous level. The second type of learning trajectory (Unsteady Worse) also demonstrated poorer performance later in the stage, but performed better on the third level in the stage than they had on the second level in the stage, resulting in a more ragged uphill trajectory.

The third type of learning trajectory (Better) performed consistently better on each of the last three levels of the stage, and the fourth type of learning trajectory (Better To

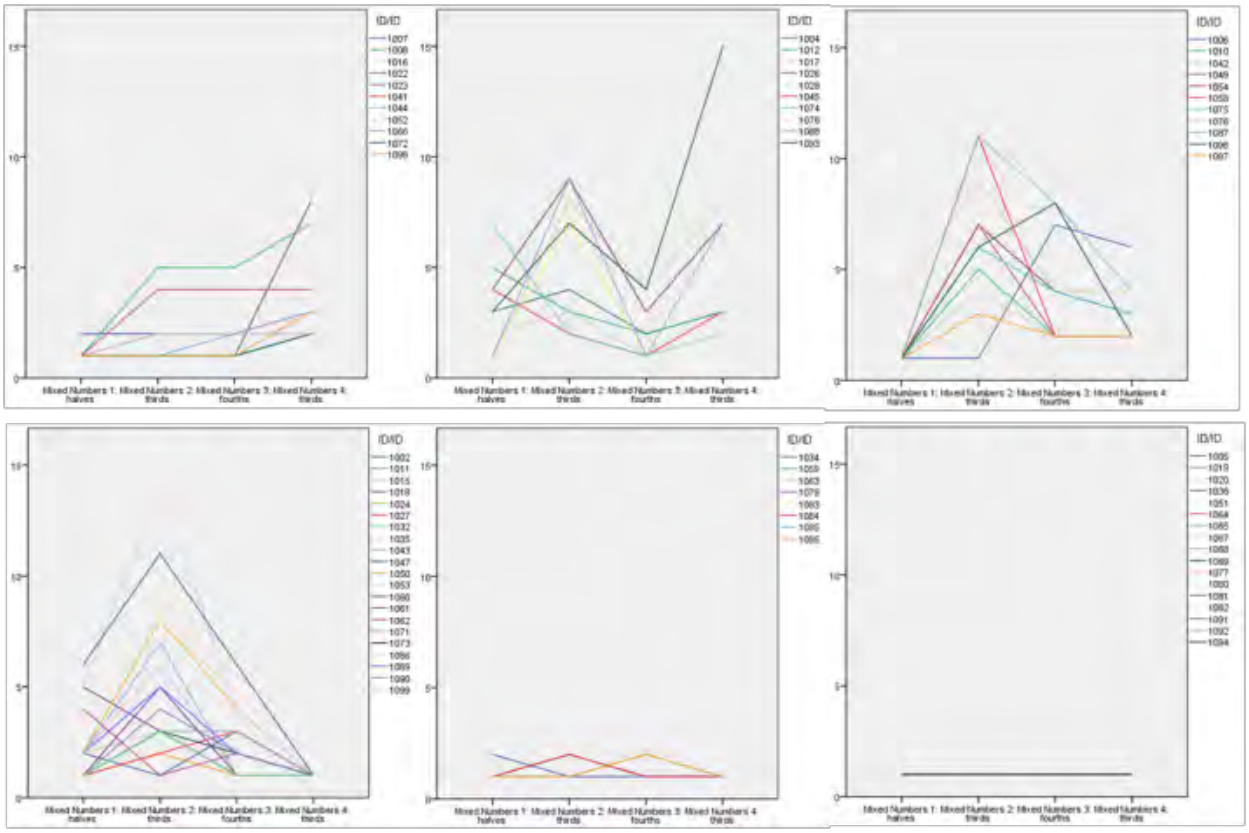


Figure 4: Student Learning Trajectories by Type

1) improved consistently better on the last three levels to the point that they solved the final level in their first attempt. The fifth type of learning trajectory (Slip From 1) solved all levels in the stage on their first attempt, except for one level which they took two attempts to solve. The sixth type of learning trajectory (Stay At 1) solved all levels in the stage on their first attempt, making no mistakes at all.

The individual learning trajectories for each student are plotted in Figure 4. The top three graphs represent (from left to right) students in the Steady Worse, Unsteady Worse, and Better learning trajectory types. The bottom three graphs represent students in the Better To 1, Slip From 1, and Stay At 1 learning trajectory types.

3.2 FINDING DIFFERENCES

In order to determine whether the learning trajectories were substantively different, and therefore worth further analysis, a number of exploratory ANOVAs were run

Students in the six different learning trajectory types differed significantly on both prior knowledge measures: the pretest score ($p < .001$) and prior math grades ($p = .024$). Slip From 1 and Stay At 1 had the highest mean pretest scores (4.42 and 4.22 respectively) and Unsteady Worse had the lowest (1.17). Similarly, Slip From 1 had the highest mean prior math grades (1.0 where 1 is an A) and Unsteady Worse and Better had the lowest (2.17 and 2.50 respectively). See Table 3 for results.

The learning trajectory types also differed significantly on in-game performance measures. There were significant differences between types in the percent of game levels completed ($p < .001$), but not the time they spent playing ($p = .889$), with Slip From 1 and Stay At 1 having the highest mean percentage of levels completed (84% and 87% respectively) and Better having the lowest (65%).

There were also significant differences in the percentage of students in the group solving the mixed numbers test level in their first attempt ($p < .001$) and in improvement between their performance on the corresponding level in the mixed numbers stage and the test level ($p < .001$). All students in Slip From 1 solved the mixed numbers test level on their first attempt (as did 82% of Stay At 1 students). Only 20% of Unsteady Worse, and none of the Better students, solved the mixed numbers test level on their first attempt. However, the Better, Better To 1, and

Table 3: ANOVA Results

MEASURE	SIGNIFICANCE	BEST MEANS	WORST MEANS
Pretest Score	$p < .001$	Slip From 1 (4.42) Stay At 1 (4.22)	Unsteady Worse (1.17)
Prior Math Grades	$p = .024$	Slip From 1 (1.0)	Unsteady Worse (2.17) Better (2.50)
Number of Game Levels Completed	$p < .001$	Stay At 1 (87%) Slip From 1 (84%)	Better (65%)
Time Spent Playing	$p = .889$	*no difference*	*no difference*
Solved Test Level on First Attempt	$p < .001$	Slip From 1 (100%) Stay At 1 (82%)	Unsteady Worse (20%) Better (0%)
Improve on Test Level	$p < .001$	Better (3.18) Unsteady Worse (2.80) Better To 1 (2.48)	Know (-0.18) Worse (-0.80)
Immediate Posttest	$p = .012$	Stay At 1 (5.78) Slip From 1 (5.50)	Unsteady Worse (2.71)
Delayed Posttest	$p = .010$	Stay At 1 (5.84) Slip From 1 (4.75)	Unsteady Worse (2.82)
Self-Belief in Math Before the Game	$p = .221$	*no difference*	*no difference*
Self-Belief in Math After the Game	$p = .022$	Stay At 1 (3.44) Slip From 1 (3.21)	Steady Worse (2.57) Unsteady Worse (2.33)

Unsteady Worse students all showed improvement between the corresponding level in the stage and the test level, taking an average of 3.18, 2.48, and 2.80 fewer attempts respectively to solve the test level.

Students in the different learning trajectory types also differed significantly on the immediate posttest ($p = .012$) and delayed posttest ($p = .010$), retaining most of the significant differences present in the pretest measures. As with the pretest, Slip From 1 and Stay At 1 had the highest mean immediate posttest scores (5.50 and 5.78 respectively) and delayed posttest (4.75 and 5.84), and Unsteady Worse had the lowest immediate posttest (2.71) and delayed posttest (2.82). However, the learning trajectory types also differed in their self-belief in math after the game ($p = .022$), though there was no significant difference before the game ($p = .221$). Slip From 1 and Stay At 1 had highest self-belief in math after the game (3.21 and 3.44 respectively), followed by Better and Better To 1 (3.07 and 2.82 respectively), with Steady Worse and Unsteady Worse having the lowest self-belief in math (2.57 and 2.33 respectively).

4. NEXT STEPS

Now that the six different learning trajectory types have been identified and evidence exists that the differences between the groups are substantive, the next step in our research is to test different cluster analysis techniques to determine which one best classifies students into these groups.

However, the accuracy of a cluster analysis technique depends, at least in part, on the appropriateness of the attributes used to create the distance matrix it operates on. There are three possible sets of attributes that might be used. First, the learning trajectories could be seen as splines. In this case, the attribute set would consist of the spline values, initial values, and ending values of each trajectory.

On the other hand, it might be more appropriate to treat the learning trajectories as a series of connected line segments. In this case, the attribute set would consist of the initial value, slope, and ending value of each line segment in each learning trajectory.

However, examination of the learning trajectories plotted in Figure 4 indicate that the value of each point may not be as important in determining which cluster a given learning trajectory falls in as the general shape of the trajectory. In this case, the attribute set would consist of a binary indicator of whether or not the initial value of each line segment was 1 or more than 1, a binary indicator of whether or not the ending value of each line segment was 1 or more than 1, and a set of binary indicators of whether the slope of each line segment was positive, negative, or neutral. These three options are summarized below.

1. Splines: initial value, spline values, ending value
2. Line Segments: initial value, slope, ending value
3. Binary Line Segments: initial value of 1 or more than 1, positive, negative, or neutral slope, ending value of 1 or more than 1

The distance matrix created from each of these three attribute sets will be fed into a hierarchical, partitioning, and fuzzy clustering algorithm. This will result in nine clustering techniques. Each of these clustering techniques will be run over the 10% of students whose learning trajectories have already been hand clustered in order to determine which technique best classifies the students.

Once the best clustering technique has been identified, it will be used to classify the remaining 90% of students in the sample into the learning trajectory type which best describes their in-game performance. Then a MANOVA will be run to determine which learning trajectory types differ on which measures across the entire sample (as opposed to the 10% reported in Table 3). If differences are found, the clustering technique could then be used (without requiring additional manual analysis) on attempt data from other stages in *Save Patch*, other *Save Patch* data collections, or other stages in similar games.

5. DISCUSSION

The logging technique used in this study resulted in a dataset that eased preprocessing and feature extraction. Additionally, the hand clustering led to the identification of six different types of learning trajectories who differed substantively on measures of prior knowledge, in-game performance, and posttest performance.

Perhaps the most interesting types of learning trajectories are the Better To 1, Better, and Unsteady Worse types. These trajectories appear to identify the potential learners for a given game, students who don't know the material but are capable of learning from the game play. In contrast, the Stay At 1 and Slip From 1 trajectory types seem to identify students who already know the material and the Steady Worse trajectory type seems to identify students who do not know the material and are not learning from the game.

The results of this study seem to indicate that using data mining techniques to cluster learning trajectories would be a worthwhile endeavor, as the different clusters appear to correspond to substantively different groups of students. If the data mining results support the results of this study, it would not only support claims that educational video games and simulations can be used as stand-alone measures of student knowledge, but also provide the designers of those games with the information about which students' needs are being met by the game.

However, it is possible that the findings of this study will not be supported by the data mining. This is only partially because the data mining might classify students differently than the hand clustering, and is mostly due to the fact that the small sample size in the hand clustered subset combined with the use of multiple ANOVAs rather than a single MANOVA might have identified some differences between learning trajectory types that occurred merely by chance. Currently, this study represents a promising process for analyzing data from educational video games, but the specific findings about performance should not be considered definitive without support from further studies.

Acknowledgements

The work reported herein was supported under the Educational Research and Development Centers Program, PR/Award Number R305C080015. The findings and opinions expressed here do not necessarily reflect the positions or policies of the National Center for Education Research, the Institute of Education Sciences, or the U.S. Department of Education.

References

Amershi, S., & Conati, C. (2011). Automatic recognition of learner types in exploratory learning environments. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S.J.d. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 389-416). Boca Raton, FL: CRC Press.

Avouris, N, Komis, V., Fiotakis, G., Margaritis, M., & Voyiatzaki, E. (2005). Logging of fingertip actions is not enough for analysis of learning activities. In *Proceedings of the Workshop on Usage Analysis in Learning Systems at the 12th International Conference on Artificial Intelligence in Education*.

Bejar, I. I. (1984). Educational diagnostic assessment. *Journal of Educational Measurement*, *21*(2), 175-189.

Carpenter, T. P., Fennema, E., Franke, M. L., Levi, L. W., & Empson, S. B. (2000). *Cognitively Guided Instruction: A Research-Based Teacher Professional Development Program for Elementary School Mathematics*. Madison, WI: National Center for Improving Student Learning and Achievement in Mathematics and Science.

Chung, G. K. W. K., & Kerr, D. (2012). *A primer on data logging to support extraction of meaningful information from educational games: An example from Save Patch* (CRESST Report 814). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing.

Conati, C., & Merten, C. (2007). Eye-tracking for user modeling in exploratory learning environments: An empirical evaluation. *Knowledge Base Systems*, *20*(6), 557-574.

Famili, F., Shen, W. M., Weber, R., & Simoudis, E. (1997). Data pre-processing and intelligent data analysis. *International Journal on Intelligent Data Analysis*, *1*(1), 3-23.

Gallistel, C. R., Fairhurst, S., & Balsam, B. (2004). The learning curve: Implications of a quantitative analysis. *Proceedings of the National Academy of Sciences*, *101*(36), 13124-13131.

Hickey, D. T., Ingram-Goble, A. A., & Jameson, E. M. (2009). Designing assessments and assessing designs in virtual educational environments. *Journal of Science Education and Technology*, *18*, 187-208.

Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of the 26th annual SIGCHI Conference on Human Factors in Computing Systems* (pp. 443-452).

Koedinger, K. R., Baker, R. S.J.d., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2011). A data repository for the EDM community: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S.J.d. Baker (Eds.) *Handbook of Educational Data Mining* (pp. 43-55). Boca Raton, FL: CRC Press.

Leighton, J. P., & Gierl, M. J. (2007). Defining and evaluating models of cognition used in educational measurement to make inferences about examinees' thinking processes. *Educational Measurement: Issues and Practice*, *26*(2), 3-16.

McNeil, N. M., & Alibali, M. W. (2005). Why won't you change your mind? Knowledge of operational patterns hinders learning and performance on equations. *Child Development*, *76*(4). 883-899.

Merceron, A., & Yacef, K. (2004). Mining student data captured from a web-based tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research*, 15, 319-346.

Mislevy, R. J., Almond, R. G., & Lukas, J. F. (2004). *A brief introduction to evidence centered design* (CSE Report 632). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing.

Mostow, J., Beck, J. E., Cuneao, A., Gouvea, E., Heiner, C., & Juarez, O. (2011). Lessons from Project LISTEN's session browser. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S.J.d. Baker (Eds.), *Handbook of educational data mining* (pp. 389-416). Boca Raton, FL: CRC Press.

Muehlenbrock, M. (2005) Automatic action analysis in an interactive learning environment. In Choquet, C., Luengo, V. and Yacef, K. (Eds.), *Proceedings of the workshop on Usage Analysis in Learning Systems at AIED-2005*.

National Council of Teachers of Mathematics. (2000). *Principles and standards for school mathematics*. Reston, VA.

National Research Council. (2011). *Learning science through computer games and simulations*. Washington, DC: The National Academies Press.

National Science and Technology Council (2011). *The federal science, technology, engineering, and mathematics (STEM) education portfolio*. Washington, DC: Executive Office of the President.

Quellmalz, E. S., & Pellegrino, J. W. (2009). Technology and testing. *Science*, *323*, 75-79.

Rahkila, M., & Karjalainen, M. (1999). Evaluation of learning in computer based education using log systems. In *Proceedings of 29th ASEE/IEEE Frontiers in Education Conference (FIE '99)* (pp. 16-22).

Romero, C., Gonzalez, P., Ventura, S., del Jesus, M. J., & Herrera, F. (2009). Evolutionary algorithms for subgroup discovery in e-learning: A practical application using Moodle data. *Expert Systems with Applications*, *39*, 1632-1644.

Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, *35*, 135-146.

Rupp, A. A., Gushta, M., Mislevy, R. J., & Shaffer, D. W. (2010). Evidence centered design of epistemic games: Measurement principles for complex learning environments. *The Journal of Technology, Learning, and Assessment*, *8*(4).

Scheuer, O., Muhlenbrock, M., & Melis, A. (2007). Results from action analysis in an interactive learning environment. *Journal of Interactive Learning Research*, *18*(2), 185-205.

Siebert, & Gaskin (2006). Creating, naming, and justifying fractions. *Teaching Children Mathematics*, *12*(8), 394-400.

U.S. Department of Education (2008). *Foundations for success: The final report of the National Mathematics Advisory Panel*. Washington, DC.

Ueno, M. & Nagaoka, K. (2002). Learning log database and data mining system for e-learning: On line statistical outlier detection of irregular learning processes. In *Proceedings of the International Conference on Advanced Learning Technologies* (pp. 436-438).

Vee, M. N., Meyer, B., & Mannock, M. L. (2006). Understanding novice errors and error paths in Object-oriented programming through log analysis. In *Proceedings of the Workshop on Educational Data Mining* (pp. 13-20).

Vendlinski, T. P., Delacruz, G. C., Buschang, R. E., Chung, G. K. W. K., & Baker, E. L. (2010). *Developing high-quality assessments that align with instructional video games* (CRESST Report 774). Los Angeles, CA, University of California, National Center for Research on Evaluation, Standards, and Student Testing.

Transforming Personal Artifacts into Probabilistic Narratives

Setareh Rafatirad
Department of
Applied Information Technology
George Mason University
Fairfax, VA 22030
srafatir@gmu.edu

Kathryn B. Laskey
Department of
Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030
klaskey@gmu.edu

Abstract

An approach focused on inferring probabilistic narratives from personal artifacts (including photographs) is presented in this work using personal photos metadata (timestamp, location, and camera parameters), formal event models, mobile device connectivity, external data sources and web services. We introduce plausibility measure — the occurrence-likelihood of an event node in the output graph. This measure is used to find the best event among the merely possible candidates. In addition, we propose a new clustering method that uses timestamp, location, and camera parameters in the EXIF header of the input photos to create event boundaries used to detect events.

1 INTRODUCTION

The technology of current smart phones comes with multiple sensors like camera, and GPS, which enables the device to record time, GPS location, and camera parameters with the photo's EXIF header. There is a high-demand for searching through personal photo archives to relive the events evidenced by the photos. Annotating personal artifacts with expressive tags supports this demand. We propose a technique that automatically creates a context-aware event graph by combining event models with contextual information related to personal photos and information, and heterogeneous data sources. Our technique automatically computes the occurrence-likelihood for the event nodes in the output graph; we refer to this value as plausibility measure. Events are key cues to recall personal photos (Naaman,2004); they can be used to create searchable description metadata for them. Events, in general, are structured and their subevents have relatively more expressive power (Rafatirad, 2009), e.g., the event *Giving a Talk* is more expressive than its superevent, *Professional Trip*. In addition, instance events are contextual and should be augmented with context cues (like place, time, weather). This makes instance events more expressive than event types. For example, the instance event *Giving a Talk at UCF at most two hours before meeting with Ted on a windy day* is far more expressive than the event type *Giving a Talk*. We define flexible expressiveness as follows: *a)* multi-granular conceptual description: provides conceptual hierarchy in multiple levels using containment event relationships e.g. *subevent-of*, *subClassOf*; *b)* multi-context adaptation of conceptual description: adapts a concept to multiple contextual descriptions (e.g., event type *visit-landmark* may have two instances; one instance associated with *Forbidden City* and the other to *Great Wall of China*). Consider the following example: A person takes a photograph at an airport less than 1 hour after his flight arrives. To explain this photograph, we first need the background knowledge about the events that generally occur in the domain of a trip. These semantics can only come from an event-ontology that provides the vocabulary for event/entity and event relationships related to a domain. An event-ontology allows explicit specification of models that could be modified using context information to provide very flexible models for high-level semantics of events. We refer to this modification as *Event Ontology Augmentation* or *EOA*. It constructs a more robust and refined version of an event-ontology either fully or semi-automatically. Secondly, given the uncertain metadata of a photo (like GPS that is not always accurate), the event type that the photo witnesses is not decisive; it might either be *rent a car*, or *baggage claim* that are two possible conclusions — sometimes no single obvious explanation is available, but rather, several competing explanations exist and we must select the best one. In this work, reasoning from a set of incomplete information (observations) to the most related conclusion out of all possible ones (explanations) is performed through a ranking algorithm that incorporates the plausibility measure; this ranking process is used in *EOA*.

Problem Formulation

We assume that every input photo has context information (specifically, timestamp, location, and camera parameters) and a user/creator. Each photo belongs to a photo stream P of an event with a basic domain event-ontology $O(V, E)$ whose nodes (V) are event/entity classes, and edges (E) are event/entity relationships, handcrafted by a group of domain experts. We assume that there is a bucket (B) in which external data sources are represented with a schema. The sources in B can be queried using the metadata of the input photographs and information about the associated user. Given P, B, O, and information associated to the user, how does one find the finest possible event tag that can be assigned to a photo or a group of similar photographs in P?

Solution Strategy: We propose Event Ontology Augmentation (EOA) technique described as follows: select a relevant domain event ontology $O(V, E)$ through the information related to both the user and P. Using P, B, O, and the user information, infer S that consists of the best relevant subevent categories to P where $S \subseteq V$. An event category in S is the most plausible one among other competing candidates that have failed to be selected. For each competing candidate s_i, a plausibility measure m_{ij}^p is calculated using function f to rank s_i and indicate how much it is relevant to c_j such that $c_j \subset P$ and c_j is a group of similar photographs: $f(s_i, c_j) = m_{ij}^p$. Next, augment S using the information from B to obtain expressive event tags T. We define an event tag $t_i^e \in T$ as a subevent of an event that either exists in O, or can be derived from O such that t_i^e is the finest subevent tag that can be assigned to a group of similar photos. Also, if t_i^e is an assignable tag to any photo, and t_i^e does not exist in O, we intend to augment O by adding t_i^e to O using the shortest composition path such that the constraints governing O are preserved. Simply put, the final step is adding T to O by preserving the rules that govern O if $T \notin O$. The output is an extension to O that is referred as O_r. We argue that O_r (see fig 1) can be used for an event recognition task in photo annotation applications. The key insight in our proposed approach is to infer event characteristics from the image metadata, information about the user, ontological event model, mobile device connectivity, web services, and external data sources. We argue that attribute values related to an inferred event need to be obtained, refined, and validated as much as possible to create very expressive and reliable metadata for digital photographs and facilitate image search and retrieval. Fig 4 depicts the processing components of our proposed approach in the context of personal photo annotation. Several event semantics are utilized in this work like spatiotemporal attributes/constraints of events, subevent structure, and spatiotemporal proximity. Unlike machine learning approaches that are limited to the training data set and require an extensive amount of annotation, we propose a technique in which existing knowledge sources are modified and expanded with context information in personal data sources (like Google Calendar, and social interactions), public data sources (like public event/weather directories, local business databases), and digital media archives (like personal photographs). With this knowledge expansion, new infrastructures are constructed to serve relevant data to communities. Event tags are propagated with event title, place information (like city, category, place name), time, weather, etc. Our proposed technique provides two unique key benefits as follows: *1)* A sufficiently flexible structure to express context attributes for events such that the attributes are not hardwired to events, but rather they are discovered on the fly. This feature does not limit our approach to a single data set; *2)* leveraging context data across multiple sources could facilitate building a consistent, unambiguous knowledge base.

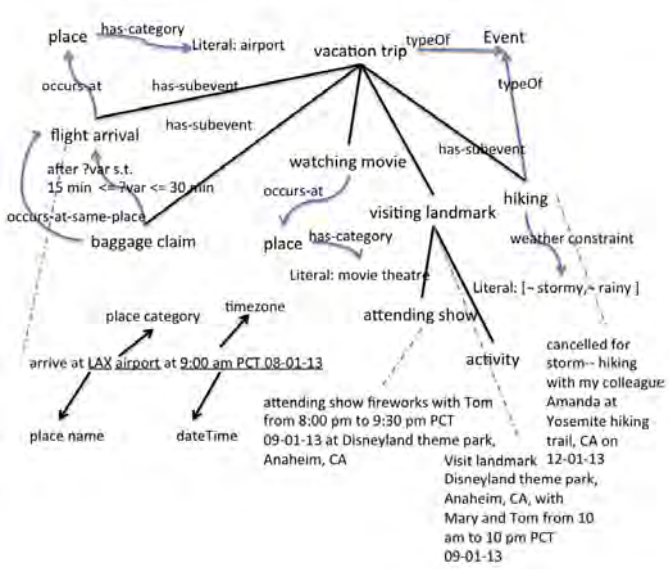


Figure 1: An Example of Augmented Event Ontology.

EOA has several challenges: *a)* we need a language that can model different types of entity properties and relationships related to a domain. OWL is widely used for developing ontologies. However, this language is limited in terms of its capability of describing the semantics of events. A major challenge is to create an extension of OWL and provide the grammar for that extension; *b)* collecting and combining information from multiple sources is a daunting task. It needs a general mechanism to automatically query sources and represent the output. It also needs a validation mechanism to ensure the coherency of the obtained data; *c)* currently, publicly available benchmark data sets such as those offered by TRECVid do not suit the purpose of this research (they deal with low level events

i.e.,activities). However, higher-level events have relatively more contextual characteristics; *d)* according to the useful properties of photoset, relevant event categories in the model must be discovered. This paper is organized as follows: in section 2, we review the prior art for annotating photos; in section 3 we explain our clustering method; in section 4 and 5, our solution strategy is explained; followed by section 6 that demonstrates our experiments, and section 7 which is the conclusion.

2 RELATED WORK

The important role of context in image retrieval is emphasized in (Datta, 2008) and (Jain, 2010). Context information and ontological event models are used in conjunction by (Viana, 2007,2008), (Fialho, 2010). (Cao, 2008) presents an approach for event recognition in image collections using image timestamp, location, and a compact ontology of events and scenes. In this work, event tags do not address the subevents of an event. (Liu, 2011) reports a framework that converts each event description from existing event directories (like Last.fm) into an event ontology that is a minimal core model for any general event. This approach is not flexible to describe domain events (like 'trip') and their structure (like 'subevent' structure). (Paniagua, 2012) propose an approach that builds a hierarchy of events using the contextual information of a photo based on moving away from routine locations, and string analysis of English album titles (annotated by people) for public web albums in Picasaweb. The limitations of this approach are: 1) human-induced tags are noisy, and 2) subevent relationship is more than just spatiotemporal containment. For instance, albeit a 'car accident' may occur in the spatiotemporal extent of a 'trip', it is not part of the subevent-structure of the 'trip'. According to (Brown, 2005), events form a hierarchical narrative-like structure that is connected by causal, temporal, spatial and subevent relations. If these aspects are carefully modeled for events, they can be used to create a descriptive knowledge base for interpreting multimedia data. The importance of building event hierarchies is also addressed in (Rafatirad, 2009) where the main focus is on the issues of event composition using the subeventOf relationship between events. In (Rafatirad, 2011), an image annotation mechanism is proposed that exploits context sources in conjunction with subevent-structure of an event. The limitation of this approach is no matter how much an event category is relevant to a group of photos in a photo stream, it is used in photo annotation. As a result of this operation, the quality of annotation degrades.

3 CLUSTERING

We consider two images to be similar if they belong to the same type of event. Partitioning a photo stream based on the context of its digital photographs can create separate event boundaries for the photos related to one event (Pigeau, 2005). An event is a temporal entity. However, using time as the only dimension in clustering means ignoring other context semantics about events. Much better results can be obtained with time and location information. (Gong, 2008) propose a framework for photo stream from single user that applies hierarchical mixture of Gaussian models based on context information including time, location, and optical camera parameters (such as ISO, Focal Length, Flash, Scene Capture Type, Metering Mode, and Subject Distance). In photos, optical camera parameters provide useful information related to the environment at which an event occurs, like 'indoor', 'outdoor', and 'night' (Sinha, 2008). We propose an agglomerative clustering that partitions a photo stream hierarchically according to the context information of photos, specifically timestamp, location, and Optical Camera Parameters (*OCP*). Agglomerative clustering has several advantages; it is (a) fully unsupervised, (b) applicable to any attribute types, and (c) clusters can be formed flexibly at multiple levels (from coarser to finer). In general, larger events like 'trip' are often described using spatiotemporal characteristics whereas the subevent structure is limited by space and time. However, the depth of a spatiotemporal agglomerative clustering dendrogram can be extended using OCP to refine the precision of the clusters. Our clustering approach is described as follows: primarily, a photo stream is partitioned using timestamp, gps-latitude, and gps-longitude; the blue cluster structure in Fig 2, referred as *ST-cluster tree*, shows the output for this stage of the clustering. Next, for each ST-cluster in the blue structure, its content is partitioned based on OCP to create *ST-OCP cluster tree*. The orange structure in fig 2 shows the output of this stage. Although the orange hierarchy extends the blue one, it is important to know that these two structures are orthogonal to each other. We refer to this approach as *ST-OCP Agglomerative Clustering*. We asked 20 people (including the owner of photos, the people in the photos, and third party judges) to relatively assign a number to the result of each clustering experiment between the range of 0 to 6 based on the event boundaries produced by our clustering approach. This experiment was conducted on 30 different photo streams captured in different cities inside US. Our technique did a better job compared to the other agglomerative clustering approaches in terms of providing coarser and finer precision for event and subevent boundaries. We compared the dendrograms of ST (location and time), ST-OCP (our approach), OCP, and STOCP

clustering (in which location and time and OCP attributes are used together in the distance function). The arrangement of clusters depends on the image attributes used in the clustering. The photos are sorted in chronological order. Image content features are not used in these cluster arrangements. The equation $'OCP \prec S \prec T \prec STOCP \prec ST \prec ST-OCP'$ shows that the arrangement of clusters improved from left to right — S and T, respectively, mean that agglomerative clustering is conducted using the location, and the timestamp attributes of photos. We used single linkage clustering and Euclidean distance in our clustering technique. However, one can use other approaches and refine the results

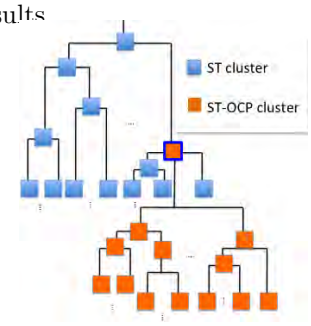


Figure 2: ST-OCP Agglomerative Clustering.

4 EOA

We present the observations with a set of descriptors. Each descriptor is a formula for a photo or a cluster — a cluster is a group of contextually similar photos. In this section, we show that it is feasible to go from a set of descriptors D to the best subevent category, when the following conditions are satisfied: (a) the descriptors in D are consistent among themselves, (b) the descriptors in D satisfy subevent categories, (c) axioms of a subevent category are consistently formulated in an event ontology, and (d) the inferred subevent categories are sound and complete.

4.1 EVENT MODEL

We use a basic derivation of E* model (Gupta, 2011) as our core event model, to specify the general relationships between events and entities. Specifically, we utilized the relationships *subeventOf*, which specifies the event structure and event containment. The expression e_1 *subeventOf* e_2 indicates that e_1 occurs within the spatiotemporal bounds of e_2, and e_1 is part of the regular structure of e_2. Additionally, we used the spatiotemporal relationships like *occurs-during* and *occurs-at* to specify the space and time properties of an event. The time and space model in this work is mostly derived from E*. We use the relationships *co-occurring-with*, and *co-located-with*, *spatially-near*, *temporal-overlap* , *before*, and *after* to describe the spatiotemporal neighborhood of an event. We used several other relationships to describe additional constraints about events (e.g., e_1 has-ambient-constraint A, and A has-value *indoor*). To express a certain group of temporal constraints, we utilized some of Linear Temporal Logic, Metric Temporal Logic, and Real-Time Temporal Logic formulas (Koymans, 1990), (Alur, 1991). We developed a language $\mathcal{L}$ with a syntax and grammar as an extension to OWL to embrace complex temporal formulas. Further, we extended the language to support a combination of classical propositional operators, linear spatial constraints, and spatial distance functions which can not be expressed in OWL; equation $f_{eucDist}(e_1, e_2, @ \leq 100)$ shows a relative spatial constraint in $\mathcal{L}$, which states the event e_1 occurs at most 100 meters away from the place at which event e_2 occurs.

Domain Event Ontology

A domain event ontology provides specialized taxonomy for a certain domain like *trip*, see fig 3. The *Miscellaneous* subevent category in this model is used to annotate the photos that are not matched with any other category. The general vocabulary in a core event model is reused in a domain event ontology. For instance, *Parking* in fig 3, is a *subClassOf* of *Occurrent* (or event) concept in the core event ontology. Also, relationships like *subeventOf* are reused from the core event ontology. We assume that domain event ontologies are handcrafted by a group of domain experts.

4.2 DESCRIPTOR REPRESENTATION MODEL

We represent a descriptor using the schema in script $\{type_d : value_d, confidence_d : val\}$, in which $type_d$, $value_d$, and val indicate the type, value,and certainty (between 0 and 1) of the descriptor, respectively. For instance, the descriptor $\{Flash : `off`, confidence : 1.0\}$ for a photo, states that the flash was off when the photo was captured with 100% certainty. Photo and cluster descriptors follow the same representation model, however the rules for computing the value of $confidence_d$ are different. We will describe these rules in the following paragraphs. The descriptor model of a cluster includes two fields in addition to that of a photo: plausibility-weight ≥ 0 , and implausibility-weight < 0. Later, we will explain the usage of these fields. All descriptors are either *direct* or *derived*. For photo descriptors, by convention, we assume that a direct descriptor is straightly extracted from the EXIF metadata of a photo, and its confidence is 1, as in the above example. The direct descriptors that we used in this paper are related to time, location, and optical parameters of photos like *GPSLatitude* ,*GPSLongitude* , *Orientation*, *Timestamp*, and *ExposureTime*. For a derived descriptor like $\{sceneType : `indoor`, confidence : 0.6\}$, the descriptor value 'indoor' is computed using direct descriptors like *Flash*, through a sequence of computations that extract information from a bucket of data sources. Some of these descriptors are *PlaceCategory*[1], and *HoursOfOperation*[2]. The confidence score is obtained from the processing unit used

[1]The nearest local business category to a photo's location.

[2]The hours during which a local business is open.

to compute the descriptor value — we developed several information retrieval algorithms for this purpose including the tools in our lab (Sinha, 2008). If a descriptor value is directly extracted from an external data source, $confidence_d$ is equal to 1. Direct descriptors of a cluster must represent all photos contained in it; some of these descriptors represent *boundingbox*, *time-interval*, and *size* of the cluster. The confidence value for direct descriptors is equal to 1, for instance, the descriptor $\{size : 5, confidence_d : 1.0\}$ indicates the number of photos in a cluster. Given a photo p_i in a photo stream P, and the cluster c that groups p_i with the most similar photos in P, a processing unit produces the descriptors of c using the descriptors of the photos in c, and more importantly, this process is guided by the descriptors of p_i. Every photo in c must support every *derived* descriptor of p_i; such cluster is referred as a *sound cluster* for p_i, and the *derived* descriptors for c are represented by the distinct union of the *derived* descriptors of the photos in c. For a derived cluster descriptor d, the value of $confidence_d$ is calculated using the formula in equation 1, in which $|c|$ is the size of the cluster, p_j is every photo in c that is represented by d, and $g(p_j, d)$ gives the confidence value of d in p_j. To find a sound cluster for a photo, the hierarchical structure that is produced by the *clustering* unit, is traversed using depth-first search — the halting condition for this navigation, if no sound cluster was found, is when current cluster is a leaf node.

$$confidence_d = \frac{1}{|c|} \times \sum g(p_j, d) \quad (1)$$

Descriptor Consistency

Consistency among a set of descriptors is a mandatory condition to infer the best possible conclusion from it. We make sure that consistency exists among the descriptors of a photo as well as the descriptors of a cluster, using entailment rules described below. *(a)* $v_i \rightarrow v_k$: if v_i implies v_k, then the rules for v_k must also be applied to v_i. This is referred as *transitive entailment rule*. For instance, suppose a photo/cluster has the following description, $'outdoorSeating : true'$; $'sceneType : outdoor'$; $'weatherCondition : storm'$, which implies that the nearest local business (e.g. restaurant) to the photo/cluster, offers *outdoorSeating*, and the weather was stormy when the photo(s) were captured. Given the sequence of rules below,

$outdoorSeating \wedge outdoor \rightarrow fineWeather; fineWeather \rightarrow \neg storm$

rule $outdoorSeating \wedge outdoor \rightarrow \neg storm$ is entailed that indicates an inconsistency among the descriptors of a photo/cluster. *(b)* $v_i \rightarrow func_{remove}(v_k)$: v_i implies removing the descriptor v_k. This is referred as a *deterministic entailment rule*. *(c)* $v_i \wedge v_k \rightarrow truth\ value$: rules of this type are referred as *non-deterministic entailment rules* in which the inconsistency is expressed by a false truth value e.g. $closeShot \wedge landscape \rightarrow false$. In that case, further decisions on keeping,modifying, or discarding either of the descriptors v_i or v_k will be based on the confidence value assigned to each descriptor — this operation is referred as *update*, which is executed when an inconsistency occurs between two candidate descriptors. The following rules are used by this process: (a) for two descriptors with the same type, the descriptor with lower confidence score is discarded, (b) for two descriptors with different types, the one with lower confidence score gets modified until the descriptors are consistent. The modification is defined as either *negation* or *expansion* within the search space. In case of negation, e.g. $\neg outdoor \rightarrow indoor$, the confidence value for *indoor* descriptor is calculated by subtracting the confidence value of *outdoor* descriptor from 1. An example of expansion is increasing a window size to discover more local businesses near a location. To avoid falling inside an infinite loop, we limit the count of negation, and the size of search space during expansion, by a threshold. We assign *null* to the descriptor that has already reached a threshold and is still inconsistent. *null* is universally consistent with any descriptor. The vocabulary that is used to model the descriptors for a photo/cluster is taken from the vocabulary that is specified in the core event model.

4.3 BUCKET OF DATA SOURCES

We represent each data source with a declarative schema, using the vocabulary of the core event model. This schema indicates the type of source output, as well as the type of input attributes a source needs to deliver the output. Data sources are queried using the SPARQL language. The following script shows an example used to query a source; var_1 is a query variable (output that must be delivered by the source); $attr_1$ is the input attribute of the source; $class_w$ indicates a class type, and rel_a indicates a relationship. The class types and relationships used in such queries are constructed using the vocabulary of the core event model.

$SELECT\ ?var_1\ FROM\ <SourceURI>\ WHERE\{$
$attr_1\ core : typeOf\ class_w;\ var_1\ core : typeOf\ class_u;$
$?var_1\ core : rel_a\ ?x;\ ?x\ core : rel_b\ ?y;\ ?y\ core : rel_d\ attr_1.\ \}$

The above query is constructed automatically using the schema of data sources, and the available information. Simply put, a source is selected if its input attributes match the available information I. At every iteration, I is incrementally updated with new data that is delivered by a source. The next source is selected if its input attributes are included in I. This process continues until no more source with matching attributes is left in the bucket B.

4.4 EVENT INFERENCE

From a set of consistent cluster descriptors (*observations*), we developed an algorithm to infer the most plausible subevent category described in a domain event ontology. This algorithm, uses the domain event model, which is a graph; we represent this graph with the notation $O(V, E)$ in which V includes event classes, and E includes event relationships. Traversing the event graph O starts with the root of hierarchical subevent structure. The algorithm visits event candidates in E through some of the relationships in E like *subeventOf*, *co-occurring-with*, *co-located-with*, *spatially-near*, *temporal-overlap*, *before*, and *after* — these relationships help to reach other event

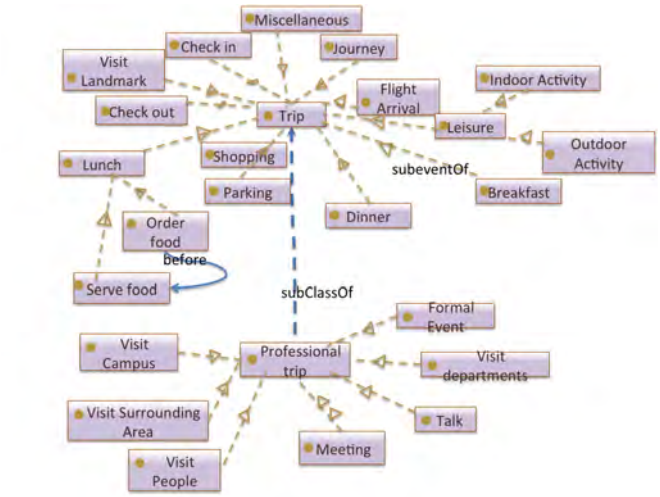


Figure 3: An event ontology for the domain *professional trip*.

candidates that are in the spatiotemporal neighborhood of an event. An expandable list, referred as L_v, is constructed from E, to maintain the visited event/subevent nodes during an iteration i — if an event is added to L_v, it cannot be processed again during the extent of i. At the end of each iteration, L_v is cleared. In every iteration, the best subevent category is inferred through a ranking process, from a set of consistent observations. We introduce *Measure of Plausibility* (m_{ij}^p) to rank event candidates, and find the most plausible subevent category. We compute m_{ij}^p using 2 parameters (a) granularity score (w_g), and (b) plausibility score (w_{AX}). w_g is equivalent to the level of the event in the subevent hierarchy in the domain event ontology. To compute w_{AX}, we used 'plausibility-weight' (w^+) and 'implausibility-weight' (w^-) which are two fields of a cluster descriptor. The value of w^+ is equal to the confidence value assigned to a descriptor, and the value of w^- is equal to $-w^+$. If a descriptor could not be mapped to any event constraint, w_{AX} remains unchanged. If a descriptor with $w^+ = \alpha$ satisfies an event constraint, then w^+ is added to w_{AX}, otherwise, w^- is added to w_{AX} (i.e., $w_{AX} = w_{AX} - \alpha$). The only exception is for the cluster descriptors *time-interval* and *boundingbox*; if either one of these descriptors satisfies an explanation, then $w^+ = 1$; in the opposite case, $w^- \leq -100$ — when a cluster has no overlap with the spatiotemporal extent of an event s_i, $w^- \leq -100$ makes s_i the least plausible candidate in the ranking. According to the formula in 4.4, w_{AX} also depends on the fraction of satisfied event constraints; N is the total number of constraints for an event candidate.

$$w_{AX} = \frac{1}{N} \sum w_{AX}^j \quad , 1 \leq j \leq N \tag{2}$$

The following instructions are used to compare two event candidates e_1 and e_2: when e_1 is subsumed by e_2, m_{ij}^p for each event candidate is normalized using the formula in equation 3, in which $e_i \equiv e_1$ and $e_j \equiv e_2$, otherwise, $e_i.m_{ij}^p = e_i.w_{AX}$. The candidate with the highest m_{ij}^p is the most plausible subevent category.

$$e_i.m_{ij}^p = \frac{e_i.w_{AX}}{\max(e_i.w_{AX}, e_j.w_{AX})} + \frac{e_i.w_g}{\max(e_i.w_g, e_j.w_g)} \tag{3}$$

When a subevent category is inferred from a set of observations, it will not be considered again as a candidate for the next set of observations. Event inference halts if no more subevent category is left to be inferred from the domain event ontology.

4.5 REFINEMENT, VALIDATION, EXTENSION

The inferred subevent categories E' are refined with the context data extracted from data sources in the bucket B, through the refinement process. First, let us elaborate this process by introducing the notion of *seed event*, which is an instance of an inferred category in E', which is not yet augmented with information. An augmented seed-event is an expressive event tag. The seed-event is continuously refined with information from multiple sources. Our algorithm uses a

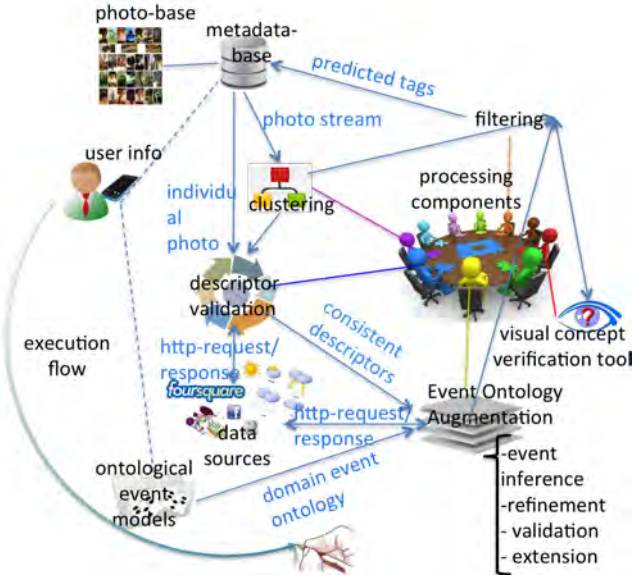


Figure 4: The Big Picture. Photos and their metadata are stored in *photo-base* and *metadata-base* respectively. Using *user info*, including events' type, time, and space in a user's calendar, a photo stream is queried, and its metadata is passed to *clustering*. In *descriptor validation*, a set of consistent descriptors is obtained from the cluster that best represents an individual photo — the component *event inference* uses these descriptors in addition to a domain event ontology that is selected according to *user info*. *EOA* derives the most relevant subevent categories to the input photo stream, and refines the derived categories by propagating their instances with the information extracted from *data sources*. The subevent tags are then validated using external sources. These tags are added to the event ontology (extension) — the extended event ontology is used in *filtering* that integrates *visual concept verification tool*. In this stage, first, irrelevant cluster branches are pruned. Next, for each matched cluster, less relevant photos to a subevent tag are filtered. The output is a set of photos labeled with some tags; these tags are then stored as new metadata for the photos. The remaining photos are tagged as *miscellaneous*.

similar strategy to what we described earlier in subsection 4.3. The only difference is that the attributes of a data source at each iteration is supplemented by the user information and the attributes of a seed-event (I) that is represented with the same schema that is described in the event ontology. Given a sequence of input attributes, if a data source returns an output-

array of size K, then our algorithm creates K new instances of events with the same type as in the seed-event, and augments them with the information in the output-array. The augmented seed-events are added to I for the next iteration; I is constantly updated until all the event categories in E' are augmented, and/or there is no more data source (in the bucket B) to query. To avoid falling into an infinite loop of querying data sources, we set the following condition: a data source cannot be queried more than once for each seed-event. We defined some queries manually that are expressed through the relative spatiotemporal relationships in the event ontology, and the augmented seed-events; these queries are used to augment the seed-events with relative spatiotemporal properties. When a seed-event gets augmented with information, our technique validates the event tag by using the event constraints, augmented event attributes, and a sequence of entailment rules that specify the *cancel* status for an event. For instance, if the weather attribute for an event is *heavy rain*, and the weather constraint *fine weather* is defined for an event, then the status of the event tag becomes *canceled*. After the validation, event tags are added to the domain event ontology by extending event classes through *typeOf* relationship. This step produces an augmented event ontology that is the extended version of the prior model (see fig 1).

5 FILTERING

Filtering is a two-step process; (1) redundant and irrelevant clusters are pruned from the hierarchical cluster structure produced by the *clustering* component, see fig 5-step-1. Equation 4 describes the *prune-rule*, and *match-rule* in this step. *traverse-rule* in equation 4 is used to visit cluster nodes— c implies cluster.

$$\neg Inside_{ST}(tag_e\ ,\ c) \rightarrow Prune(c). \quad (4)$$

$$Inside_{ST}(c\ ,\ tag_e) \rightarrow Match(c, tag_e). \quad (5)$$

$$Inside_{ST}(tag_e, c) \wedge hasChild(c) \rightarrow Trvs(c.child). \quad (6)$$

(2) filter redundant photos from the matched cluster, see fig 5-step-2. This is accomplished by applying the context and visual constraints of the expressive tag that is matched to the cluster. We used a concept verification tool[3] to verify the visual constraints of events using image features. This tool uses pyramids of color histogram and GIST features. Filtering operation is deeply guided by the expressive tags. During this operation, subevent relations are used for navigating the augmented event model.

6 EXPERIMENTAL EVALUATIONS

We focused on 3 domain scenarios vacation, professional trip, and wedding.

6.0.1 Experimental Data Set

We crawled Flickr, Picasaweb, and our lab data sets. Based on the assumption that people store their personal photos according to events,we collected the data sets based on time, space, and event types (like travel, conference, meeting, workshop, vacation, and wedding). We developed some crawlers to download about 700 albums of the day's featured photos. In addition, we crawled photo albums uploaded since the year 2010; the reason was that most of the older collections did not contain geo-tagged photos. After 4 months, we collected 84,021 albums (about 6M photos) from which only 570 albums (about 60K photos) had the required EXIF information containing location, timestamp, and optical camera parameters. We ignored the albums a) smaller than 30 photos, b) with non-English annotations. The average number of photos per album was 105. We used the albums from the most active users based on the amount of user annotation; we ended up with a diverse collection of 20 users with heterogeneous photo albums in terms of time period and geographical sparseness. The geographic sparseness of albums ranged from being across continents, to cities of the same country/state. Some of the users return to prior locations, and some do not. Fig 6 sketches the geographic distribution of our data set. We noticed that data sources do not equally support all the geographic regions; for instance,only a small number of data sources supported the data sets captured inside India. The photos for vacation/professional-trip domains have higher temporal and geographical sparseness compared to photos related to wedding domain. The number of albums for vacation domain exceeds the other two.

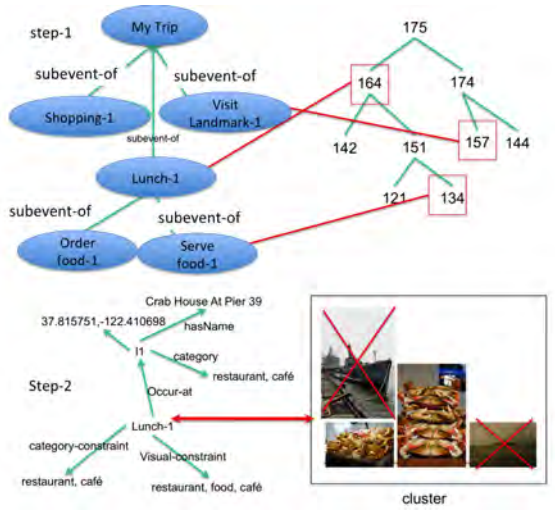


Figure 5: Filtering Operation.

6.0.2 Experimental Set-Up

We picked the 4 most active users (based on the amount of user annotation) from our non-lab, downloaded data set, and 2 most active users from our lab data set (based on the number of collections they own). As ground-truth for the lab data set, we asked the

[3] http://socrates.ics.uci.edu/Pictorria/public/demo

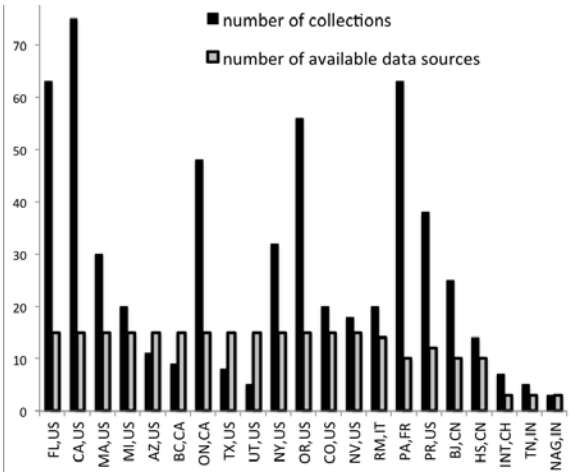


Figure 6: Data set geographical distribution. The black bars show the number of albums in each geographic region, and the gray bars show the number of data sources that supported the corresponding geographic region.

owners to annotate the photos using their personal experiences, and an event model that best describes the data set, while providing them with three domain event ontologies (wedding, professional trip, and vacation). For the non-lab data set, the ground truth provides a manual and subjective event labeling done by the very owner of the data set being unaware of the experiments. Because of the subjective nature of the non-lab data set, the event types that were not contained in the event domain ontology are replaced with event type *miscellaneous* that is an event type in every domain event ontology in this work. For each experiment, we compute standard information retrieval measures (precision, recall, and F1-measure), for the event types used in tags. In addition to that, we introduce a measure of correctness for event tags. The score is obtained based on multiple context cues. For instance, label *meeting with Tom Johnson at RA Sushi Japanese Restaurant in Broadway, San Diego, during time interval "blah" in a sunny day, in an outdoor environment*, specifies type of the event, its granularity in the subevent hierarchy, place, time, and environment condition. We developed an algorithm that evaluates each cue with a number in the range of 0 to 1 as follows: *1)* event type: wrong = 0, correct = 1, somehow correct = $\frac{L_p}{L_{TP}}$ such that L_p is the subevent-granularity level for a predicted tag and L_{TP} is the subevent granularity level for the true-positive tag (the predicted tag is the direct or indirect superevent of the true-positive tag i.e., $\frac{L_p}{L_{TP}} \leq 1$); *2)* place: includes place name, category and geographical region. If the place name is correct, score 1 is assigned and the other attributes will not be checked. Otherwise, 0 is assigned; for the category and/or geographical region if correct, score 1 is assigned, and 0 otherwise. The average of these values represent the score for place; *3)* for weather, optical, and visual constraint: wrong=0, correct =1, unsure = 0.5; *4)* time interval: if the predicted event tag occurs anytime during the true-positive event tag, 1 is the score, otherwise 0. The average of the above scores represents the correctness measure for a predicted event tag. We introduce *average correctness* of annotation that is calculated using the formula in equation 7, where w_j is the score for the j^{th} predicted tag.

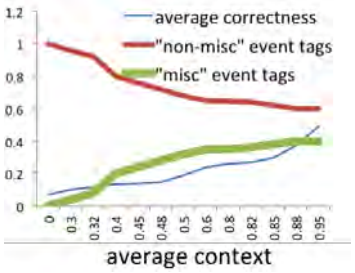


Figure 7: Role of context in improving the correctness of event tags.

$$\overline{correctness} = \frac{\sum_{j=1}^{L} w_j}{L} \; ; \; \overline{context} = 1 - \overline{Err} \qquad (7)$$

The metric $\overline{context}$ in equation 7 is used to measure the average context provided by data sources for annotating a photo stream; parameter $\overline{Err}$ is the average error related to the information provided by data sources used for annotating a photo stream ($0 \leq \overline{Err} \leq 1$); the following guidelines are applied automatically, to measure this value: *(a)* if the information in a data source is related to the domain of a photo stream, but it is irrelevant to the context of the photo stream, assign error-score 1. For instance, data source *TripAdvisor* returns zero results related to *Things-To-Do* for the country at which a photo stream is created. Also, if a photo stream for a vacation trip does not include any picture taken in any landmark location, *TripAdvisor* does not provide any coverage; *(b)* assign error-score 0 if the type of a source is relevant as well as its data (i.e. non-empty results); *(c)* if the data from a relevant source is insufficient for a photo stream, assign error-score 0.5. For instance, only a subset of business venues in a region are listed in data source *Yelp*; as a result, the data source returns information for less than 30% of the photo stream; *(d)* for a data source, multiply the error-score by a fraction in which the numerator is the number of photos tagged using this data source, and the denominator is the size of the photo stream. Do this for all the sources and obtain the weighted average of the error-scores. The result is $\overline{Err}$. The implication of our result in fig 7 is as follows: while the correctness of event tags (for a photo stream of an event) peaks with the increase in $\overline{context}$, relatively, smaller percentage of photos are tagged using *non-miscellaneous* events, and larger percentage of photos are tagged using *miscellaneous* event. This means if the suitable event type for a group of photos does not exist in an event ontology, the photos are not tagged with an irrelevant *non-miscellaneous* event; instead, they are tagged with *miscellaneous* event which means *other*. The right side of the figure indicates that even though the number of miscellaneous and

non-miscellaneous event tags does not change, the correctness is still increasing; this means that the tags get more expressive since more context cues are attached to them. The quality of annotations is increased when more context information is available. This shows that event ontology by itself is not as effective as augmented event ontology. We demonstrate three classes of experiments in table 1. This table shows the average values (between 0 to 1) for the measure metrics discussed earlier (precision, recall, F1, $\overline{correctness}$). We use the work proposed in (Paniagua, 2012) as a baseline. It is based on space and time to detect event boundaries in conjunction with using English album descriptions. This baseline approach, with F1-measure about 0.6 and correctness of almost 0.56, illustrates that time and space are important parameters to detect event boundaries. On the other hand, the baseline approach is limited to using only spatiotemporal containment for detecting subevent hierarchy, it does not support other types of relationships among events (like co-occurring events, relative temporal relationships) and other semantic knowledge about the structure of events. Also, it requires human-induced tags which are noisy. For the second set of experiments, we use an event domain ontology without augmenting it with context information. This approach gives worse results since the context information is disregarded during detecting event boundaries. It provides the F1-measure of almost 0.32 and correctness of 0.13. Our last experiment leverages our proposed approach, and achieves F1-measure of about 0.85, and correctness of 0.82. Compared to our baseline approach, we obtain about 26% improvement in the quality of tags which is a very promising result.

6.0.3 CPU-Performance

The running time for EOA, and visual concept verification is shown in fig 8, which illustrates the results for data sets of two sources i.e., lab, and non-lab (including Flickr, and Picasaweb), and three event domains.

Stage 1: Intra-Domain Comparison

In general, we found smaller number of context sources for wedding data sets compared to the other two domains; as a result, the EOA process exits relatively faster, and the running time for the concept verification process increases. We observed the correctness of event tags degrades when EOA process exists fast. This observation confirms the findings of fig 7.

Stage 2: Intra-Source Comparison

Within each domain, we compared the cpu-performance among lab and non-lab data sets; EOA exits relatively faster for non-lab data sets. The justification for this observation is that we could obtain user-related context like facebook events/check-ins from our lab users (U3, U4), but such information was missing in the case of non-lab data sets. This absence of

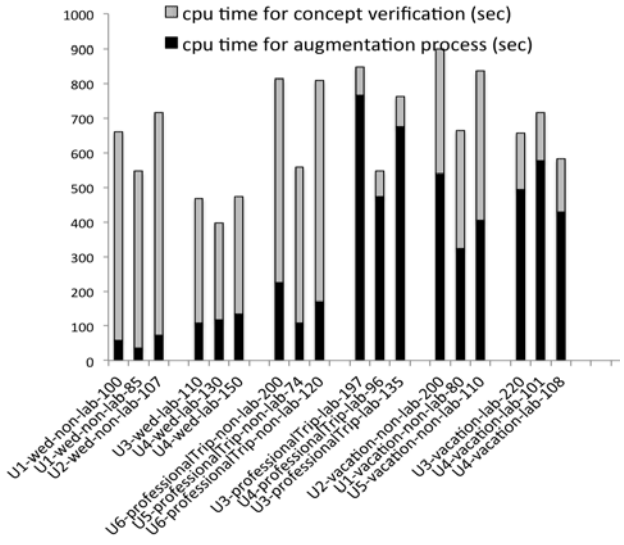


Figure 8: CPU-Time for experimental data sets of the 6 most active users. Each data set is represented by its owner, domain type, source, and size. The domain *wed* implies *wedding* domain.

Table 1: Results for automatic photo annotation for the data sets owned by the 6 most active users.

Users		U1	U2	U3	U4	U5	U6
baseline	prec	0.65	0.58	0.39	0.53	0.74	0.61
	recall	0.89	0.4	0.61	0.64	0.8	0.43
	f1	0.75	0.47	0.48	0.6	0.77	0.5
	corr	0.63	0.62	0.52	0.62	0.28	0.69
event ontology	prec	0.41	0.17	0.3	0.48	0.12	0.53
	recall	0.4	0.2	0.5	0.43	0.24	0.3
	f1	0.4	0.18	0.37	0.45	0.16	0.38
	corr	0.2	0.08	0.12	0.2	0.03	0.19
proposed	prec	0.74	0.83	0.95	0.92	0.88	0.79
	recall	0.91	0.93	0.88	0.7	0.97	0.82
	f1	0.81	0.88	0.91	0.79	0.92	0.8
	corr	0.8	0.75	0.85	0.79	0.9	0.88

information impacts wedding data sets the most, since the context information in the *wedding* scenario largely includes personal information such as guest list, and wedding schedule that are not publicly available on photo sharing websites. In *professionalTrip* scenario, this impact is smaller than *wedding*, and larger than *vacation*; the missing data is due to the lack of context information related to personal meetings, and conference schedules. In *vacation* scenario, data sources are mostly public; only a small portion of context information comes from the user-related context such as flight information,and facebook check-ins; therefore, we did not find a significant change in the cpu-time between lab and non-lab data sets.

7 CONCLUSIONS

Our proposed technique addresses a broad range of research challenges to achieve a powerful event-based system that can adapt to different scenarios and applications like those in intelligence community, multimedia applications, and emergency response. This is the starting step for combining complex models with big data.

References

M. Naaman, S. Harada, Q.Y. Wang, H. Garcia-Molina, and A. Paepcke (2004). Context data in geo-referenced digital photo collections. *In Proceedings of the 12th annual ACM international conference on Multimedia.*

M. Naaman, Y.J. Song, A. Paepcke, and H. Garcia-Molina (2004). Automatic organization for digital photographs with geographic coordinates. Digital Libraries. In Proceedings of the 2004 Joint ACM/IEEE Conference.

R. Datta, D. Joshi,J. Li, and J.Z. Wang (2008). Image retrieval: Ideas, influences, and trends of the new age, ACM Computing Surveys (CSUR).

R. Jain, and P. Sinha (2010). Content without context is meaningless, Proceedings of the international conference on Multimedia, ACM.

W. Viana, J.Filho, J.Gensel, M. Villanova Oliver, and H.Martin (2007). PhotoMap–Automatic Spatiotemporal Annotation for Mobile Photos, Web and Wireless Geographical Information Systems, Springer.

W. Viana, J. Bringel Filho,J. Gensel,M. Villanova-Oliver,and H. Martin (2008). PhotoMap: from location and time to context-aware photo annotations, Journal of Location Based Services, Taylor & Francis.

X. Liu, R. Troncy, and B. Huet (2011). Finding media illustrating events, Proceedings of the 1st ACM International Conference on Multimedia Retrieval.

A. Fialho, R. Troncy, L. Hardman, C. Saathoff, and A. Scherp (2010). What's on this evening? Designing User Support for Event-based Annotation and Exploration of Media, 1st International Workshop on EVENTS-Recognising and tracking events on the Web and in real life.

L. Cao, J. Luo, H. Kautz, and T.S. Huang (2008). Annotating collections of photos using hierarchical event and scene models, Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference.

J. Paniagua, I. Tankoyeu, J. Stöttinger, and F. Giunchiglia (2012). Indexing media by personal events, Proceedings of the 2nd ACM International Conference on Multimedia Retrieval.

N.R. Brown (2005). On the prevalence of event clusters in autobiographical memory, Social Cognition, Guilford Press.

A. Pigeau, and M. Gelgon (2005). Building and tracking hierarchical geographical & temporal partitions for image collection management on mobile devices, Proceedings of the 13th annual ACM international conference on Multimedia.

B. Gong, U. Westermann, S. Agaram, and R. Jain (2006). Event Discovery in Multimedia Reconnaissance Data Using Spatio-Temporal Clustering, Proceeding of the AAAI Workshop on Event Extraction and Synthesis.

B. Gong, and R. Jain (2008). Hierarchical photo stream segmentation using context, Proceedings of SPIE, Multimedia content Access: Algorithms and System.

P. Sinha, and R. Jain (2008). Classification and annotation of digital photos using optical context data, CIVR.

A. Gupta, and R. Jain (2011). Managing Event Information: Modeling, Retrieval, and Applications, Synthesis Lectures on Data Management, Morgan & Claypool Publishers.

R. Koymans (1990). Specifying real-time properties with metric temporal logic, Real-Time Syst.,2(4).

R. Alur and T.A. Henzinger (1991). Logics and models of real time: A survey, J. W de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, REX Workshop, Springer.

S. Rafatirad, A. Gupta, and R. Jain (2009). Event composition operators: ECO, Proceedings of the 1st ACM international workshop on Events in multimedia.

S. Rafatirad, and R. Jain (2011). Contextual Augmentation of Ontology for Recognizing Sub-events, Semantic Computing (ICSC), 2011 Fifth IEEE International Conference.

Learning Parameters by Prediction Markets and Kelly Rule for Graphical Models

Wei Sun
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Robin Hanson
Department of Economics
George Mason University
Fairfax, VA 22030

Kathryn B. Laskey
Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030

Charles Twardy
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Abstract

We consider the case where a large number of human and machine agents collaborate to estimate a joint distribution on events. Some of these agents may be statistical learners processing large volumes of data, but typically any one agent will have access to only some of the data sources. Prediction markets have proven to be an accurate and robust mechanism for aggregating such estimates (Chen and Pennock, 2010), (Barbu and Lay, 2011). Agents in a prediction market trade on futures in events of interest. Their trades collectively determine a probability distribution. Crucially, limited trading resources force agents to prioritize adjustments to the market distribution. Optimally allocating these resources is a challenging problem. In the economic spirit of specialization, we expect prediction markets to do even better if agents can focus on beliefs, and hand off those beliefs to an optimal trading algorithm. Kelly (1956) solved the optimal investment problem for single-asset markets. In previous work, we developed efficient methods to update both the joint probability distribution and user's assets for the graphical model based prediction market (Sun et al., 2012). In this paper we create a Kelly rule automated trader for combinatorial prediction markets and evaluate its performance by numerical simulation.

1 INTRODUCTION

There was a time when "big" data meant too big to fit into memory. As memory capacity expanded, what was once big became small, and "big" grew ever bigger. Then came cloud computing and the explosion of "big" data to a planetary scale. Whatever the scale of "big," a learner faced with big data is by definition forced to subsample, use incremental methods, or otherwise specialize. Humans are no strangers to specialization: scientific knowledge alone has exceeded the capacity of any single head for at least four centuries. In this paper we consider a mechanism for fusing the efforts of many specialized agents attempting to learn a joint probability space which is assumed to be larger than any one of them can encompass. We seek a mechanism where agents contribute only where they have expertise, and where each question gets the input of multiple agents. As we discuss below, our mechanism is a kind of prediction market. However, in contrast to (Barbu and Lay, 2011), we wish our human and machine agents to concentrate on their beliefs, not on playing the market. Therefore we formulate and develop a helper agent which translates partial beliefs into near-optimal trades in a combinatorial market of arbitrary size. We do not here actually apply it to a big dataset.

It is well known that prediction accuracy increases as more human and/or machine forecasters contribute to a forecast (Solomonoff, 1978). While one approach is to ask for and average forecasts from many individuals, an approach that often works better is to combine forecasts through a prediction market (Chen and Pennock, 2010; Barbu and Lay, 2011). In a market-maker based prediction market, a consensus probability distribution is formed as individuals either edit probabilities directly or trade in securities that pay off contingent on an event of interest. Combinatorial prediction markets allow trading on any event that can be specified as a combination of a base set of events. However, explicitly representing the full joint distribution is infeasible for markets with more than a few base events. Tractable computation can be achieved by using a factored representation (Sun et al., 2012). Essentially, the probabilities being edited by users are the parameters of the underlying graphical model representing the joint distribution of events of interest. In

another words, prediction markets work as a crowdsourcing tool for learning model parameters from human or automated agents.

Prediction markets appear to achieve improved accuracy in part because individuals can focus on contributing to questions about which they have the most knowledge, and in part because individuals can learn by watching the trades of others. However, one important disadvantage of prediction markets is that users must figure out how to translate their beliefs into trades. That is, users must decide when to trade how much, and whether to make a new offer or to accept an existing offer. Prediction markets with market makers can simplify this task, allowing users to focus on accepting existing offers. Edit-based interfaces can further simplify the user task, by having users browse for existing estimates they think are mistaken, and then specify new values they think are less mistaken.

However, even with these simplifications, participants must think about resources as well as beliefs. For example, if users just make edits whenever they notice that market estimates differ from their beliefs, participants are likely to quickly run out of available resources, which will greatly limit their ability to make further edits. To avoid this problem, users must try to keep track of their best opportunities for trading gains, and avoid or undo trades in other areas in order to free up resources to support their best trades.

This problem is made worse in combinatorial prediction markets. By allowing users to trade on any beliefs in a large space of combinations of some base set of topics, combinatorial markets allow users to contribute much more information, and to better divide their labors. But the more possible trades there are, the harder it becomes for users to know which of the many possible trades to actually make.

Ideally, prediction market users could have a trading tool available to help them manage this process of translating beliefs into trades. Users would tell this tool about their beliefs, and the tool would decide when to trade how much. But how feasible is such a tool? One difficulty is that optimal trades depend in principle on expectations about future trading opportunities. A second difficulty is that users must not only tell the tool about their current beliefs, they must also tell the tool how to change such stated beliefs in response to changes in market prices. That is, the trading tool must learn from prices in some manner analogous to the way the user would have learned from such prices.

To make this problem manageable, we introduce four simplifications. First, we set aside the problem of how users can easily and efficiently specify their current beliefs, and assume that a user has somehow specified a full joint probability distribution over some set of variables. Second, we set aside the problem of guessing future trading opportunities, by assuming that future opportunities will be independent of current opportunities. Third, we assume that a user can only accept trade offers made by a continuous market maker, and cannot trade directly with other users. Fourth, we set aside the problem of how a tool can learn from market prices, by assuming that it would be sufficient for the tool to optimize a simple utility function depending on the user's assets and market prices.

Given these assumptions, the prediction market trading tool design problem reduces to deciding what prediction market trades to make any given moment, given some user-specified joint probability distribution over a set of variables for which there is a continuous combinatorial market maker. It turns out that this problem has largely been solved in the field of evolutionary finance.

That is, if the question is how, given a set of beliefs, to invest among a set of available assets to minimize one's chances of going broke, and maximize one's chance of eventually dominating other investors, the answer has long been known. The answer is the "Kelly rule" (Kelly, 1956), which invests in each category of assets in proportion to its expected distant future fraction of wealth, independent of the current price of that category. When they compete with other trading rules, it has been shown that Kelly rule traders eventually come to dominate (Lensberg and Schenk-Hoppé, 2007).

In this paper we report on an implementation of such a Kelly rule based trading tool in the context of a combinatorial prediction market with a continuous market maker. Section 2 reviews the basics of such a combinatorial prediction market. Section 3 reviews the basics of a Kelly rule and then describes how to apply the Kelly rule in a combinatorial prediction market to achieve automated trading. Section 4 reports on simulation tests of an implementation of the Kelly autotrader. Last, in Section 6 we summarize our work and point to potentially promising future research opportunities.

2 COMBINATORIAL PREDICTION MARKETS

In a prediction market, forecasters collaboratively form a probability distribution by trading on assets that pay off contingent on the occurrence of relevant events. In a logarithmic market scoring rule based (LMSR-based) prediction market, a market maker of-

fers to buy and sell assets on any relevant events, varying its price exponentially with the quantity of assets it sells. Tiny trades are fair bets at the consensus probabilities (Hanson, 2003). Larger trades change the consensus probabilities; we call such trades "edits." Suppose $\{z_i, i = 1, \cdots, n\}$ is a set of n possible outcomes for a relevant event, and prior to making a trade, the user's assets contingent on occurrence of z_i are a_i. Suppose the user makes an edit that changes the current consensus probability from p_i to x_i. In a LMSR-based market, as a result of the trade, the user's assets contingent on occurrence of z_i will now be $a_i + b \log_2(\frac{x_i}{p_i})$.

A combinatorial prediction market increases the expressivity of a traditional prediction market by allowing trades on Boolean combinations of a base set of events (e.g., "A and B") or contingent events defined on the base events (e.g., "A given B"). While it is in general NP-hard to maintain correct LMSR prices across an exponentially large outcome space (Chen et al., 2008), limiting the consensus distribution to a factored representation of the joint distribution provides a tractable way to achieve the expressiveness of a combinatorial market. Sun, et al. (2012) showed how adapt the junction tree algorithm to jointly manage each user's assets along with a market consensus probability distribution for a combinatorial prediction market. Our probability and asset management approach has been implemented in a combinatorial prediction market for forecasting geopolitical events (Berea et al., 2013).

3 KELLY RULE AUTO-TRADER

Unlike financial or commodities markets, where financial gain or loss is the primary purpose, the aim of a prediction market is to form consensus forecasts from a group of users for events of interest. A successful prediction market depends on participants who are knowledgeable about and interested in the events in the market. However, lack of experience with or interest in the management of assets can be a significant barrier to participation for some forecasters. Finding a way to engage such content-knowledgeable but market-challenged forecasters could significantly improve the performance of a prediction market. Thus, there is a need for a tool to translate beliefs of forecasters into market trades that efficiently allocate assets according to the forecaster's beliefs.

Fortunately, just such a tool is available from the finance literature. A firmly established result is that we should expect financial markets in the long run to be dominated by investment funds which follow a "Kelly Rule" of investing. Such a strategy invests in each category of assets in proportion to its expected future financial value (Evstigneev et al., 2006; Lensberg and Schenk-Hoppé, 2007; Amir et al., 2001). That is, a Kelly Rule fund that expects real estate to be 20% of distant future wealth should invest 20% of its holdings in real estate. In our context this is equivalent to maximizing an expected log asset holdings, as shown below.

3.1 OPTIMAL ASSET ALLOCATION

The Kelly rule (Kelly, 1956) determines the best proportion of a user's assets to invest in order to achieve the maximum asset growth rate. We briefly review how the Kelly rule works in a simple binary lottery, where a loss means losing one's investment and a win means gaining the amount bet times the payoff odds. Suppose an investor starts with assets y_0; the return is z for a unit bet; and each investment is a fixed percentage c of the user's current assets. After each lottery, the user's assets are multiplied by $(1 + zc)$ in case of a win and $(1 - c)$ in case of a loss. This yields an expected exponential growth rate of

$$\begin{aligned} g(c) &= E\left[\log(\frac{y_n}{y_0})^{\frac{1}{n}}\right] \\ &= E\left[\frac{w}{n}\log(1+zc) + \frac{l}{n}\log(1-c)\right] \end{aligned} \tag{1}$$

where n is the number of trades the user has made; w is the number of times the user has won; and l is the number of times the user has lost. As n approaches infinity, Eq. 1 becomes

$$\begin{aligned} &\lim_{n\to+\infty} E\left[\log(\frac{y_n}{y_0})^{\frac{1}{n}}\right] \\ &\quad = p\log(1+zc) + (1-p)\log(1-c) \end{aligned} \tag{2}$$

Now, suppose there are n possible outcomes $\{z_i, i = 1, \cdots, n\}$; the user's probability for outcome z_i is h_i; and the user's current assets if z_i occurs are a_i. The current market distribution is $\{p_i, i = 1, \cdots, n\}$, and the user is contemplating a set of edits that will change the distribution to $x_i, i = 1, \cdots, n$. Given these edits, the user's assets if outcome z_i occurs will be $\log(a_i + b\log_2(\frac{x_i}{p_i}))$. The maximum asset growth rate is obtained by solving the following optimization problem:

$$\max_x \sum_{i=1}^{N}\left[h_i \times \log(a_i + b\log_2(\frac{x_i}{p_i}))\right] \tag{3}$$

subject to

$$\sum_{i=1}^{N}(x_i) = 1$$

$$0 < x_i < 1, \forall i$$

and

$$a_i + b \log_2(\frac{x_i}{p_i}) >= 0.$$

where i is the global joint state.

3.2 APPROXIMATELY OPTIMAL ALLOCATION

It is straightforward to define the optimization shown in Equation (3) for a joint probability space. However, as noted above, representing the full joint space is in general intractable. We therefore consider the problem in which the user's edits are further constrained to *structure-preserving* edits, which respect the underlying factored representation, ensuring computational tractability (if probabilistic inference itself is tractable).

The objective function in Equation (3) is the expected updated asset w.r.t. the user's beliefs $\{h_i\}$ over the entire joint space. It is desirable to decompose the optimization according to the cliques in the junction tree of the graphical model. However, this is non-trivial because of the logarithm. If edits are structure preserving, assets decompose additively (Sun et al., 2012) as:

$$a_i = \sum_{c \in \mathcal{C}} a_c - \sum_{s \in \mathcal{S}} a_s, \tag{4}$$

where $\mathcal{C}$ is the set of cliques and $\mathcal{S}$ is the set of separators in the junction tree. Therefore (Cowell et al., 1999), computation of expected assets can be decomposed via a local propagation algorithm. However, the logarithmic transformation $\log(a_i + b \log_2(\frac{x_i}{p_i}))$ is not additively separable, and no local propagation algorithm exists for computing the expected utility.

We therefore seek a local approximate propagation algorithm. It is often reasonable to assume that a user has beliefs on only a few of the variables in the market. If all the edited variables are confined to a single clique c_j, we know

$$\frac{x_i}{p_i} = \frac{x^i_{c_j}}{p^i_{c_j}}.$$

One approximation approach is to initialize the user's asset tables $\{a_c, c \in \mathcal{C}\}$ and $\{a_s, s \in \mathcal{S}\}$ with all zeros, and keep a separate cash account containing the user's assets prior to any trades. Each single trade is confined to a single clique. At the time of the trade, we move the cash amount into the asset table into the clique the user is editing. We then solve the optimization problem 3 for the single clique of interest. The user invests the optimal amount. We then find the minimum post-trade assets for the clique of interest. Because of the logarithmic utility, this amount will be greater than zero. We then subtract this positive amount from the clique asset table and add it back to the cash account. We then move to another clique and make the optimal edit there in the same way.

We call this process *local cash-asset management*. This process iteratively moves all cash into a clique, finds the optimal edit confined to that clique, and then moves the maximal amount back to cash while ensuring that all entries in the clique asset table are non-negative. This process proceeds through all the cliques, and increases the expected utility at each step. Local cash-asset management always leaves all separators with zero assets, thus removing the need to manage separators. Furthermore, the global asset computation is simplified to be the sum of all clique assets.

The separately maintained cash amount is guaranteed to be less than or equal to the global minimum assets as computed by the algorithm of (Sun et al., 2012). Thus, this approach is a conservative strategy, improving the user's expected utility but is not guaranteed to reach the maximum expected utility.

4 NUMERICAL SIMULATION

We implemented the Kelly Auto-Trader in MATLAB with an open-source nonlinear optimization solver called IPOPT (Wächter and Biegler, 2006).

4.1 EXPERIMENT DESIGN

To test market performance with the Kelly auto-trader, we simulated a market over a 37-node network whose structure matches *ALARM* (Beinlich et al., 1989). The ALARM network (see Figure 1) is often used as a benchmark for graphical model algorithms, and is substantial enough to provide a reasonable test of our approach. The approach itself is limited only by the efficiency of the nonlinear solver.

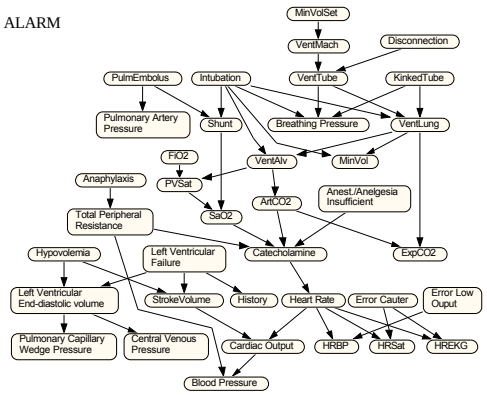


Figure 1: *ALARM*

ALARM has 27 cliques in its junction tree, among which the biggest clique has 5 variables and table size of 108.

To ensure consistency of beliefs across cliques, we use the following procedure to generate a simulated user's beliefs:

1. Choose n cliques to have beliefs, usually about 1/3 of the total number of cliques;
2. Proceeding sequentially for each of the chosen cliques, generate its random belief by a random walk, simulating an efficient market. Update beliefs on the clique and propagate to other cliques using the junction tree algorithm. This procedure gives n junction tree propagations in total.
3. After all propagations, take the potentials on the chosen cliques as the user's final beliefs.

We simulate market participation using three types of traders.

Type 1: EVmaxer invests all of her cash in the market to achieve the maximum asset gain. Formally, EVmaxer solves the following optimization to find the best edit x with her beliefs h,

$$\max_x \sum_{i=1}^{N} \left[h_i \times b \log_2(\frac{x_i}{p_i}) \right] \tag{5}$$

subject to

$$\sum_{i=1}^{N}(x_i) = 1$$

$$0 < x_i < 1, \forall i$$

and

$$a_i + b \log_2(\frac{x_i}{p_i}) >= 0.$$

where i is the global joint state. EVmaxer can suffer catastrophic losses.

Type 2: Kelly-trader is risk-averse and theoretically has the best growth rate in long run. Kelly-trader uses Equation 3 to find the best edit at each trade opportunity.

Type 3: Noiser trades randomly. In any market, we expect a number of noisy traders who have less knowledge than other traders. We simulated Noiser's behavior by moving the current market distribution by random walk of white noise with 15% deviation.

At each trade step, we determine the trader type by taking a random draw from a distribution on the proportion of trader types. Assuming both EVmaxer and Kelly-trader know the pre-generated beliefs, we allow them to determine their optimal edits according to their respective objective functions. They make their edits and the market distribution is updated. Edits continue in this way until interrupted by a question is resolved – that is, its value becomes known to all participants, and all trades depending on the resolved question are paid off by the market maker. The inter-arrival time for question resolutions is modeled by an exponential distribution with mean μ_t. When resolving, we track the min-asset and max-asset for EVmaxer and Kelly-traders to measure their performance. Further, after resolving a question, we add a new question back to the model at the same place where the resolved question was located in the network. Finally, after a certain number of trades, we resolve all questions one by one based on their probabilities. The final assets for different types of users are then calculated and compared.

There are some free parameters in the simulation setting, such as the user's initial assets, the market scaling parameter b, the proportions of different types of traders, etc. The following are the parameter values for a typical simulation run, with explanation of how to choose appropriate values:

- Initial assets $S = 20$ – the small starting assets of 20 will show how EVmaxer goes broke because of her aggressive trading, and how the Kelly-trader is able to grow her assets from a small starting point;
- Market scaling parameter $b = 1000$ – the bigger b, the less influence each trader has; a large b mimics a thick market with many traders;
- Number of trades 1000 – to model the concept of long run effect;
- Mean time between resolutions 30 trades – actual resolutions are drawn from an exponential inter-

arrival distribution with this mean; 30 trades provides sufficient opportunities for well-informed traders to move the market to the correct direction before resolving questions.

- Market participants are composed of 20%, 20%, 60% of EVmaxer, Kelly-trader, and Noiser respectively. Basically, we expect better accuracy of the market probability estimation when there are more Kelly traders, and/or more frequent editing by Kelly traders.

4.2 RESULTS AND ANALYSIS

For a typical run of 1000 trades, Figure 2 presents the marginal probabilities of the resolving states (totally 35 resolutions in this run). At each resolution point, a question was randomly chosen from the market and resolved based on its marginal distribution at the moment. Those resolving probabilities are the "ground truth" values generated by the random walk. Figures 3 and 4 show, in the same simulation run, the performance for EVmaxer and Kelly-trader in terms of their min-asset and max-asset at each resolution point.

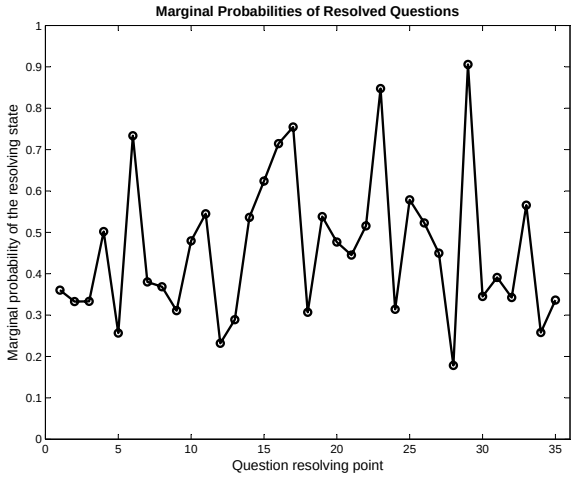


Figure 2: Marginal Probabilities of Resolving State

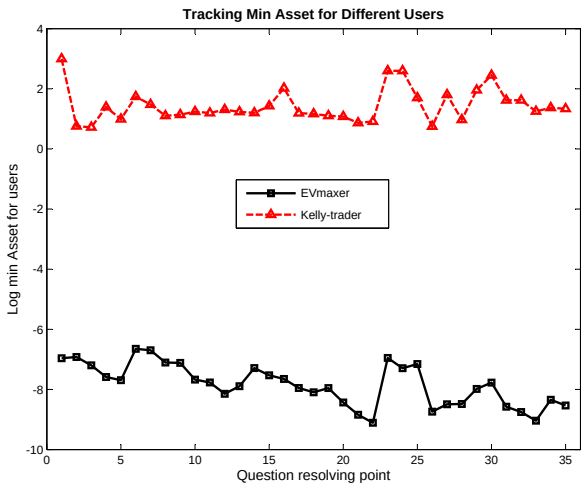


Figure 3: Min-asset Comparison between EVmaxer and Kelly-trader

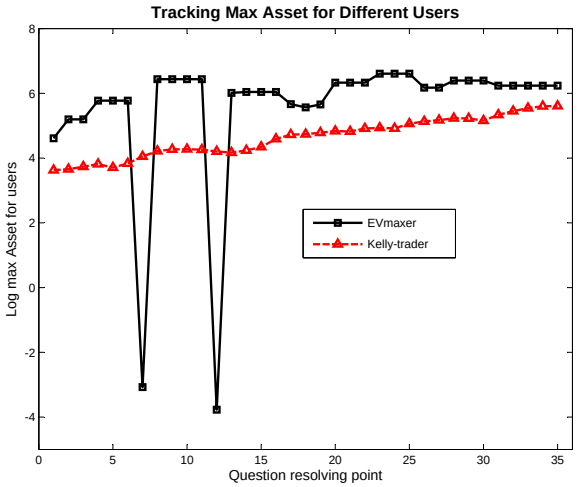


Figure 4: Max-asset Comparison between EVmaxer and Kelly-trader

As expected, Figure 3 shows that the Kelly trader always reserves some assets, while EVmaxer makes very aggressive bets. Notice in Figure 4 that EVmaxer went broke twice at the 7^{th} and 12^{th} question resolution points (we re-initialize EVmaxers' assets at bankruptcy to let them continue to trade). Basically, EVmaxer has a lot bigger volatility while Kelly-trader grows assets consistently. In the simulation, we re-initialize the EVmaxer with the starting asset. But in practice, just one strike will make the EVmaxer deeply hurt. At the end of this run, we sampled the market distribution for 1000 times. Using each sample as the final resolving states for all questions, we compared the final payoff asset for both EVmaxer and

Kelly-trader. Histograms of the final assets are shown in Figure 5, and 6. As you may notice, Kelly-trader has very consistent distribution with mean of 58, and standard deviateion of about 31. But EVmaxer's final asset is distributed very sparsely. Most of time (almost dominant), EVmaxer will have final asset close to zero, although its possible maximum value can be more than 1000.

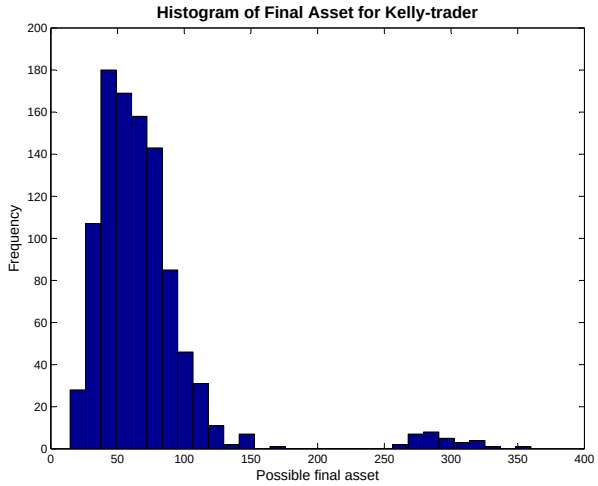


Figure 5: Histogram of the final assets for Kelly-trader

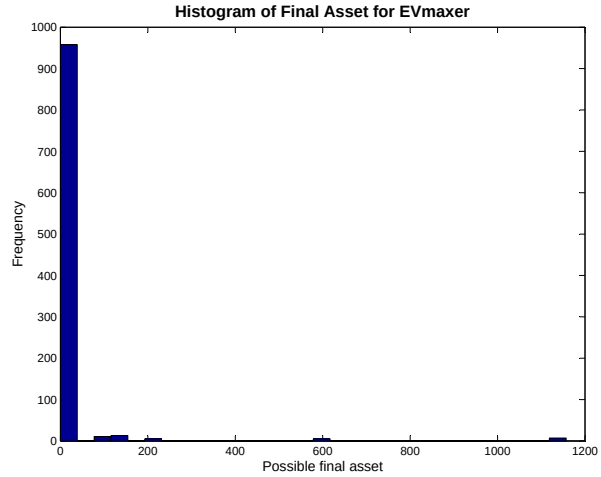


Figure 6: Histogram of the final assets for EVmaxer

5 TAYLOR APPROXIMATION

In this section, we present an alternate approach in which we approximate the utility function by a second-order Taylor series, yielding an approximation to the expected utility in terms of first and second moments of the utility random variable. We then apply methods for local computation of moments of real-valued functions defined on graphical models (Nilsson, 2001; Cowell et al., 1999) to obtain an approximation to the expected utility.

5.1 NOTATION AND DEFINITIONS

Let $\{Z_v : v \in V\}$ be a set of finitely many discrete random variables; let Ω_v denote the set of possible values of Z_v, and let z_v denote a typical element of Ω_v. For $C \subset V$, we write Z_C for $\{Z_v : v \in C\}$, Ω_C for the Cartesian product $\times_{c \in C} \Omega_c$ and z_c for a typical element of Ω_C.

A junction tree $\mathcal{T}$ on V is an undirected graph in which the nodes are labeled with subsets $C \subset V$ called *cliques*; each arc is labeled with the intersection $S = C \cap D$, called the *separator*, of the cliques at the ends of the arc; every $v \in V$ is in at least one clique; there is exactly one path between any two cliques, i.e., $\mathcal{T}$ is a tree; and $C \cap D$ is contained in very clique along the path from C to D. We let $\mathcal{C}$ denote the set of cliques and $\mathcal{S}$ denote the set of separators.

We say a real-valued function f on Ω_V factorizes on the junction tree $\mathcal{T}$ if there exist non-negative real-valued functions $\{h_C : C \in \mathcal{C}\}$ on Ω_C and and $\{h_D : D \in \mathcal{D}\}$ on Ω_D such that for all $v \in V$:

$$f(z_V) = \frac{\prod_{C \in \mathcal{C}} h_C(z_C)}{\prod_{D \in \mathcal{D}} h_D(z_D)} \tag{6}$$

where z_C and z_D denote the components of z_V corresponding to C and D, respectively, and $h_D(z_D) = 0$ only if there is at least one clique D with $D \cap C \neq \emptyset$ and $h_C(z_C) = 0$. In case $h_D(z_D) = 0$ and $h_C(z_C) = 0$ for $D \cap C \neq \emptyset$, we take $h_C(z_C)/h_D(z_D)$ to be 0.

We say a function f on Ω_V decomposes additively on the junction tree $\mathcal{T}$ if there exist non-negative real-valued functions $\{h_C : C \in \mathcal{C}\}$ on Ω_C and and $\{h_D : D \in \mathcal{D}\}$ on Ω_D such that for all $v \in V$:

$$f(z_V) = \sum_{C \in \mathcal{C}} h_C(z_C) - \sum_{D \in \mathcal{D}} h_D(z_D) \tag{7}$$

where z_C and z_D denote the components of z_V corresponding to C and D, respectively. The functions $h_C(x_c)$ and $h_D(x_D)$ in (6) and (7) are called *potentials*; the set $\{h_B : B \in \mathcal{C} \cup \mathcal{D}\}$ is called a (multiplicative or additive, respectively) *potential representation* of f.

5.2 ASSETS AND PROBABILITIES

We assume the user's probability distribution g factorizes according to the junction tree $\mathcal{T}$. We assume trades are constrained to be structure preserving with respect to $\mathcal{T}$; hence, the before-trade market distribution p and the market distribution x after a structure preserving trade also factorize according to $\mathcal{T}$. Consequently, as proven in (Sun et al., 2012), the user's current assets $a(z_V)$ decompose additively on $\mathcal{T}$. Further, if the user makes a structure-preserving trade to change p to x, the user's new assets

$$y(z_V) = a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \tag{8}$$

decompose additively on $\mathcal{T}$.

The expected assets $\mu_Y = \sum_{z_V} g(z_V) y(z_V)$ can be computed efficiently by junction tree propagation (Nilsson, 2001).

We assume the user has a logarithmic utility function

$$u(z_V) = \log y(z_V) = \log \left(a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \right). \tag{9}$$

We seek to maximize

$$EU = \sum_{z_V} q(z_V) \log \left(a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \right) \tag{10}$$

The utility (9) does not decompose additively, and exact computation of the expected utility is intractable. However, we can approximate the utility by the first-order Taylor expansion of $\log y(z_V)$ around μ_Y as:

$$u(z_V) \approx \log y(z_V) + \frac{(y(z_V) - \mu_Y)}{\mu_Y} - \frac{(y(z_V) - \mu_Y)^2}{2\mu_Y^2}. \tag{11}$$

We therefore wish to optimize

$$\begin{aligned} EU &\approx \sum_{z_V} \log q(z_V) \left(y(z_V) - \frac{(y(z_V) - \mu_Y)^2}{2\mu_Y^2} \right) \\ &= \log \mu_Y - \frac{\sigma_Y^2}{2\mu_Y^2}, \end{aligned} \tag{12}$$

where σ_Y^2 is the variance of Y. The objective function (12) can be calculated from the first and second moments of Y. Nilsson (2001) showed how to modify the standard junction tree propagation algorithm to compute first and second moments of additively decomposable functions efficiently. These results can be applied to efficient calculation of the approximate expected utility (12), as described below.

5.3 PROPAGATING SECOND MOMENTS

The standard junction tree algorithm (Jensen, 2001; Dawid, 1992; Lauritzen and Spiegelhalter, 1988) operates on a potential representation $\{h_B : B \in \mathcal{C} \cup \mathcal{D}\}$ for a probability distribution g that factorizes on a junction tree $\mathcal{T}$. A clique C is said to be eligible to receive from a neighboring clique D if either C is D's only neighbor or D has already received a message from all its neighbors other than C. Any schedule is allowable that sends messages along arcs only when the recipient is eligible to receive from the sender, and that terminates when messages have been sent in both directions along all arcs in the junction tree.

Passing a message from D to C has the following effect:

$$h'_S(z_S) = \sum_{D \setminus S} h_D(z_D), and \tag{13}$$

$$h'_C(z_C) = h_C(z_C) \left(\frac{h'_S(z_S)}{h_S(z_S)} \right) \tag{14}$$

That is, the new separator potential is obtained by marginalizing the sender's potential over the variables not contained in the separator, and the new recipient clique potential is obtained by multiplying the old recipient potential by the ratio of new to old separator potentials.

It is clear that message passing preserves the factorization constraint (6). Furthermore, when the algorithm terminates, the new clique and separator potentials are the marginal distributions $g_B(z_B) = \sum_{V \setminus B} g_V(z_V), B \in C \cup S$.

Now, suppose in addition to the multiplicative potential representation for g, we have an additive potential representation $\{t_B : B \in \mathcal{C} \cup \mathcal{S}\}$ for an additively decomposable function y on Ω_V. We modify the message-passing algorithm to pass a bivariate message along each arc. Now, in addition to the effects on the multiplicative potential for the probability distribution g, a message from D to C results in the following change to the additive potential for y:

$$t'_S(z_S) = \frac{\sum_{D \setminus S} h_D(z_D) t_D(z_D)}{\sum_{D \setminus S} h_D(z_D)}, and \tag{15}$$

$$t'_C(z_C) = t_C(z_C) + t'_S(z_S) - t_S(z_S) \tag{16}$$

After the algorithm terminates with messages sent in both directions along all arcs, the final clique and separator multiplicative potentials contain the associated marginal distributions; and the clique and separator additive potentials contain the conditional expectation of Y given the clique/separator state $\mu_{Y|B}(z_B) = \sum_{V \setminus B} g_V(z_V) y_V(z_V) / \sum_{V \setminus B} g_V(z_V), B \in C \cup S$.

After propagation, finding the first moment of Y is straightforward: we simply marginalize the additive potential on any clique or separator:

$$\mu_Y = \sum_B \mu_{Y|B}(z_B) \tag{17}$$

Recall that after propagation, the multiplicative potentials are the marginal probabilities $g_B(z_B)$ and the multiplicative potentials are the conditional expectations $\mu_{Y|B}(z_B)$ for $B \in C \cup S$. Nilsson (2001) proved that the second moment of the additively decomposable function Y can be calculated from the post-propagation additive and multiplicative potentials as follows:

$$\begin{aligned} E[Y^2] = &\sum_{C \in \mathcal{C}} \sum_{z_C} g_C(z_C) \mu_{Y|C}(z_C)^2 \\ &- \sum_{S \in \mathcal{S}} \sum_{z_S} g_D(z_S) \mu_{Y|S}(z_S)^2. \end{aligned} \tag{18}$$

To see why (18) is correct, note that by definition, $E[Y^2] = \sum_{z_u} g(z_u) y(z_u)^2$. Then:

$$\begin{aligned} E[Y^2] &= \sum_{z_u} g(z_u) y(z_u)^2 \\ &= \sum_{z_u} g(z_u) y(z_u) \left(\sum_{C \in \mathcal{C}} \mu_{Y|C}(z_C) - \sum_{S \in \mathcal{S}} \mu_{Y|S}(z_S) \mu \right) \\ &= \sum_{C \in \mathcal{C}} \sum_{z_u} \{ g_u(z_u) y(z_u) \mu_{Y|C}(z_C) \} \\ &\quad - \sum_{D \in \mathcal{D}} \sum_{z_u} \{ g_u(z_u) y(z_u) \mu_{Y|S}(z_S) \} \\ &= \sum_{C \in \mathcal{C}} \{ \sum_{z_C} \mu_{Y|C}(z_C) \sum_{z_{U \setminus C}} g_u(z_u) y(z_u) \} \\ &\quad - \sum_{S \in \mathcal{S}} \{ \sum_{z_S} \mu_{Y|S}(z_S) \sum_{z_{U \setminus S}} \{ g_u(z_u) y(z_u) \} \\ &= \sum_{C \in \mathcal{C}} \sum_{z_C} g_C(z_C) \mu_{Y|C}(z_C)^2 \\ &\quad - \sum_{S \in \mathcal{S}} \sum_{z_S} g_S(z_S) \mu_{Y|S}(z_S)^2. \end{aligned}$$

For a more accurate approximation, we can add additional terms to the Taylor expansion and apply Cowell's method (1999, Sec. 6.4.7) for local propagation of higher moments. In general, we need to propagate $n + 1$ potentials to compute the first n moments of the distribution of Y.

6 SUMMARY

By any definition, big data requires learners to consider only a portion of the data at a time. The problem is then to fuse the beliefs of these specialized agents. We consider a market to be an effective mechanism for fusing the beliefs of an arbitrary mixture of human experts and machine learners, by allowing each agent to concentrate on those areas where they can do the most good (and therefore earn the most points). However, agents that are good at learning are not necessarily good at trading. Our contribution is a general formulation of an agent which will translate a set of beliefs into optimal or near-optimal trades on a combinatorial market that has been represented in factored form such as a Bayesian network.

It is known from theory and empirical results in evolutionary finance that an informed trader seeking to maximize her wealth should allocate her assets according to the Kelly (1956) rule. This rule is very general, and applies even to combinatorial markets. However, it would be intractable to solve on an arbitrary joint state. We have derived two efficient ways to calculate Kelly investments given beliefs specified in a factored joint distribution – such as a Bayesian network. We tested the more conservative rule and found that it has the desired properties: an exponential wealth increase which never goes broke, slightly underperforming an EV-maximizer in the short run, but outperforming it in the long run because the EV-maximizer will go broke.

Our results mean that we can let experts focus on their beliefs in any effective manner, and let an automated agent convert those beliefs into trades. This ability has frequently been requested by participants in the IARPA ACE geopolitical forecasting tournament, and we expect to offer the feature this autumn in our new Science & Technology market.

As a reminder, we assume we have a current set of beliefs from our expert. In practice, we will have to specify a rate at which expressed beliefs converge to the market, in the absence of future updates from the expert, so as not forever to chain our participants with the ghosts of expired beliefs. In addition, we plan to extend our results for systems using approximate inference to update the probability distribution.

Acknowledgments

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20062. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Amir, R., Evstigneev, I. V., Hens, T., and Schenk-Hopp, K. R. (2001). Market selection and survival of investment strategies. Working Paper 91, University of Zurich. Institute for Empirical Research in Economics.

Barbu, A. and Lay, N. (2011). An introduction to artificial prediction markets for classification. *arXiv:1102.1465*.

Beinlich, I., Suermondt, G., Chavez, R., and Cooper, G. (1989). The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceeding of 2nd European Conference on AI and Medicine*.

Berea, A., Twardy, C., Laskey, K., Hanson, R., and Maxwell, D. (2013). Daggre: An overview of combinatorial prediction markets. In *Proceedings of the Seventh International Conference on Scalable Uncertainty Management (SUM)*.

Chen, Y., Fortnow, L., Lambert, N., Pennock, D. M., and Wortman, J. (2008). Complexity of combinatorial market makers. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 190–199.

Chen, Y. and Pennock, D. M. (2010). Designing markets for prediction. *AI Magazine*, 31(4):42–52.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36.

Evstigneev, I. V., Hens, T., and Schenk-Hopp, K. R. (2006). Evolutionary stable stock markets. *Economic Theory*, 27(2):449–468.

Hanson, R. (2003). Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag, Berlin, 2001 edition.

Kelly, J. J. (1956). A new interpretation of information rate. *Bell System Technical Journal*, 35:917–926.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their applications to expert systems. In *Proceedings of the Royal Statistical Society, Series B.*, volume 50, pages 157–224.

Lensberg, T. and Schenk-Hoppé, K. R. (2007). On the evolution of investment strategies and the kelly rule — a darwinian approach. *Review of Finance*, 11:25–50.

Nilsson, D. (2001). The computation of moments of decomposable functions in probabilistic expert systems. In *booktitle = "Proceedings of the 3rd International Symposium on Adaptive Systems"*, pages 116–121.

Solomonoff, R. J. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432.

Sun, W., Hanson, R., Laskey, K., and Twardy, C. (2012). Probability and asset updating using bayesian networks for combinatorial prediction markets. In *Proceedings of the 28th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Wächter, A. and Biegler, L. T. (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Predicting Latent Variables with Knowledge and Data: A Case Study in Trauma Care

Barbaros Yet[1], William Marsh[1], Zane Perkins[2], Nigel Tai[3], Norman Fenton[1]

[1]School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, UK
[2]Centre for Trauma Sciences, Queen Mary, University of London, London, UK
[3]Royal London Hospital, London, UK

barbaros@eecs.qmul.ac.uk

Abstract. Making a prediction that is useful for decision makers is not the same as building a model that fits data well. One reason for this is that we often need to predict the true state of a variable that is only indirectly observed, using measurements. Such 'latent' variables are not present in the data and often get confused with measurements. We present a methodology for developing Bayesian network (BN) models that predict and reason with latent variables, using a combination of expert knowledge and available data. The method is illustrated by a case study into the prediction of acute traumatic coagulopathy (ATC), a disorder of blood clotting that can cause fatality following traumatic injuries. There are several measurements for ATC and previous models have predicted one of these measurements instead of the state of ATC itself. Our case study illustrates the advantages of models that distinguish between an underlying latent condition and its measurements, and of a continuing dialogue between the modeller and the domain experts as the model is developed using knowledge as well as data.

Keywords: Bayesian Networks, Latent Variables, Knowledge Engineering

This paper is published as an abstract only.

A Lightweight Inference Method for Image Classification

John Mark Agosta
johnmark.agosta@gmail.com

Preeti J. Pillai
preetipillai04@gmail.com

Abstract

We demonstrate a two phase classification method, first of individual pixels, then of fixed regions of pixels for scene classification—the task of assigning posteriors that characterize an entire image. This can be realized with a probabilistic graphical model (PGM), without the characteristic segmentation and aggregation tasks characteristic of visual object recognition. Instead the spatial aspects of the reasoning task are determined separately by a segmented partition of the image that is fixed before feature extraction. The partition generates histograms of pixel classifications treated as virtual evidence to the PGM. We implement a sampling method to learn the PGM using virtual evidence. Tests on a provisional dataset show good (+70%) classification accuracy among most all classes.

1 Introduction

Scene recognition is a field of computer understanding for classification of scene types by analysis of a visual image. The techniques employed for scene recognition are well known, relying on methods for image analysis and automated inference. The fundamental process is to assign probabilities over a defined set of categories—the scene characteristics—based on analysis of the current visual state. This paper shows the practicability of a lightweight approach that avoids much of the complexity of object recognition methods, by reducing the problem to a sequence of empirical machine learning tasks.

The problem we have applied this to is classification of scene type by analysis of a video stream from a moving platform, specifically from a car. In this paper we address aspects of spatial reasoning—clearly there is also a temporal reasoning aspect, which is not considered here. In figurative terms the problem may be compared with Google's *Streetview*® application. *Streetview's* purpose is to tell you what your surroundings look like by knowing your location. The scene recognition problem is the opposite: to characterize your location from what your surroundings look like.

In this paper we consider a classification scheme for images where the image is subject to classification in multiple categories. We will consider outdoor roadway scenes, and these classification categories:

1. surroundings, zoning, development (urban, residential, commercial, mountainous, etc.)
2. visibility (e.g., illumination and weather),
3. roadway type,
4. traffic and other transient conditions,
5. roadway driving obstacles.

An image will be assigned one label from each of the set of five categories.

1.1 Uses of Scene Classification

There are numerous uses where the automated classification assigned to a scene can help. The purpose of scene classification is to capture the gist of the current view from its assigned category labels. For example, how would you describe a place from what you see? Certainly this is different from what you would know from just the knowledge of your lat-long coordinates. These are some envisioned uses:

- A scene classification provides context. For example in making a recommendation, the context could be to consider the practicality of the request: For instance, "Do you want to get a latte now? This is not the kind of neighborhood for that."

- Supplement search by the local surroundings. For example, "Find me a winery in a built-up area." "Find me a restaurant in a remote place." "Find a park in a less-travelled residential area."
- Coming up with a score for the current conditions. How is the view from this place? How shaded or sunny is the area? What fraction of the surroundings are natural versus artificial? Taking this one step further, given an individual driver's ratings of preferred locations, suggest other desirable routes to take, possibly out of the way from a "best" route.
- Distributed systems could crowd-source their findings about nearby locations to form a comprehensive picture of an area. For example, "How far does this swarm (road-race, parade) extend?"

1.2 Relevant previous work

One of the earliest formulations of image understanding as a PGM is found in Levitt, Agosta, and Binford (1989) and Agosta (1990). The approach assumed an inference hierarchy from object categories to low-level image features, and proposed aggregation operators that melded top-down (predictive) with bottom-up (diagnostic) reasoning over the hierarchy.

The uses of PGMs in computer vision have expanded into a vast range of applications. Just to mention a couple of examples, L. Fei-Fei, Fergus and P. Perona (2003) developed a Bayesian model for learning new object categories using a "constellation" model with terms for different object parts. In a paper that improved upon this, L. Fei-Fei and P. Perona (2005) proposed a Bayesian hierarchical theme model that automatically recognizes natural scene categories such as forest, mountains, highway, etc. based on a generalization of the original texton model by T. Leung and J. Malik (2001) and, L. Fei-Fei R. VanRullen, C. Koch, and P. Perona (2002). In another application of a Bayesian model, Sidenbladh, Black, and Fleet (2000) develop a generative model for the appearance of human figures. Both of these examples apply model selection methods to what are implicitly PGMs, if not explicitly labeled as such.

Computer vision approaches specifically to scene recognition recognize the need to analyze the image as a whole. Hoiem, Efros, and Hebert (2008) approach the problem by combining results from a set of *intrinsic images*, each a map of the entire image for one aspect of the scene. Oliva and Torralba (2006) develop a set of scene-centered global image features that capture the spatial layout properties of the image. Similar to our approach, their method does not require segmentation or grouping steps.

1.3 How Scene Classification differs from Object Recognition

Scene classification implies a holistic image-level inference task as opposed to the task of recovering the identity, presence, and pose of objects within an image. Central to object recognition is to distinguish the object from background of the rest of the image. Typically this is done by segmenting the image into regions of smoothly varying values separated by abrupt boundaries, using a bottoms-up process. Pixels may be grouped into "super-pixels" whose grouping is further refined into regions that are distinguished as part of the foreground or background. Object recognition then considers the features and relationships among foreground regions to associate them with parts to be assembled into the object, or directly with an entire object to be recovered.

Scene classification as we approach it does not necessarily depend on segmenting the image into regions, or identifying parts of the image. Rather it achieves a computational economy by treating the image as a whole; for example, to assign the image to the class of "indoor," "outdoor," "urban landscape," or "rural landscape," etc. from a set of pre-defined categories. We view classification as assigning a posterior to class labels, where the image may be assigned a value over multiple sets of labels; equivalently, the posterior may be a joint distribution over several scene variables.

Despite the lack of a bottoms-up segmentation step in our approach, our method distinguishes regions of the image by a partition that is prior to analyzing the image contents. This could be a fixed partition, which is appropriate for a camera in a fixed location such as a security camera, or it could depend on inferring the geometry of the location from sources distinct from the image contents, such as indicators of altitude and azimuth of the camera. In our case, the prior presumption is that the camera is on the vehicle, facing forward, looking at a road.

The rest of this paper is organized as follows. Section 2 describes the inference procedure cascade; the specific design and learning of the Bayes network PGM is the subject of Section 3, and the results of the learned model applied to classification of a set of images is presented in Section 4.

2 Lightweight inference with virtual evidence

In treating the image as a whole, our approach to inference for scene classification takes place by a sequence of two classification steps:

- First the image's individual pixels are classified, based on pixel level features. This classifier resolves the pixel into one of n discrete types, representing the kind of surface that generated it. In our examples $n = 8$: sky, foliage, building-structure, road-surface, lane, barrier-sidewalk, vehicle, and pedestrian.

- In the second step, the pre-defined partitions are applied to the image and in each partition the pixel types are histogrammed, to generate a likelihood vector for the partition. These likelihoods are interpreted as virtual evidence[1] for the second level image classifier, the scene classifier, implemented as a PGM. The classifier returns an joint distribution over the scene variables, inferred from the partitions' virtual evidence.

There is labeled data for both steps, to be able to learn a supervised classifier for each. Each training image is marked up into labeled regions using the open source *LabelMe* tool, (Russell, Torralba, K. Murphy and Freeman, 2007) and also labeled by one label from each category of scene characteristics. From the region labelings a dataset of pixels, with color and texture as features, and the region they belong to as labels can be created. In the second step we learn the structure and parameters of a Bayes network—a discrete valued PGM—from the set of training images that have been manually labeled with scene characteristics. Each image has one label assigned for each scene characteristic. The training images are reduced to a set of histograms of the predicted labels for the pixels, one for each partition. The supervised data for an image consists of the histogram distributions and the label set.

Scene recognition output is a summarization of a visual input as an admittedly modest amount of information from a input source orders of magnitude greater–even mores than for the object recognition task. From the order of 10^6 pixel values we infer a probability distribution over a small number of discrete scene classification variables. To obtain computational efficiency, we've devised an approach that summarizes the information content of the image in an early stage of the process that is adequate at later stages for the classification task.

2.1 Inference Cascade

The two phases in the inference cascade can be formalized as follows, starting from the pixel image and resulting in a probability distribution over scene characteristics. Consider an image of pixels p_{ij} over $i \times j$, each pixel described by a vector of features $\mathbf{f}_{ij}$. The features are derived by a set of filters, e.g. for color and texture, centered at coordinate (i, j). A pixel-level classifier is a function from the domain of $\mathbf{f}$ to one of a discrete set of n types, $C : \mathbf{f} \rightarrow \{c^{(1)}, \cdots c^{(n)}\}$. The result is an array of classified image pixels.

A pre-determined segmentation, G_m partitions the pixels in the image into M regions by assigning each pixel to one region, $r_m = \{p_{ij} \,|\, p_{ij} \in G_m\}, m = 1 \ldots M$, to form regions that are contiguous sets of pixels. Each region is described by a histogram of the pixel types it contains: $H_m = \left(|C(\mathbf{f}_{ij}) = c^{(1)}|, \cdots |C(\mathbf{f}_{ij}) = c^{(n)}|\right) s.t. \mathbf{f}_{ij} \in G_m$, for which we introduce the notation, $H_m = \left(|c_{ij}^{(1)}|_m, \cdots |c_{ij}^{(n)}|_m\right)$, where $|c^{(i)}|_m$ denotes the count of pixels of type $c^{(i)}$ in region m. The scene classifier is a PGM with virtual evidence nodes corresponding to the M regions of the image. See Figure 3. Each evidence node receives virtual evidence in the form of a lambda message, λ_m, with likelihoods in the ratios given by H_m. The PGM model has a subset of nodes $\mathbf{S} = \{S_1, \cdots S_v\}$, distinct from its evidence nodes, for scene characteristic variables, each with a discrete state space. Scene classification is completely described by $\mathsf{P}(\mathbf{S} \,|\, \lambda_1, \cdots \lambda_M)$, the joint of $\mathbf{S}$ when the λ_m are applied, or by a characterization of the joint by the MAP configuration over $\mathbf{S}$, or just the posterior marginals of $\mathbf{S}$.

2.2 Partitions of Pixel-level Data

As mentioned we avoid segmenting the image based on pixel values by using a fixed partition to group classified pixels. We introduce a significant simplification over conventional object recognition methods by using such a segmentation. This makes sense because we are not interested in identifying things that are in the image, but only in treating the image as a whole. For instance in the example we present here, the assumption is that the system is classifying an outdoor roadway scene, with sky above, road below, and surroundings characteristic of the scene to either side. The partitions approximate this division. The image is partitioned symmetrically into a set of twelve wedges, formed by rays emanating from the image center.

For greater efficiency the same method could be applied over a smoothed, or down-sampled image, so that every pixel need not be touched, only pixels on a regular grid. The result of the classification step is a *discrete class-valued image* array. See Figure 2. Despite the classifier ignoring local dependencies, neighboring pixels tend to be classed similarly, and the class-valued

[1] Sometimes called "soft evidence." We prefer the term virtual evidence, since soft evidence is also used to mean an application of Jeffrey's rule of conditioning that can change the CPTs in the network.

image resembles a cartoon version of the original.

Figure 1: The original image. The barriers bordering the lane are a crucial feature that the system is trained to recognize.

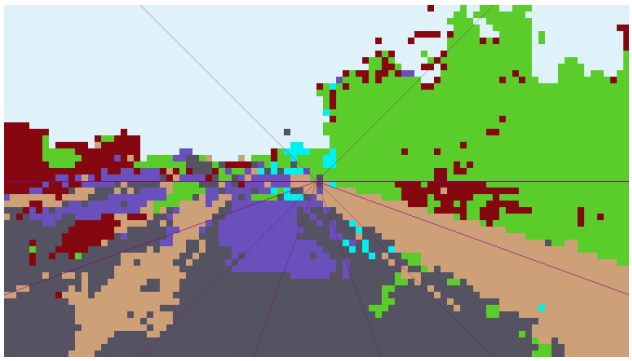


Figure 2: The image array of $C(\mathbf{f}_{ij})$, the pixel classifier, on an image down-sampled to 96 by 54. Rays emanating from the image center show the wedge-shaped regions. Colors are suggestive of the pixel class, e.g. green indicates foliage and beige indicates barriers.

2.2.1 Inferring partition geometry

The point chosen as the image center, where the vertices of the wedges converge approximates the vanishing point of the image. Objects in the roadway scene tend to conform (very) roughly to the wedge outlines so that their contents are more uniform, and hence, likelihoods are more informative. For example, the contents of the image along the horizon will fall within one wedge, and the road surface within another.

2.3 The image as a source of virtual evidence

For each wedge that partitions the image, the evidence applied to the Bayes network from the wedge m is: $\lambda_m \propto |c_{ij}^{(1)}|_m : |c_{ij}^{(2)}|_m : \cdots : |c_{ij}^{(n)}|_m$. One typically thinks of virtual evidence as a consequence of measurements coming from a sensor that garbles the precise value of the quantity of interest—where the actual observed evidence value is obscured by an inaccuracy in the sensor reading. Semantically, one should not think of the virtual image evidence as a garbled sensor variable. Rather it *is* the evidence that describes the region.

3 Bayes network design

Formally a Bayes network is a factorization of a joint probability distribution into local probability models, each corresponding to one node in the network, with directed arcs between the nodes showing the conditioning of one node's probability model on another's (Koller and Friedman, 2010). Inference—for example, classification—operates in the direction against the causal direction of the arc. In short, inference flows from lower level evidence in the network upward to the class nodes at the top of the network where it generates the posterior distributions over the class variables, in this case, the scene characteristics. We learn a fully observable Bayes network with virtual evidence for scene classification.

3.1 How the structure and parameters are defined

The design of the Bayes network model is fluid: It is easily re-learned under different partition inputs, output categories and structural constraints. The ability to easily modify the model to test different kinds of evidence as inputs, or differently defined nodes as outputs is an advantage of this approach. The structure of the model discovers dependencies among the model variables that reveal properties of the domain.

Learning the Bayes network is composed of two aspects; the first, learning the variables' structure, the second, learning the parameters of the variable conditional probability tables. The algorithm used is SMILE's *Bayesian Search* (Druzdzel et al., 1997), a conventional fully observable learning algorithm, with a Bayesian scoring rule used to select the preferred model. Learning structure and parameters occur simultaneously.

The model is structured into two levels, the top level of outputs and the lower level of inputs as shown in Figure 3. This is the canonical structure for classification with a Bayes network, in this case a multi-classifier with multiple output nodes. In the learning procedure this node ordering is imposed as a constraint on the structure, so that conditioning arcs cannot go from the lower level to the upper level.

Further constraints are used to limit in-degree and node ordering. The in-degree of evidence nodes is

limited to two. Node ordering of output nodes follows common sense causal reasoning: for instance, the "Surroundings" variable influences the "Driving Conditions" and not the other way around. The model consequently follows an approximately näive Bayes structure for each scene variable, but with additional arcs that are a consequence of the model selection performed during learning. The resulting network is relatively sparse and hence learning a network of this size, let alone running inference on it can be done interactively.

3.2 Bayes Network Learning Dataset

An interesting challenge in learning this model is that there is no conventional procedure for learning from virtual evidence, such as the histogram data.

3.2.1 Consideration of partition contents as virtual evidence

We considered three ways to approximate learning the Bayes network from samples that include virtual evidence.

1) Convert the dataset into an approximate equivalent observed evidence dataset by generating multiples of each evidence row, in proportion to the likelihood fraction for each state of the virtual evidence. If there are multiple virtual evidence nodes, then to capture dependencies among virtual evidence nodes this could result in a combinatorial explosion of row sets, one multiple for each combination of virtual evidence node states, with multiplicities in proportion to the likelihood of the state combination. This is equivalent in complexity to combining all virtual evidence nodes into one node for sampling.

Similarly one could sample from the combination of all virtual evidence nodes and generate a sample of rows based on the items in the sample. This is a bit like logic sampling the virtual states.

Both these methods make multiple copies of a row in the learning set as a way to emulate a training weight. Instead one could apply a weight to each row in the sampled training set, in proportion to its likelihood.

2) One could also consider a mixture, a "multi-net," of learned deterministic evidence models. The models would have the same structure, so the result would be a mixture of CPTs, weighted (in some way) by the likelihoods. It appears this would also suffer a combinatorial explosion of mixture components, and might be amenable to reducing the set by sampling.

3) Alternatively, one could consider the virtual evidence by a virtual node that gets added as a child to the evidence node, which is then instantiated to send the equivalent lambda msg to its parent. This is the method used in Refaat, Choi and Darwiche (2012). With many cases, there would be a set of virtual nodes added to the network for each case, again generating a possibly unmanageable method. Perhaps there is an incremental learning method that would apply: Build a network with one set of nodes, do one learning step, then replace the nodes with the next set, and repeat a learning step.

4 Results on a sample dataset

In this section we present the evaluation of the Bayes network as a classifier. We argue that the first-stage pixel-level classifier, whose accuracy approaches 90%, is a minor factor in the scene classification results, since the partition-level inputs to the Bayes network average over a large number of pixels, although this premise could be tested.

4.1 Learning from a sampled dataset

The sample dataset to learn the model was a further approximation on alternative 1), where each virtual evidence node was sampled independently to convert the problem into an equivalent one with sampled data. Each histogram was sampled according to its likelihood distribution, to generate a set of conventional evidence samples that approximated the histogram. The result was an expanded dataset that multiplied the number rows by the sample size for each row in the histogram dataset. The resulting dataset description is:

1. Original data set: 122 rows of 12 region histograms of images labeled by 5 scene labels.
2. Each region histogram is sampled 10 times, to generate 1220 rows
3. Final data set of 5 labels and 12 features by 1220 rows

4.2 Inference Results

As mentioned, the second-stage Bayes network classifier infers a joint probability distribution over the set of scene characteristic nodes—the nodes shown in orange in Figure 3. We will evaluate the scene classifier by the accuracy of the predicted marginals, comparing the highest posterior prediction for each scene variable with the true value.[2]

[2]The "dynamic environment" variable is not counted in the evaluation results, since most all labeled data was collected under overcast conditions, making the predicted results almost always correct, and uninteresting.

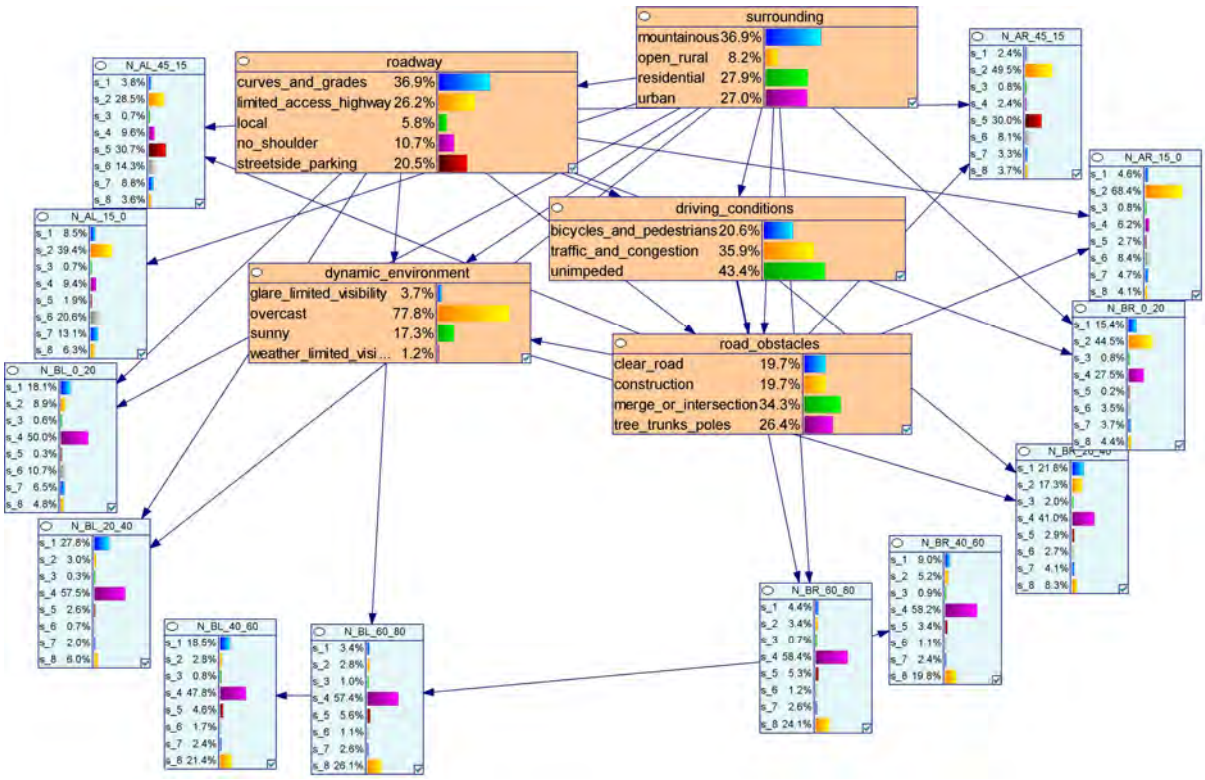


Figure 3: The entire Bayes network used for scene classification. Input nodes, corresponding to the wedges that partition the image are shown in light blue, and output nodes for the scene variables are in orange. The input nodes are arranged roughly in the positions of the corresponding wedges in the image. The input node histograms show the virtual evidence applied from that wedge. The labels used here for virtual evidence states, $s\_1, \ldots s\_8$ correspond to the classifier outputs $c^{(1)}, \ldots c^{(8)}$.

The matrix of counts of the true class by the predicted class is called a confusion matrix. The row sum of the confusion matrix for any class divided into the diagonal (true count) is the fraction of correct cases out of those possible, known as the recall or the coverage for that class. The column sum divided into the diagonal element is the fraction classified with that columns class label that truly belong to that class, which is called the precision. Tables $1-4$ show the recall and precision for each class, for each of the scene variables. As may be expected "Surroundings" that takes in the entire image performs better than "Road obstacles" that requires attention to detail in just the car's lane. This poor performance is even more true with "Bicycles and pedestrians," in Table 3 that appear in small areas of the image. In other classes either precision or recall approach 1.0, except for "Local" roads, where all cases were confused with "Curves and grades," again due to the limited variety in the training set.

Beyond evaluating the accuracy of marginal predictions, we can also make observations about the structure learned for the Bayes network. Arcs in the learned model show which wedge histograms are relevant to which scene variables. These arcs are relatively sparse, in part due to the afore-mentioned design constraint in-degree arc limit of two. The arcs chosen by the structure learning algorithm show a strong association between the location of the partitions, and different scene variables. We see this in the associations where the "Driving conditions" scene variable connects to partitions at the base of the image, and "Surroundings" connects to partitions on the image periphery. The relevance of the two wedges at the bottom of the diagram is limited, since their only incoming arcs are from other wedges, indicating that their evidence is supported entirely by neighboring wedges. We leave them in the model, since in the case of virtual evidence they will still have some information value for classification. Further along these lines, in terms of wedge dependencies, only one arc was learned between wedge histograms, indicating that the evidence contributed to the scene is conditionally independent in all but this case. The sub-network of scene variables is more connected, indicating strong dependencies among the

scene variables. Some of these are to be expected, for instance "Curves and grades" correlates strongly with "Mountainous" surroundings. Some are spurious, as a result of biased selection of the training sample images, (e.g. all divided highway images corresponded to overcast scenes) and have been corrected by adding more samples.

4.3 Discussion and Conclusion

We have demonstrated a novel scene classification algorithm that takes advantage of the presumed geometry of the scene to avoid computationally expensive image processing steps characteristic of object recognition methods, such as pixel segmentation, by a cascade of a pixel level and fixed partition level multi-classifier, for which we learn a Bayes network. As a consequence of the partition-level data we learn the Bayes network with virtual evidence.

The Bayes network classifies the scene in several dependent dimensions corresponding to a set of categories over which a joint posterior of scene characteristics is generated. Here we have only considered the marginals over categories, however it is a valid question whether a MAP interpretation—of the most likely combination of labels—is more appropriate.

The use of virtual evidence also raises questions about whether it is proper to consider the virtual evidence likelihood as a convex combination of "pure" image data. Another interpretation is that the histograms we are using are better"sliced and diced" to generate strong evidence from certain ratios of partition content. For instance a partition that includes a small fraction of evidence of roadway obstacles—think evidence of a small person—may be a larger concern than a partition obviously full of obstacles, and should not be considered a weaker version of the extreme partition contents. These subtleties could be considered as we expand the applicability of the system. In this early work it suffices that given the approximations, useful and accurate results can be achieved at modest computational cost.

Acknowledgements

This work would not have been possible without valuable discussions with Ken Oguchi and Joseph Dugash, and by Naiwala Chandrasiri, Saurabh Barv, Ganesh Yalla, and Yiwen Wan.

References

Agosta, J. M., 1990. The structure of Bayes networks for visual recognition, UAI (1988) North-Holland Publishing Co., pp. 397 - 406.

Druzdzel, M. et. al., 1997. SMILE (Structural Modeling, Inference, and Learning Engine) http://genie.sis.pitt.edu/.

Fei-Fei, L., R. Fergus, P. Perona, 2003. A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories (ICCV 2003), pp. 1134-1141 vol.2.

Fei-Fei L. and P. Perona, 2005. A Bayesian Hierarchical Model for Learning Natural Scene Categories. (CVPR 2005), pp. 524-531.

Fei-Fei, L., R. VanRullen, C. Koch, and P. Perona, 2002. Natural scene categorization in the near absence of attention(PNAS 2002), 99(14):95969601.

Hoiem, D., A. A. Efros, and M. Hebert. 2008. Closing the Loop in Scene Interpretation.2008 IEEE Conference on Computer Vision and Pattern Recognition (June): 18.

Koller, D., and Friedman, N. 2010. Probabilistic Graphical Models: Principles and Techniques. Cambridge, Massachusetts: The MIT Press.

Leung, T. and J. Malik, 2001. Representing and recognizing the visual appearance of materials using three-dimensional textons(IJCV 2001), 43(1):2944.

Levitt, T., J. M. Agosta, T.O. Binford, 1989. Model-based influence diagrams for machine vision, (UAI 1989).

Oliva, A., and A. Torralba. 2006. Building the Gist of a Scene: The Role of Global Image Features in Recognition.Progress in Brain Research 155 (January): 2336.

Oliva, A., A. Torralba, 2001. Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision, Vol. 42(3): 145-175.

Refaat, K. S. , A. Choi and A. Darwiche, 2012. New Advances and Theoretical Insights into EDML. (UAI 2012), pp. 705-714 .

Russell, B., A. Torralba, K. Murphy, W. T. Freeman, 2007. "LabelMe: a database and web-based tool for image annotation" International Journal of Computer Vision.

Sidenbladh, H., M. J. Black, and D. J. Fleet, 2000. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. ECCV 2000: 702718.

	Mountainous	Open rural	Residential	Urban
Recall	1.0	0.9	0.794	0.45
Precision	0.642	1.0	0.964	1.0
Accuracy				0.784

Table 1: Surroundings

	Curves and grades	Limited access highway	Local	No shoulder	Streetside parking
Recall	1.0	0.75	0.0	0.85	0.56
Precision	0.61	1.0	NaN	1.0	1.0
Accuracy					0.770

Table 2: Roadways

	Bicycles and pedestrians	Traffic and congestion	Unimpeded
Recall	0.56	0.6	0.98
Precision	0.875	0.93	0.67
Accuracy			0.754

Table 3: Driving Conditions

	Clear road	Construction	Merge intersection	Tree trunks and poles
Recall	0.42	0.96	0.6	1.0
Precision	1.0	0.92	0.86	0.55
Accuracy				0.738

Table 4: Road Obstacles

Product Trees for Gaussian Process Covariance in Sublinear Time

David A. Moore
Computer Science Division
University of California, Berkeley
Berkeley, CA 94709
dmoore@cs.berkeley.edu

Stuart Russell
Computer Science Division
University of California, Berkeley
Berkeley, CA 94709
russell@cs.berkeley.edu

Abstract

Gaussian process (GP) regression is a powerful technique for nonparametric regression; unfortunately, calculating the predictive variance in a standard GP model requires time $O(n^2)$ in the size of the training set. This is cost prohibitive when GP likelihood calculations must be done in the inner loop of the inference procedure for a larger model (e.g., MCMC). Previous work by Shen et al. (2006) used a k-d tree structure to approximate the predictive mean in certain GP models. We extend this approach to achieve efficient approximation of the predictive covariance using a tree clustering on pairs of training points. We show empirically that this significantly increases performance at minimal cost in accuracy. Additionally, we apply our method to "primal/dual" models having both parametric and nonparametric components and show that this enables efficient computations even while modeling longer-scale variation.

1 Introduction

Complex Bayesian models often tie together many smaller components, each of which must provide its output in terms of probabilities rather than discrete predictions. As a natively probabilistic technique, Gaussian process (GP) regression (Rasmussen and Williams, 2006) is a natural fit for such systems, but its applications in large-scale Bayesian models have been limited by computational concerns: training a GP model on n points requires $O(n^3)$ time, while computing the predictive distribution at a test point requires $O(n)$ and $O(n^2)$ operations for the mean and variance respectively.

This work focuses specifically on the fast evaluation of GP likelihoods, motivated by the desire for efficient inference in models that include a GP regression component. In particular, we focus on the predictive covariance, since this computation time generally dominates that of the predictive mean. In our setting, training time is a secondary concern: the model can always be trained offline, but the likelihood evaluation occurs in the inner loop of an ongoing inference procedure, and must be efficient if inference is to be feasible.

One approach to speeding up GP regression, common especially to spatial applications, is the use of covariance kernels with short lengthscales to induce sparsity or near-sparsity in the kernel matrix. This can be exploited directly using sparse linear algebra packages (Vanhatalo and Vehtari, 2008) or by more structured techniques such as space-partitioning trees (Shen et al., 2006; Gray, 2004); the latter approaches create a query-dependent clustering to avoid considering regions of the data not relevant to a particular query. However, previous work has focused on efficient calculation of the predictive mean, rather than the variance, and the restriction to short lengthscales also inhibits application to data that contain longer-scale variations.

In this paper, we develop a tree-based method to efficiently compute the predictive covariance in GP models. Our work extends the weighted sum algorithm of Shen et al. (2006), which computes the predictive mean. Instead of clustering points with similar weights, we cluster *pairs* of points having similar weights, where the weights are given by a kernel-dependent distance metric defined on the *product space* consisting of all pairs of training points. We show how to efficiently build and compute using a *product tree* constructed in this space, yielding an adaptive covariance computation that exploits the geometric structure of the training data to avoid the need to explicitly consider each pair of training points. This enables us to present what is to our knowledge the first account of GP regression in which the major test-time operations

(predictive mean, covariance, and likelihood) run in time sublinear in the training set size, given a suitably sparse kernel matrix. As an extension, we show how our approach can be applied to GP models that combine both parametric and nonparametric components, and argue that such models present a promising option for modeling global-scale structure while maintaining the efficiency of short-lengthscale GPs. Finally, we present empirical results that demonstrate significant speedups on synthetic data as well as a real-world seismic dataset.

2 Background

2.1 GP Regression Model

We assume as training input a set of labeled points $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, where we suppose that

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

for some unknown function $f(\cdot)$ and i.i.d. Gaussian observation noise $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. Treating the estimation of $f(\cdot)$ as a Bayesian inference problem, we consider a Gaussian process prior distribution $f(\cdot) \sim GP(0, k)$, parameterized by a positive-definite *covariance* or *kernel* function $k(x, x')$. Given a set X^* containing m test points, we derive a Gaussian posterior distribution $f(X^*) \sim \mathcal{N}(\mu_*, \Sigma_*)$, where

$$\mu_* = K^{*T} K_y^{-1} \mathbf{y} \quad (1)$$

$$\Sigma_* = K^{**} - K^{*T} K_y^{-1} K^* \quad (2)$$

and $K_y = K(X, X) + \sigma_n^2 I$ is the covariance matrix of training set observations, $K^* = k(X, X^*)$ denotes the $n \times m$ matrix containing the kernel evaluated at each pair of training and test points, and similarly $K^{**} = k(X^*, X^*)$ gives the kernel evaluations at each pair of test points. Details of the derivations, along with general background on GP regression, can be found in Rasmussen and Williams (2006).

In this work, we make the additional assumption that the input points $\mathbf{x}_i$ and test points $\mathbf{x}_p^*$ lie in some metric space $(\mathcal{M}, d)$, and that the kernel is a monotonically decreasing function of the distance metric. Many common kernels fit into this framework, including the exponential, squared-exponential, rational quadratic, and Matérn kernel families; anisotropic kernels can be represented through choice of an appropriate metric.

2.2 k-d and Metric Trees

Tree structures such as k-d trees (Friedman et al., 1977) form a hierarchical, multiresolution partitioning of a dataset, and are commonly used in machine learning for efficient nearest-neighbor queries. They have also been adapted to speed up nonparametric regression (Moore et al., 1997; Shen et al., 2006); the general approach is to view the regression computation of interest as a sum over some quantity associated with each training point, weighted by the kernel evaluation against a test point. If there are sets of training points having similar weight – for example, if the kernel is very wide, if the points are very close to each other, or if the points are all far enough from the query to have effectively zero weight – then the weighted sum over the set of points can be approximated by an unweighted sum (which does not depend on the query and may be precomputed) times an estimate of the typical weight for the group, saving the effort of examining each point individually. This is implemented as a recursion over a tree structure augmented at each node with the unweighted sum over all descendants, so that recursion can be cut off with an approximation whenever the weight function is shown to be suitably uniform over the current region.

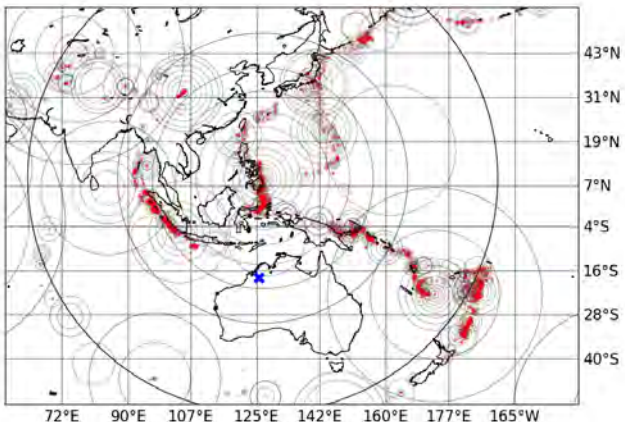


Figure 1: Cover tree decomposition of seismic event locations recorded at Fitzroy Crossing, Australia (with X marking the station location).

Major drawbacks of k-d trees include poor performance in high dimensions and a limitation to Euclidean spaces. By contrast, we are interested in non-Euclidean metrics both as a matter of practical application (e.g., in a geophysical setting we might consider points on the surface of the earth) and because some choices of kernel function require our algorithm to operate under a non-Euclidean metric even if the underlying space is Euclidean (see section 3.2). We therefore consider instead the class of trees having the following properties: (a) each node $\mathbf{n}$ is associated with some point $x_\mathbf{n} \in \mathcal{M}$, such that all descendants of $\mathbf{n}$ are contained within a ball of radius $r_\mathbf{n}$ centered at $x_\mathbf{n}$, and (b) for each leaf $\mathbf{L}$ we have $x_\mathbf{L} \in X$, with exactly one leaf node for each training point $\mathbf{x}_i \in X$. We call any tree satisfying these properties a *metric tree*.

```
function WeightedMetricSum(node n, query points (x*_i, x*_j),
                           accumulated sum Ŝ, tolerances ε_rel, ε_abs)
    δ_n ← δ((x*_i, x*_j), (n_1, n_2))
    if n is a leaf then
        Ŝ ← Ŝ + (K_y^{-1})_n · ( k(d(x*_i, n_1)) · k(d(x*_j, n_2)) )
    else
        w_min ← k^prod_lower (δ_n + r_n)
        w_max ← k^prod_upper (max(δ_n − r_n, 0))
        if w_max · S^Abs_n ≤ ( ε_rel | Ŝ + w_min · S^UW_n | + ε_abs ) then
            Ŝ ← Ŝ + ½ (w_max + w_min) · S^UW_n
        else
            for each child c of n
             sorted by ascending δ((x*_i, x*_j), (c_1, c_2)) do
                Ŝ ← Ŝ + WeightedMetricSum(c, (x*_i, x*_j), Ŝ, ε_rel, ε_abs)
            end for
        end if
    end if
    return Ŝ
end function
```

Figure 2: Recursive algorithm to computing GP covariance entries using a product tree. Abusing notation, we use $\mathbf{n}$ to represent both a tree node and the pair of points $\mathbf{n} = (\mathbf{n}_1, \mathbf{n}_2)$ associated with that node.

Examples of metric trees include many structures designed specifically for nearest-neighbor queries, such as ball trees (Uhlmann, 1991) and cover trees (Beygelzimer et al., 2006), but in principle any hierarchical clustering of the dataset, e.g., an agglomerative clustering, might be augmented with radius information to create a metric tree. Although our algorithms can operate on any metric tree structure, we use cover trees in our implementation and experiments. A cover tree on n points can be constructed in $O(n \log n)$ time, and the construction and query times scale only with the *intrinsic* dimensionality of the data, allowing for efficient nearest-neighbor queries in higher-dimensional spaces (Beygelzimer et al., 2006). Figure 1 shows a cover-tree decomposition of one of our test datasets.

3 Efficient Covariance using Product Trees

We consider efficient calculation of the GP covariance (2). The primary challenge is the multiplication $K^{*T} K_y^{-1} K^*$. For simplicity of exposition, we will focus on computing the (i,j)th entry of the resulting matrix, i.e., on the multiplication $\mathbf{k_i^{*T}} K_y^{-1} \mathbf{k_j^*}$ where $\mathbf{k_i^*}$ denotes the vector of kernel evaluations between the training set and the ith test point, or equivalently the ith column of K^*. Note that a naïve implementation of this multiplication requires $O(n^2)$ time.

We might be tempted to apply the vector multiplication primitive of Shen et al. (2006) separately for each row of K_y^{-1} to compute $K_y^{-1}\mathbf{k_j^*}$, and then once more to multiply the resulting vector by $\mathbf{k_i^*}$. Unfortunately, this requires n vector multiplications and thus scales (at least) linearly in the size of the training set. Instead, we note that we can rewrite $\mathbf{k_i^{*T}} K_y^{-1} \mathbf{k_j^*}$ as a weighted sum of the entries of K_y^{-1}, where the weight of the (p,q)th entry is given by $k(\mathbf{x_i^*}, \mathbf{x_p}) k(\mathbf{x_j^*}, \mathbf{x_q})$:

$$\mathbf{k_i^{*T}} K_y^{-1} \mathbf{k_j^*} = \sum_{p=1}^{n} \sum_{q=1}^{n} (K_y^{-1})_{pq} k(\mathbf{x_i^*}, \mathbf{x_p}) k(\mathbf{x_j^*}, \mathbf{x_q}). \quad (3)$$

Our goal is to compute this weighted sum efficiently using a tree structure, similar to Shen et al. (2006), except that instead of clustering points with similar weights, we now want to cluster *pairs* of points having similar weights.

To do this, we consider the *product space* $\mathcal{M} \times \mathcal{M}$ consisting of all pairs of points from $\mathcal{M}$, and define a *product metric* δ on this space. The details of the product metric will depend on the choice of kernel function, as discussed in section 3.2 below. For the moment, we will assume a SE kernel, of the form $k_{SE}(d) = \exp(-d^2)$, for which a natural choice is the 2-product metric:

$$\delta((\mathbf{x}_a, \mathbf{x}_b), (\mathbf{x}_c, \mathbf{x}_d)) = \sqrt{d(\mathbf{x}_a, \mathbf{x}_c)^2 + d(\mathbf{x}_b, \mathbf{x}_d)^2}.$$

Note that this metric, taken together with the SE kernel, has the fortunate property

$$k_{SE}(d(\mathbf{x}_a, \mathbf{x}_b)) k_{SE}(d(\mathbf{x}_c, \mathbf{x}_d)) = k_{SE}(\delta((\mathbf{x}_a, \mathbf{x}_b), (\mathbf{x}_c, \mathbf{x}_d))),$$

i.e., the property that evaluating the kernel in the product space (rhs) gives us the correct weight for our weighted sum (3) (lhs).

Now we can run any metric tree construction algorithm (e.g., a cover tree) using the product metric to build a *product tree* on all *pairs* of training points. In principle, this tree contains n^2 leaves, one for each pair of training points. In practice it can often be made much smaller; see section 3.1 for details. At each leaf node $\mathbf{L}$, representing a pair of training points, we store the element $(K_y^{-1})_{\mathbf{L}}$ corresponding to those two training points, and at each higher-level node $\mathbf{n}$ we cache the unweighted sum $S_{\mathbf{n}}^{UW}$ of these entries over all of its descendant leaf nodes, as well as the sum of absolute values $S_{\mathbf{n}}^{Abs}$ (these cached sums will be used to determine when to cut off recursive calculations):

$$S_{\mathbf{n}}^{\mathrm{UW}} = \sum_{\mathbf{L} \in \mathrm{leaves}(\mathbf{n})} (K_y^{-1})_{\mathbf{L}} \quad (4)$$

$$S_{\mathbf{n}}^{\mathrm{Abs}} = \sum_{\mathbf{L} \in \mathrm{leaves}(\mathbf{n})} \left| (K_y^{-1})_{\mathbf{L}} \right|. \quad (5)$$

Given a product tree augmented in this way, the weighted-sum calculation (3) is performed by the WeightedMetricSum algorithm of Figure 2. This algorithm is similar to the WeightedSum and WeightedXtXBelow algorithms of Shen et al. (2006) and Moore et al. (1997) respectively, but

adapted to the non-Euclidean and non-binary tree setting, and further adapted to make use of bounds on the product kernel (see section 3.2). It proceeds by a recursive descent down the tree, where at each non-leaf node it computes upper and lower bounds on the weight of any descendant, and applies a cutoff rule to determine whether to continue the descent. Many cutoff rules are possible; for predictive mean calculation, Moore et al. (1997) and Shen et al. (2006) maintain an accumulated lower bound on the total overall weight, and cut off whenever the difference between the upper and lower weight bounds at the current node is a small fraction of the lower bound on the overall weight. However, our setting differs from theirs: since we are computing a weighted sum over entries of K_y^{-1}, which we expect to be approximately sparse, we expect that some entries will contribute much more than others. Thus we want our cutoff rule to account for the weights of the sum *and* the entries of K_y^{-1} that are being summed over. We do this by defining a rule in terms of the current running weighted sum,

$$w_{\max} \cdot S_{\mathbf{n}}^{\mathrm{Abs}} \leq \left(\epsilon_{\mathrm{rel}} \left|\hat{S} + w_{\min} \cdot S_{\mathbf{n}}^{\mathrm{UW}}\right| + \epsilon_{\mathrm{abs}}\right), \quad (6)$$

which we have found to significantly improve performance in covariance calculations compared to the weight-based rule of Moore et al. (1997) and Shen et al. (2006). Here $\hat{S}$ is the weighted sum accumulated thus far, and ϵ_{abs} and ϵ_{rel} are tunable approximation parameters. We interpret the left-hand side of (6) as computing an upper bound on the contribution of node $\mathbf{n}$'s descendents to the final sum, while the absolute value on the right-hand side gives an estimated lower bound on the magnitude of the final sum (note that this is not a true bound, since the sum may contain both positive and negative terms, but it appears effective in practice). If the leaves below the current node $\mathbf{n}$ appear to contribute a negligible fraction of the total sum, we approximate the contribution from $\mathbf{n}$ by $\frac{1}{2}(w_{\max} + w_{\min}) \cdot S_{\mathbf{n}}^{\mathrm{UW}}$, i.e., by the average weight times the unweighted sum. Otherwise, the computation continues recursively over $\mathbf{n}$'s children. Following Shen et al. (2006), we recurse to child nodes in order of increasing distance from the query point, so as to accumulate large sums early on and increase the chance of cutting off later recursions.

3.1 Implementation

A naïve product tree on n points will have n^2 leaves, but we can reduce this and achieve substantial speedups by exploiting the structure of K_y^{-1} and of the product space $\mathcal{M} \times \mathcal{M}$:

Sparsity. If K_y is sparse, or can be well-approximated by a sparse matrix, then K_y^{-1} is often also sparse (or well-approximated as sparse) in practice. This occurs in the case of compactly supported kernel functions (Gneiting, 2002; Rasmussen and Williams, 2006), but also even when using standard kernels with short lengthscales. Note that although there is no guarantee that the inverse of a sparse matrix must itself be sparse (with the exception of specific structures, e.g., block diagonal matrices), it is often the case that when K_y is sparse many entries of K_y^{-1} will be very near to zero, since points with negligible covariance generally also have negligibly small correlations in the precision matrix, so K_y^{-1} can often be well-approximated as sparse. When this is the case, our product tree need include only those pairs $(\mathbf{x}_p, \mathbf{x}_q)$ for which $(K_y^{-1})_{pq}$ is non-negligible. This is often a substantial advantage.

Symmetry. Since K_y^{-1} is a symmetric matrix, it is redundant to include leaves for both $(\mathbf{x}_p, \mathbf{x}_q)$ and $(\mathbf{x}_q, \mathbf{x}_p)$ in our tree. We can decompose $K_y^{-1} = U + D + U^T$, where $D = \mathrm{diag}(K_y^{-1})$ is a diagonal matrix and $U = \mathrm{triu}(K_y^{-1})$ is a strictly upper triangular (zero diagonal) matrix. This allows us to rewrite

$$\mathbf{k_i^*}^T K_y^{-1} \mathbf{k_j^*} = \mathbf{k_i^*}^T U \mathbf{k_j^*} + \mathbf{k_i^*}^T D \mathbf{k_j^*} + \mathbf{k_i^*}^T U^T \mathbf{k_j^*},$$

in which the first and third terms can be implemented as calls to WeightedMetricSum on a product tree built from U; note that this tree will be half the size of a tree built for K_y^{-1} since we omit zero entries. The second (diagonal) term can be computed using a separate (very small) product tree built from the nonzero entries of D. The accumulated sum $\hat{S}$ can be carried over between these three computations, so we can speed up the later computations by accumulating large weights in the earlier computations.

Factorization of product distances. In general, computing the product distance δ will usually involve two calls to the underlying distance metric d; these can often be reused. For example, when calculating both $\delta((\mathbf{x}_a, \mathbf{x}_b), (\mathbf{x}_c, \mathbf{x}_d))$ and $\delta((\mathbf{x}_a, \mathbf{x}_e), (\mathbf{x}_c, \mathbf{x}_d))$, we can reuse the value of $d(\mathbf{x}_a, \mathbf{x}_c)$ for both computations. This reduces the total number of calls to the distance function during tree construction from a worst-case n^4 (for all pairs of pairs of training points) to a maximum of n^2, and in general much fewer if other optimizations such as sparsity are implemented as well. This can dramatically speed up tree construction when the distance metric is slow to evaluate. It can also speed up test-time evaluation, if distances to the same point must be computed at multiple levels of the tree.

3.2 Other Kernel Functions

As noted above, the SE kernel has the lucky property that, if we choose product metric $\delta = \sqrt{d_1^2 + d_2^2}$, then the product of two SE kernels is equal to the kernel of

Kernel	$k(d)$	$k(d_1)k(d_2)$	$\delta(d_1,d_2)$	$k^{\text{prod}}_{\text{lower}}(\delta)$	$k^{\text{prod}}_{\text{upper}}(\delta)$
SE	$\exp\left(-d^2\right)$	$\exp\left(-d_1^2-d_2^2\right)$	$\sqrt{d_1^2+d_2^2}$	$\exp\left(-(\delta)^2\right)$	$\exp\left(-(\delta)^2\right)$
γ-exponential	$\exp\left(-d^\gamma\right)$	$\exp\left(-d_1^\gamma-d_2^\gamma\right)$	$\left(d_1^\gamma+d_2^\gamma\right)^{1/\gamma}$	$\exp\left(-(\delta)^\gamma\right)$	$\exp\left(-(\delta)^\gamma\right)$
Rational Quadratic	$\left(1+\frac{d^2}{2\alpha}\right)^{-\alpha}$	$\left(1+\frac{d_1^2+d_2^2}{2\alpha}+\frac{d_1^2 d_2^2}{4\alpha^2}\right)^{-\alpha}$	$\sqrt{d_1^2+d_2^2}$	$\left(1+\frac{(\delta)^2}{2\alpha}+\frac{(\delta)^4}{16\alpha^2}\right)^{-\alpha}$	$\left(1+\frac{(\delta)^2}{2\alpha}\right)^{-\alpha}$
Matérn ($\nu=3/2$)	$\left(1+\sqrt{3}d\right)$ $\cdot\exp\left(-\sqrt{3}d\right]\Big)$	$\left(1+\sqrt{3}\left(d_1+d_2\right)+3d_1d_2\right)$ $\cdot\exp\left(-\sqrt{3}(d_1+d_2)\right)$	d_1+d_2	$\left(1+\sqrt{3}\delta\right)$ $\cdot\exp\left(-\sqrt{3}\delta\right)$	$\left(1+\sqrt{3}\delta+3(\delta/2)^2\right)$ $\cdot\exp(-\sqrt{3}\delta)$
Piecewise polynomial (compact support), $q=1$, dimension D, $j=\left\lfloor\frac{D}{2}\right\rfloor+2$	$(1-d)_+^{j+1}$ $\cdot((j+1)d+1)$	$((1-d_1)_+(1-d_2)_+)^{j+1}$ $\cdot\left((j+1)^2d_1d_2\right.$ $\left.+\,(j+1)(d_1+d_2)+1\right)$	d_1+d_2	$(1-\delta)_+^{j+1}$ $\cdot((j+1)\delta+1)$	$\left(1-\delta+\frac{(\delta)^2}{4}\right)_+^{j+1}$ $\cdot\left((j+1)^2\left(\frac{\delta}{2}\right)^2+(j+1)\delta+1\right)$

Table 1: Bounds for products of common kernel functions. All kernel functions are from Rasmussen and Williams (2006).

the product metric δ:

$$k_{SE}(d_1)k_{SE}(d_2) = \exp\left(-d_1^2-d_2^2\right) = k_{SE}(\delta).$$

In general, however, we are not so lucky: it is not the case that every kernel we might wish to use has a corresponding product metric such that a product of kernels can be expressed in terms of the product metric. In such cases, we may resort to upper and lower bounds in place of computing the exact kernel value. Note that such bounds are all we require to evaluate the cutoff rule (6), and that when we reach a leaf node representing a specific pair of points we can always evaluate the exact product of kernels directly at that node. As an example, consider the Matérn kernel

$$k_{\text{M}}(d) = (1+\sqrt{3}d)\exp(-\sqrt{3}d)$$

(where we have taken $\nu = 3/2$); this kernel is popular in geophysics because its sample paths are once-differentiable, as opposed to infinitely smooth as with the SE kernel. Considering the product of two Matérn kernels,

$$k_{\text{M}}(d_1)k_{\text{M}}(d_2) = (1+\sqrt{3}(d_1+d_2)+3d_1d_2)\exp(-\sqrt{3}(d_1+d_2))$$

we notice that this is almost equivalent to $k_{\text{M}}(\delta)$ for the choice of $\delta = d_1+d_2$, but with an additional pairwise term of $3d_1d_2$. We bound this term by noting that it is maximized when $d_1 = d_2 = \delta/2$ and minimized whenever either $d_1 = 0$ or $d_2 = 0$, so we have $3(\delta/2)^2 \geq 3d_1d_2 \geq 0$. This yields the bounds $k^{\text{prod}}_{\text{lower}}$ and $k^{\text{prod}}_{\text{upper}}$ as shown in Table 1. Bounds for other common kernels are obtained analogously in Table 1.

4 Primal / Dual and Mixed GP Representations

In this section, we extend the product tree approach to models combining a long-scale parametric component with a short-scale nonparametric component. We introduce these models, which we refer to as *mixed primal/dual* GPs, and demonstrate how they can mediate between the desire to model long-scale structure and the need to maintain a short lengthscale for efficiency. (Although this class of models is well known, we have not seen this particular use case described in the literature). We then show that the necessary computations in these models can be done efficiently using the techniques described above.

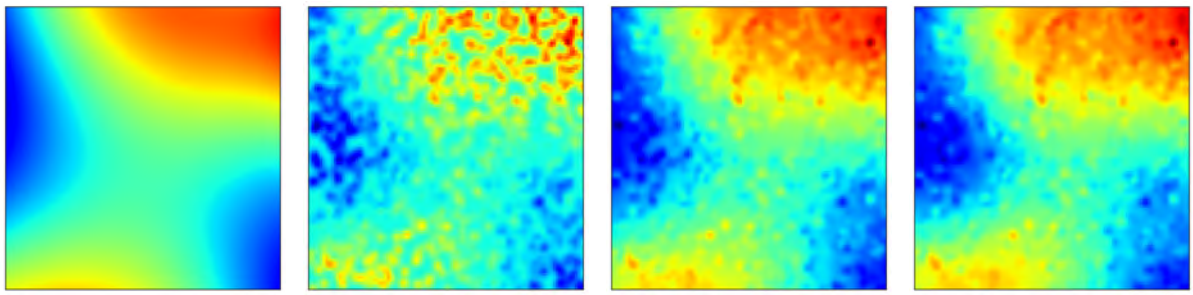
4.1 Mixed Primal/Dual GP Models

Although GP regression is commonly thought of as nonparametric, it is possible to implement parametric models within the GP framework. For example, a Bayesian linear regression model with Gaussian prior,

$$y = \mathbf{x}^T\beta+\epsilon,\ \ \beta\sim\ \mathcal{N}(\mathbf{0},I),\ \ \epsilon\sim\mathcal{N}(0,\sigma_n^2),$$

is equivalent to GP regression with a linear kernel $k(\mathbf{x},\mathbf{x}') = \langle\mathbf{x},\mathbf{x}'\rangle$, in the sense that both models yield the same (Gaussian) predictive distributions (Rasmussen and Williams, 2006). However, the two representations have very different *computational* properties: the primal (parametric) representation allows computation of the predictive mean and variance in $O(D)$ and $O(D^2)$ time respectively, where D is the input dimensionality, while the dual (nonparametric) representation requires time $O(n)$ and $O(n^2)$ respectively for the same calculations. When learning simple models on large, low-dimensional (e.g., spatial) data sets, the primal representation is obviously more attractive, since we can store and compute with model parameters directly, in constant time relative to n.

Of course, simple parametric models by themselves cannot capture the complex local structure that often appears in real-world datasets. Fortunately it is possible to combine a parametric model with a nonparametric GP model in a way that retains the advantages of both approaches. To define a combined model, we replace the standard zero-mean GP assumption with a



(a) *MAD*= 1.11
polynomial(5)

(b) *MAD*= 1.77
GP w/ $\ell = 0.02$

(c) *MAD*= 0.73
Mixed:
polynomial(5) +
GP w/ $\ell = 0.02$

(d) *MAD*= 0.65
True GP w/
$\ell_1 = 0.5$
$\ell_2 = 0.02$

Figure 3: A primal/dual mixture approximating a longer-scale GP.

parametric mean function $\mathbf{h}(\mathbf{x})^T\beta$, yielding the model

$$y = f(\mathbf{x}) + \mathbf{h}(\mathbf{x})^T\beta + \epsilon$$

where $\mathbf{h}(\mathbf{x})$ is a vector of feature values $[h_1(\mathbf{x}), \ldots, h_D(\mathbf{x})]$. The GP model is then learned jointly along with a posterior distribution on the coefficients β. Assuming a Gaussian prior $\beta \sim \mathcal{N}(b, B)$ on the coefficients, the predictive distribution $g(X^*) \sim \mathcal{N}(\mu'_*, \Sigma'_*)$ can be derived (Rasmussen and Williams, 2006) as

$$\mu'_* = H^{*T}\bar{\beta} + K^{*T}K_y^{-1}(\mathbf{y} - H^{*T}\bar{\beta}) \quad (7)$$

$$\Sigma'_* = K^{**} - K^{*T}K_y^{-1}K^* + R^T(B^{-1} + HK_y^{-1}H^T)R \quad (8)$$

where we define $H_{ij} = h_j(\mathbf{x}_i)$ for each training point x_i, similarly H^* for the test points, and we have $\bar{\beta} = (B^{-1} + HK_y^{-1}H^T)^{-1}(HK_y^{-1}\mathbf{y} + B^{-1}\mathbf{b})$ and $R = H^* - HK_y^{-1}K^*$. Section 2.7 of Rasmussen and Williams (2006) gives further details.

Note that linear regression in this framework corresponds to a choice of basis functions $h_1(x) = 1$ and $h_2(x) = x$; it is straightforward to extend this to polynomial regression and other models that are linear in their parameters. In general, any kernel which maps to a finite-dimensional feature space can be represented parametrically in that feature space, so this framework can efficiently handle kernels of the form $k(\mathbf{x}, \mathbf{x}') = \sum_i k_i(\mathbf{x}, \mathbf{x}') + k_S(\mathbf{x}, \mathbf{x}')$, where k_S is a short-lengthscale or compactly supported kernel, monotonically decreasing w.r.t. some distance metric as assumed above, and each k_i either has an exact finite-dimensional feature map or can be approximated using finite-dimensional features Rahimi and Recht (2007); Vedaldi and Zisserman (2010).

As an example, Figure 3 compares several approaches for inferring a function from a GP with long and short-lengthscale components. We drew training data from a GP with a mixture of two SE kernels at lengthscales $\ell_1 = 0.5$ and $\ell_2 = 0.02$, sampled at 1000 random points in the unit square. Figure 3 displays the posterior means of four models on a 100 by 100 point grid, reporting the mean absolute deviation (MAD) of the model predictions relative to the "true" values (drawn from the same GP) at 500 random test points. Note that although the short-scale GP (3b) cannot by itself represent the variation from the longer-scale kernel, when combined with a parametric polynomial component (3a) the resulting mixed model (3c) achieves accuracy approaching that of the true model (3d).

4.2 Efficient Operations in Primal/Dual Models

Likelihood calculation in primal/dual models is a straightforward extension of the standard case. The predictive mean (7) can be accommodated within the framework of Shen et al. (2006) using a tree representation of the vector $K_y^{-1}\left(\mathbf{y} - H^{*T}\bar{\beta}\right)$, then adding in the easily evaluated parametric component $H^{*T}\bar{\beta}$. In the covariance (8) we can use a product tree to approximate $K^{*T}K_y^{-1}K^*$ as described above; of the remaining terms, $\bar{\beta}$ and $B^{-1} + HK_y^{-1}H^T$ can be precomputed at training time, and H^* and K^{**} don't depend on the training set. This leaves $HK_y^{-1}K^*$ as the one remaining challenge; we note that this quantity can be computed efficiently using mD applications of the vector multiplication primitive from Shen et al. (2006), re-using the same tree structure to multiply each column of K^* by each row of HK_y^{-1}. Thus, all of the the operations required for likelihood computation can be implemented efficiently with no explicit dependence on n (i.e., with no direct access to the training set except through space-partitioning tree structures).

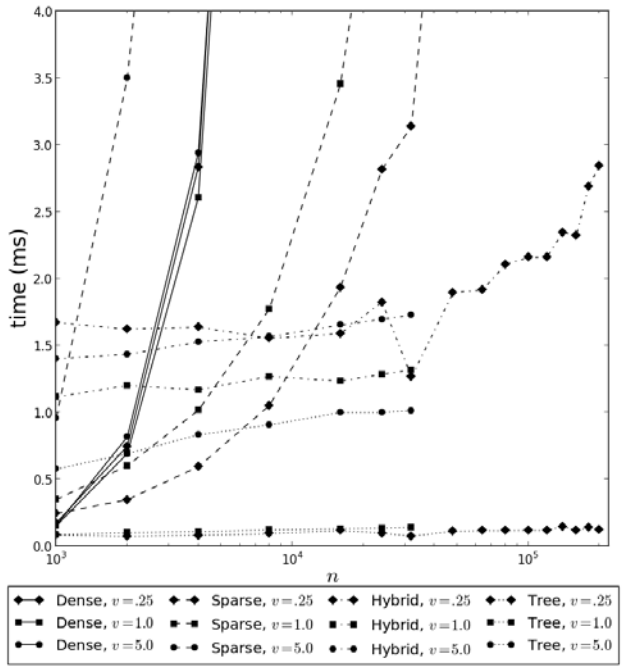


Figure 4: Mean runtimes for dense, sparse, hybrid, and product tree calculation of GP variance on a 2D synthetic dataset.

5 Evaluation

We compare calculation of the predictive variance using a product tree to several other approaches: a naïve implementation using dense matrices, a direct calculation using a sparse representation of K_y^{-1} and dense representation of $\mathbf{k}_\mathbf{i}^*$, and a hybrid tree implementation that attempts to also construct a sparse $\mathbf{k}_\mathbf{i}^*$ by querying a cover tree for all training points within distance r of the query point $\mathbf{x}_\mathbf{i}^*$, where r is chosen such that $k(r')$ is negligible for $r' > r$, and then filling in only those entries of $\mathbf{k}_\mathbf{i}^*$ determined to be non-negligible.

Our product tree implementation is a Python extension written in C++, based on the cover tree implementation of Beygelzimer et al. (2006) and implementing the optimizations from section 3.1. The approximation parameters ϵ_{rel} and ϵ_{abs} were set appropriately for each experiment so as to ensure that the mean approximation error is less than 0.1% of the exact variance. All sparse matrix multiplications are in CSR format using SciPy's sparse routines; we impose a sparsity threshold of 10^{-8} such that any entry less than the threshold is set to zero.

Figure 4 compares performance of these approaches on a simple two-dimensional synthetic data set, consisting of points sampled uniformly at random from the unit square. We train a GP on n such points and then measure the average time per point to compute the predictive variance at 1000 random test points. The GP uses an SE kernel with observation noise $\sigma_n^2 = 0.1$ and lengthscale $\ell = \sqrt{v\pi/n}$, where v is a parameter indicating the average number of training points within a one-lengthscale ball of a random query point (thus, on average there will be $4v$ points within two lengthscales, $9v$ within three lengthscales, etc.).

The results of Figure 4 show a significant advantage for the tree-based approaches, which are able to take advantage of the geometric sparsity structure in the training data. The dense implementation is relatively fast on small data sets but quickly blows up, while the sparse calculation holds on longer (except in the relatively dense $v = 5.0$ setting) but soon succumbs to linear growth, since it must evaluate the kernel between the test point and each training point. The hybrid approach has higher overhead but scales very efficiently until about $n = 48000$, where the sparse matrix multiplication's $\Omega(n)$ runtime (Bank and Douglas, 1993) begins to dominate. Conversely, the product tree remains efficient even for very large, sparse datasets, with $v = 0.25$ runtimes growing from 0.08ms at $n = 1000$ to just 0.13ms at $n = 200000$. Due to memory limitations we were unable to evaluate $v = 1.0$ and $v = 5.0$ for values of n greater than 32000.

Our second experiment uses amplitude data from 3105 seismic events (earthquakes) detected by a station in Fitzroy Crossing, Australia; the event locations are shown in Figure 1. The amplitudes are normalized for event magnitude, and the task is to predict the recorded amplitude of a new event given that event's latitude, longitude, and depth. Here our distance metric is the great-circle distance, and we expect our data to contain both global trends and local structure, since events further away from the detecting station will generally have lower amplitudes, but this may vary locally as signals from a given source region generally travel along the same paths through the earth and are dampened or amplified in the same ways as they travel to the detecting station.

Table 2 considers several models for this data. A simple parametric model, the fifth-order polynomial in event-to-station distance shown in Figure 5, is not very accurate but does allow for very fast variance evaluations. The GP models are more accurate, but the most accurate GP model uses a relatively long lengthscale of 50km, with correspondingly slow variance calculations. Depending on application requirements, the most appealing tradeoff might be given by the mixed model combining a fifth-degree polynomial with a 10km SE GP: this model achieves accuracy close to that of the 50km models, but with significantly faster variance calculations due to the shorter lengthscale, especially when using a product tree.

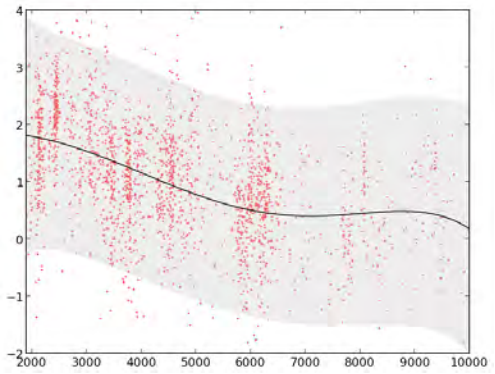


Figure 5: Normalized amplitude as a function of event-station distance, with a fifth-degree polynomial fit shading ±2std.

Model	Error	Sparse (ms)	Tree (ms)
Polynomial in distance (deg 5)	0.78	0.050	n/a
GP, SE, $\ell = 10$km	0.67	0.722 ± 0.032	0.216 ± 0.224
poly/GP, deg 5,SE, 10km	0.62	0.795 ± 0.033	0.413 ± 0.307
GP, Matérn, $\ell = 10$km	0.65	1.256 ± 0.592	0.337 ± 0.365
poly/GP, deg 5, Matérn, 10km	0.62	1.327 ± 0.602	0.654 ± 0.499
GP, SE, $\ell = 50$km	0.61	1.399 ± 0.661	1.168 ± 1.242
poly/GP, deg 5, SE, 50km	0.60	1.490 ± .677	1.551 ± 1.409

Table 2: Models for Fitzroy Crossing amplitude data, with mean absolute prediction error from five-fold cross validation and (mean ± std) time to compute the predictive variance via a direct sparse calculation versus a product tree.

6 Related Work

Previous approximations for GP mean prediction (Moore et al., 1997; Shen et al., 2006; Gray, 2004), which inspired this work, use tree structures to implement an efficient matrix-vector multiplication (MVM); the Improved Fast Gauss Transform (Morariu et al., 2008) also implements fast MVM for the special case of the SE kernel. It is possible to accelerate GP training by combining MVM methods with a conjugate gradient solver, but models thus trained do not allow for the computation of predictive variances. One argument against MVM techniques (and, by extension, our product tree approach) is that their efficiency requires shorter lengthscales than are common in machine learning applications (Murray, 2009); however, we have found them quite effective on datasets which do have genuinely sparse covariance structure (e.g., geospatial data), or in which the longer-scale variation can be represented by a parametric component.

Another set of approaches to speeding up GP regression, sparse approximations (Csató and Opper, 2002; Seeger et al., 2003; Snelson and Ghahramani, 2006; Quiñonero-Candela and Rasmussen, 2005), attempt to represent n training points using a smaller set of m points, allowing training in $O(nm^2)$ time and predictive covariance (thus likelihood) computation in $O(m^2)$ time. This is philosophically a different approach from that of this paper, where we generally want to retain all of our training points in order to represent local structure. However, there is no formal incompatibility: many sparse approaches, including all of those discussed by Quiñonero-Candela and Rasmussen (2005), yield predictive covariances of the form $\mathbf{k}_\mathbf{i}^{*T} Q \mathbf{k}_\mathbf{j}^*$ for some matrix Q (or a sum of terms of this form), where this product could be computed straightforwardly using a product tree. Several non-sparse approximations, e.g., the Nyström approximation (Williams and Seeger, 2001), also yield predictive covariances of this form.

More closely related to our setting are local approximations, in which different GPs are trained in different regions of the input space. There is some evidence that these can provide accurate predictions which are very fast to evaluate (Chalupka et al., 2013); however, they face boundary discontinuities and inaccurate uncertainty estimates if the data do not naturally form independent clusters. Since training multiple local GPs is equivalent to training a single global GP with a block diagonal covariance matrix, it should be possible to enhance local GPs with global parametric components as in section 4, similarly to the combined local/global approximation of Snelson and Ghahramani (2007).

7 Conclusion and Future Work

We introduce the *product tree* structure for efficient adaptive calculation of GP covariances using a multiresolution clustering of pairs of training points. Specific contributions of this paper include product metrics and bounds for common kernels, the adaptation to metric trees, a novel cutoff rule incorporating both the weights and the quantity being summed over, and covariance-specific performance optimizations. Additionally, we describe efficient calculation in GP models incorporating both primal and dual components, and show how such models can model global-scale variation while maintaining the efficiency of short-lengthscale GPs.

A limitation of our approach is the need to explicitly invert the kernel matrix during training; this can be quite difficult for large problems. One avenue for future work could be an iterative factorization of K_y

analogous to the CG training performed by MVM methods (Shen et al., 2006; Gray, 2004; Morariu et al., 2008). Another topic would be a better understanding of cutoff rules for the weighted sum recursion, e.g., an empirical investigation of different rules or a theoretical analysis bounding the error and/or runtime of the overall computation.

Finally, although our work has been focused primarily on low-dimensional applications, the use of cover trees instead of k-d trees ought to enable an extension to higher dimensions. We are not aware of previous work applying tree-based regression algorithms to high-dimensional data, but as high-dimensional covariance matrices are often sparse, this may be a natural fit. For high-dimensional data that do not lie on a low-dimensional manifold, other nearest-neighbor techniques such as locality-sensitive hashing (Andoni and Indyk, 2008) may have superior properties to tree structures; the adaptation of such techniques to GP regression is an interesting open problem.

References

Andoni, A. and Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122.

Bank, R. E. and Douglas, C. C. (1993). Sparse matrix multiplication package (SMMP). *Advances in Computational Mathematics*, 1(1):127–137.

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 97–104.

Chalupka, K., Williams, C. K., and Murray, I. (2013). A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14:333–350.

Csató, L. and Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, 14(3):641–668.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226.

Gneiting, T. (2002). Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83(2):493–508.

Gray, A. (2004). Fast kernel matrix-vector multiplication with application to Gaussian process learning. Technical Report CMU-CS-04-110, School of Computer Science, Carnegie Mellon University.

Moore, A. W., Schneider, J., and Deng, K. (1997). Efficient locally weighted polynomial regression predictions. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*.

Morariu, V., Srinivasan, B. V., Raykar, V. C., Duraiswami, R., and Davis, L. (2008). Automatic online tuning for fast Gaussian summation. *Advances in Neural Information Processing Systems (NIPS)*, 21:1113–1120.

Murray, I. (2009). Gaussian processes and fast matrix-vector multiplies. In *Numerical Mathematics in Machine Learning workshop at the 26th International Conference on Machine Learning (ICML 2009)*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems (NIPS)*, 20:1177–1184.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Seeger, M., Williams, C. K., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics (AISTATS)*, volume 9.

Shen, Y., Ng, A., and Seeger, M. (2006). Fast Gaussian process regression using kd-trees. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, page 1225.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. In *Artificial Intelligence and Statistics (AISTATS)*, volume 11.

Uhlmann, J. K. (1991). Satisfying general proximity / similarity queries with metric trees. *Information Processing Letters*, 40(4):175 – 179.

Vanhatalo, J. and Vehtari, A. (2008). Modelling local and global phenomena with sparse Gaussian processes. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*.

Vedaldi, A. and Zisserman, A. (2010). Efficient additive kernels via explicit feature maps. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3539–3546. IEEE.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*. Citeseer.

Latent Topic Analysis for Predicting Group Purchasing Behavior on the Social Web

Feng-Tso Sun, Martin Griss, and Ole Mengshoel
Electrical and Computer Engineering Department
Carnegie Mellon University

Yi-Ting Yeh
Computer Science Department
Stanford University

Abstract

Group-deal websites, where customers purchase products or services in groups, are an interesting phenomenon on the Web. Each purchase is kicked off by a group initiator, and other customers can join in. Customers form communities with people with similar interests and preferences (as in a social network), and this drives bulk purchasing (similar to online stores, but in larger quantities per order, thus customers get a better deal). In this work, we aim to better understand what factors influence customers' purchasing behavior for such social group-deal websites. We propose two probabilistic graphical models, i.e., a product-centric inference model (PCIM) and a group-initiator-centric inference model (GICIM), based on Latent Dirichlet Allocation (LDA). Instead of merely using customers' own purchase history to predict purchasing decisions, these two models include other social factors. Using a lift curve analysis, we show that by including social factors in the inference models, PCIM achieves 35% of the target customers within 5% of the total number of customers while GICIM is able to reach 85% of the target customers. Both PCIM and GICIM outperform random guessing and models that do not take social factors into account.

1 Introduction

Group purchasing is a business model that offers various deals-of-the-day and an extra discount depending on the size of the purchasing group. After group-deal websites, such as Groupon and LivingSocial, have gained attention, similar websites, such as **ihergo**[1] and **Taobao**,[2] have introduced social networks as a feature for their users. These group-deal websites provide an interesting hybrid of social networks (e.g., Facebook.com and LinkedIn.com) and online stores (e.g., Amazon.com and Buy.com). Customers form communities with people with similar interests and preferences (as in a social network), and this drives bulk purchasing (similar to online stores, but in larger quantities per order, thus customers get a better deal). As we see more and more social interactions among customers in group-deal websites, it is critical to understand the interplay between social factors and purchasing preferences.

In this paper, we analyze a transactional dataset from the largest social group-deal website in Taiwan, ihergo.com. Figure 1 shows a screenshot from the group-deal page of ihergo.com. Each group-purchasing event on ihergo.com consists of three major components: (1) a group initiator, (2) a number of group members, and (3) a group-deal product. A group initiator starts a group-purchasing event for a specific group-deal product. While this event will be posted publicly, the group initiator's friends will also be notified. A user can choose to join the purchasing event to become a group member.

Group initiators play important roles on this kind of group-deal websites. Usually, the merchants would offer incentives for the group initiators to initiate group-purchasing events by giving them products for free if the size of the group exceeds some threshold. In addition, to save shipping costs, the group can choose to have the whole group-deal order shipped to the initiator. In this case, the initiator would need to distribute the products to group members in person. Hence, the group members usually reside or work in the proximity of the group initiator. Sometimes, they are friends or co-workers of the initiator.

Understanding customers' purchasing behavior in this

[1] http://www.ihergo.com

[2] http://www.taobao.com

kind of social group-purchasing scenario could help group-deal websites strategically design their offerings. Traditionally, customers search for or browse products of their interests on websites like Amazon.com. However, on social group-deal websites, customers can perform not only product search, but they can also browse group deals and search for initiators by ratings and locations. Therefore, a good recommender system [1] for social group-deal websites should take this into account. If the website can predict which customers are more likely to join a group-purchasing event started by a specific initiator, it can maximize group sizes and merchants' profits in a shorter period of time by delivering targeted advertising. For example, instead of spamming everyone, the website can send out notifications or coupons to the users who are most likely to join the group-purchasing events.

In this work, we aim to predict potential customers who are most likely to join a group-purchasing event. We apply Latent Dirichlet Allocation (LDA) [2] to capture customers' purchasing preferences, and evaluate our proposed predictive models based on a one-year group-purchasing dataset from ihergo.com.

Our contributions in understanding the importance of social factors for group-deal customers' decisions are the following:

- **A new type of group-purchasing dataset.** We introduce and analyze a new type of group-purchasing dataset, which consists of 5,602 users, 26,619 products and 13,609 group-purchasing events.
- **Predictive models for group-deal customers.** Based on topic models, we propose two predictive models that include social factor. They achieve higher prediction accuracy compared to the baseline models.

In the next section, we describe related work in the area of group purchasing behavior, social recommendations, and topic models for customer preferences. Section 3 introduces and analyzes the characteristics of our real-world group-purchasing dataset. In Section 4, we first review LDA, then present two proposed predictive models for group-deal customer prediction. Experimental results are given in Section 5. Finally, conclusion and future research direction are presented in Section 6.

2 Related Work

In this section, we review related work in three areas: (1) group purchasing behavior, (2) social recommendations, and (3) topic models for customer preferences.

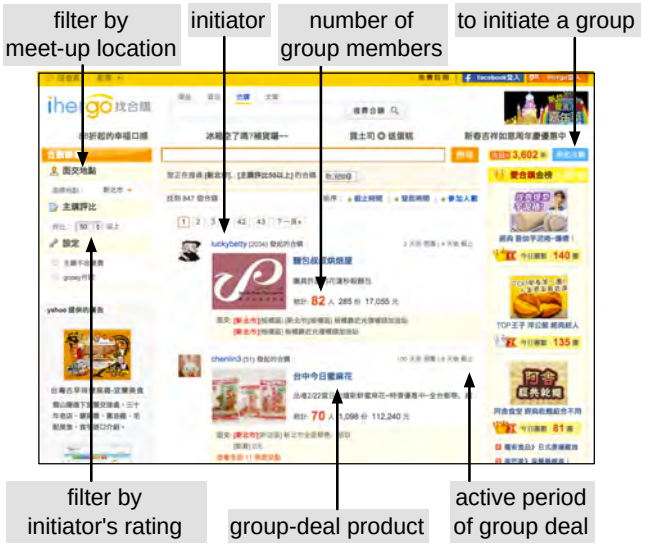


Figure 1: Screenshot of the group-deal page from ihergo.com.

Group Purchasing Behavior. Since group-deal websites such as Groupon and LivingSocial gained attention, several studies have been conducted to understand factors influencing group purchasing behavior. Byers et al. analyzed purchase histories of Groupon and LivingSocial [3]. They showed that Groupon optimizes deal offers strategically by giving "soft" incentives, such as deal scheduling and duration, to encourage purchases. Byers et al. also compared Groupon and LivingSocial sales with additional datasets from Yelp's reviews and Facebook's like counts [4]. They showed that group-deal sites benefit significantly from word-of-mouth effects on users' reviews during sales events. Edelman et al. studied the benefits and drawbacks of using Groupon from the point of view of the merchants [6]. Their work modeled whether advertising and price discrimination effects can make discounts profitable. Ye et al. introduced a predictive dynamic model for group purchasing behavior. This model incorporates social propagation effects to predict the popularity of group deals as a function of time [19]. In this work, we focus on potential customer prediction, as opposed to modeling the overall deal purchasing sales over time.

Social Recommendations. In real life, a customer's purchasing decision is influenced by his or her social ties. Guo et al. analyzed the dataset from the largest Chinese e-commerce website, Taobao, to study the relationship between information passed among buyers and purchasing decision [7]. Leskovec et al. used a stochastic model to explain the propagation of recommendations and cascade sizes [11]. They showed

that social factors have a different level of impact on user purchasing decision for different products. Moreover, previous work also tried to incorporate social information into existing recommendation techniques, such as collaborative filtering [13, 14, 20, 12]. Recently, many recommendation systems have been implemented, taking advantage of social network information in addition to users' preferences to improve recommendation accuracy. For example, Yang et al. proposed a Bayesian-inference based movie recommendation system for online social networks [18]. Our work considers the relationship between the group initiator and the group members as a social tie to augment customer prediction for group-purchasing events.

Topic Models for Customer Preference. Topic models such as LDA have been widely and successfully used in many applications including language modeling [2], text mining [17], human behavior modeling [9], social network analysis [5], and collaborative filtering [8]. Researchers have also proposed new topic models for purchasing behavior modeling. For example, topic models have been extended with price information to analyze purchase data [10]. By estimating the mean and the variance of the price for each product, the proposed model can cluster related items by taking their price ranges into account. Iwata and Watanabe proposed a topic model for tracking time-varying consumer purchase behavior, in which consumer interests and item trends change over time [9]. In this paper, we use LDA to learn topic proportions from purchase history to represent customers' purchasing preferences.

3 Group-Purchasing Dataset

The dataset for our data analysis comes from users' transactional data of a group-deal website, ihergo. It is the largest social group-deal website in Taiwan. We collected longitudinal data between October 1st 2011 and October 1st 2012. From the users' geographical profile, we are able to group them based on their living area. For this study, we include all 5,602 users living in Taipei, the capital of Taiwan. In total, our dataset contains 26,619 products and 13,609 group-purchasing events.

On ihergo, users can purchase a product by joining a group-purchasing event. There are two roles among the users: 1) the **group initiator** and 2) the **group member**. A group initiator initiates a purchase group which other users can join to become group members. Once the group size exceeds some threshold, the group members can get a discount on the product while the initiator can get the product for free. Sometimes the group initiator and the group members already know each other before they join the same group-purchasing event. Sometimes they become friends after the event. Moreover, each user can become a follower of a group initiator. When a new group-purchasing event is initiated by an initiator, the system will notify his or her followers.

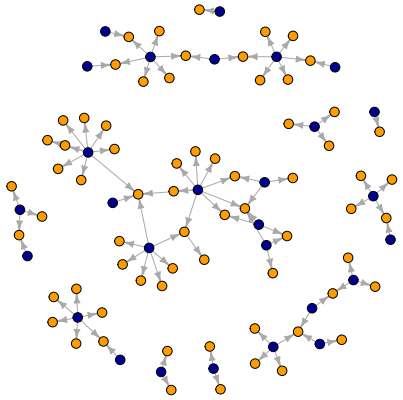


Figure 2: Part of group deal graph for ihergo dataset: illustration of the member-centric relationships between group members and initiators from a subset of randomly sampled joined group members. A directed edge is from a joined customer (dark blue) to an initiator (light orange).

Each group-purchasing deal in our dataset is composed of a set of attributes: the product description (e.g., discounted price, limited quantity, and product category), the group size, the group initiator, the group members, and the time period in which the deal is active. Group-purchasing deals are defined by a time series $\boldsymbol{D} = (D_1, D_2, ..., D_n)$, where D_i is a tuple $(t, p, o, \mathbf{m})$ denoting that a group-purchasing deal for product p is initiated by an organizer (initiator) o with joined group members $\mathbf{m} = \{m_1, ..., m_k\}$ at time t.

We represent group-purchasing events as a directed graph. Each user is a vertex in the graph. For every group-purchasing deal, we build directed edges from each group member to the initiator. There are 5,602 vertices and 16,749 edges in our ihergo dataset. The directed edges are defined by $E = \cup_{i\in[1,n]} \cup_{j\in[1,d(i)]} (m_{i,j}, o_i)$, where $d(i)$ is the number of joined customers for group deal i. The vertices in the graph are defined by $V = M \cup O$, where M denotes all group members $M = \mathbf{m_1} \cup \ldots \cup \mathbf{m_n}$ and O denotes total group-purchasing organizers (initiators) $O = \{o_1\} \cup \ldots \cup \{o_n\}$.

Figure 2 illustrates the joined customer centric graph structure by showing the relationships among a subset of randomly sampled joined customers. Light orange and dark blue vertices represent the group initiators and group members, respectively. According to this dataset, each user has joined 84 group-purchasing

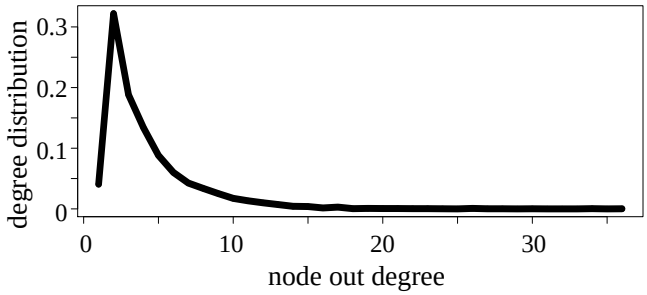


Figure 3: Node out-degree distribution for the group-purchasing graph of ihergo. 80% of the users follow five or fewer initiators.

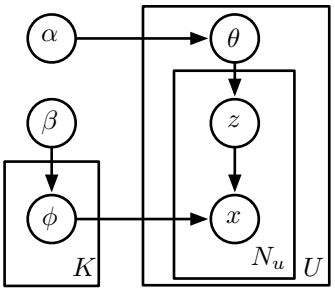


Figure 4: Graphical model representation of the latent Dirichlet allocation model.

events on average. However, one interesting observation from the graph is that the number of outgoing edges from dark blue vertices is far less than 84. This property can be even clearly seen from the out-degree distribution for the overall group-purchasing graph shown in Figure 3. We see that 80% of the users only join group-purchasing events initiated by 5 or fewer different initiators. Group members have a tendency to repeatedly join group-purchasing initiated by a relatively small number of initiators they co-bought with before.

Therefore, we hypothesize that customers' purchasing decisions are not only influenced by their own purchasing preferences but also strongly influenced by who the group initiator is. In the next section, we propose two new models to predict which customers are most likely to join a particular group-purchasing event.

4 Methodology

In this section, we first describe in Section 4.1 how we apply topic modeling to learn user purchasing preferences under the group-purchasing scenario. During the training phase, we compute for each user a mixture topic proportion by combining topic proportions of this user and the initiators with whom this user has co-bought products.

Given a new group-purchasing event, we would like to predict which customers are more likely to join. We propose two predictive models in Section 4.2. One model, which we denote as the product-centric inference model (PCIM), computes the posterior probability that a user would purchase this product given his or her mixture topic proportion. The other model, which we denote the group initiator centric inference model (GICIM), computes the posterior probability that a user would join the group-purchasing event initiated by this initiator given user's or initiator's mixture topic proportion.

4.1 Topic Model for User Purchasing Preference

We use topic modeling to characterize a user's purchasing preference. In particular, we apply LDA to our group-purchasing dataset. In a typical LDA model for text mining [2], a document is a mixture of a number of hidden topics which can be represented by a multinomial distribution, i.e. the topic proportion. A word can belong to one or more hidden topics with different probabilities. Figure 4 shows the graphical model for LDA. LDA is a generative model where each word in a document is generated by two steps: 1) sample a topic from its topic distribution and 2) draw a word from that topic. One can also use Bayesian inference to learn the particular topic proportion of each document.

In our model, we treat a user's purchase history as a document. Each purchased product can be seen as a word in a document. We make an analogy between text documents and purchasing patterns as shown in Table 1. We replace words with each purchased product and a document is one user's purchasing history. Assume that there are U users in our training data. Let $\mathbf{U}$ denote the set of users. Each user $u \in \mathbf{U}$ has a vector of purchased products $\boldsymbol{x}_u = \{x_{un}\}_{n=1}^{N_u}$ where N_u is the number of products that user u purchased.

The generative process of the LDA model for learning a user's purchasing preferences is described as following. Each user u has his or her own topic proportion (i.e., purchasing preference) $\theta_{\mathbf{u}}$ that is sampled from a Dirichlet distribution. Next, for each product x_{un} purchased by user u, a topic z_{un} is firstly chosen from the user's topic proportion θ_u. Then, a product x_{un} is drawn from the multinomial distribution $\phi_{z_{un}}$. To estimate θ_u and $\phi_{z_{un}}$, we use the collapsed Gibbs sampling method [15].

Symbol	Description for Group Purchase History	Description for Text Documents
U	Number of users	Number of documents
K	Number of latent topics	Number of latent topics
N_u	Number of purchased products of user u	Number of words of document u
z_{un}	Latent co-purchasing category of nth product	Latent topic of nth word of document u
x_{un}	nth purchased product of user u	nth word of document u
θ_u	Latent co-purchasing category proportion for user u	Topic proportin for document u
ϕ_k	Multinomial distribution over products for topic k	Mult. distribution over words for topic k
α	Dirichlet prior parameters for all θ_u	Dirichlet prior parameters for all θ_u
β	Dirichlet prior parameters for all ϕ_k	Dirichlet prior parameters for all ϕ_k

Table 1: Latent Dirichlet allocation plate model notation

4.2 Proposed Models for Predicting Group-Deal Customers

A group-purchasing event contains two kinds of critical information: who the group initiator is and what the group-deal product is. Our goal is to predict which customers are more likely to join a specific group-purchasing event. Intuitively, one may think that whether a customer would join a group-purchasing event solely depends on what the group-deal product is. However, from our observations in the dataset, we hypothesize a correlation between a customer's purchasing decision and who the group initiator is. Therefore, we would like to study how these two kinds of group-purchasing information affect the prediction accuracy by asking two questions:

1. What is the likelihood that a customer would join the event given what the *group-deal product* is?
2. What is the likelihood that a customer would join the event given who the *group-initiator* is?

This leads to our two proposed predictive models, the product centric inference model (*PCIM*) and the group initiator centric inference model (*GICIM*).

4.2.1 Product Centric Inference Model (PCIM)

Figure 5(a) shows the graphical structure of PCIM. For each user, we train a PCIM. PCIM computes the posterior probability that a user would purchase a product given his or her mixture topic proportion. Let C denote the *user's own topic proportion*, which we learned from LDA. Suppose that this user has joined group-purchasing events initiated by n group initiators, we use $\{I_i\}_{i=1}^n$ to denote the *learned topic proportions of these initiators*. Our model computes the weighted topic proportions of initiators W by linearly combining $\{I_i\}_{i=1}^n$ with the frequency distribution that the user co-bought products with them.

Intuitively, if a user joins a group-purchasing event initiated by a group initiator, they might share similar interests. Therefore, our model characterizes the user's purchasing preferences by a weighting scheme that combines C and W with a weighting parameter w. We use M to denote such a *mixture topic proportion* which encodes the overall purchasing preferences of the user.

Let P denote the *product random variable*. $\Omega(\mathrm{P}) = \{p_1, ..., p_m\}$, where p_i is the product. From each data record $D_i \in \boldsymbol{D}$, we have a tuple (t_i, p_i, o_i, $\mathbf{m_i}$) and know what group-deal product p_i corresponds to a particular group-purchasing event e_i. Our goal is to compute $Pr(P = p_i)$, the probability that the user would join a group-purchasing event e_i to buy a product p_i.

Given the topic proportion C and $\{I_i\}_{i=1}^n$ corresponding to the user and the weighting parameter w, we are able to compute

$$Pr(P) = \sum_{\mathbf{Y}=\mathbf{X}_p \setminus \{P\}} Pr(P, \mathbf{Y}) \tag{1}$$

where $\mathbf{X}_p = \{P, M, C, W, I_1, ..., I_n\}$; P a is product random variable; M is a mixture topic proportion.

To predict which users are more likely to join a group-purchasing event, we rank $\{\mathcal{P}_{p_i}^{(u_1)}, \ldots, \mathcal{P}_{p_i}^{(u_U)}\}$ in descending order where $\{u_1, \ldots, u_U\}$ denotes the set of users in our dataset and $\mathcal{P}_{p_i}^{(u_j)}$ denotes $Pr(P = p_i)$ of user u_j.

4.2.2 Group Initiator Centric Inference Model (GICIM)

The graphical illustration of GICIM is shown in Figure 5(b). GICIM computes the posterior probability that a user would join a group-purchasing event initiated by a particular initiator given user's or initiator's mixture topic proportion. GICIM and PCIM only differ in their leaf nodes. While PCIM considers only what the group-deal product is, GICIM models our observation that the decision of whether or not a user joins a

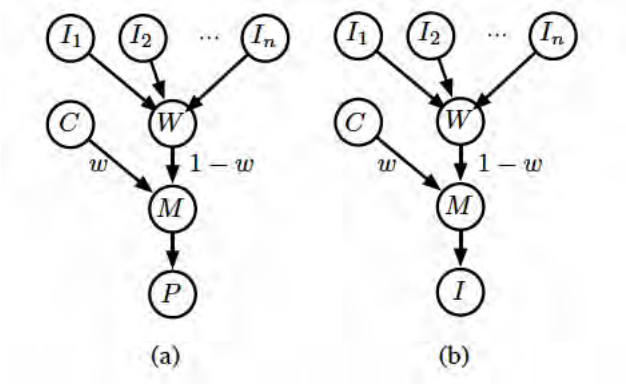


Figure 5: Our proposed models for predicting potential customers given a group-purchasing event. (a) Product centric inference model (PCIM). (b) Group initiator centric inference model (GICIM).

group-purchasing event is strongly influenced by who the group initiator of that event is.

Let I denote the initiator random variable and i_i denote the initiator of the group-purchasing event e_i. Again, from each data record $D_i \in \boldsymbol{D}$, we know who the group initiator is. Instead of evaluating $Pr(P = p_i)$ as in PCIM, we use GICIM to compute

$$Pr(I) = \sum_{\mathbf{Y}=\mathbf{X}_p \setminus \{I\}} Pr(I, \mathbf{Y}) \quad (2)$$

where $\mathbf{X}_p = \{I, M, C, W, I_1, ..., I_n\}$; I is an initiator random variable; M is a mixture topic proportion.

To predict which users are more likely to join a group-purchasing event e_i, we rank $\{\mathcal{P}_{o_i}^{(u_1)}, \ldots, \mathcal{P}_{o_i}^{(u_U)}\}$ in descending order where $\{u_1, \ldots, u_U\}$ denotes the set of users in our dataset and $\mathcal{P}_{o_i}^{(u_j)}$ denotes $Pr(I = o_i)$ of user u_j.

5 Experimental Evaluation

5.1 Data Pre-processing

We evaluate the proposed PCIM and GICIM models with the ihergo group-purchasing dataset. In order to capture meaningful user purchasing preferences, we remove users who purchased fewer than 10 products during the pre-processing step. We use ten-fold cross-validation to generate our training and testing datasets.

5.2 LDA Topic Modeling on Group Purchasing Dataset

To measure the performance of LDA for different number of topics (20, 40, 60, 80, 100) in our group-purchasing dataset, we compute the perplexity. It measures how well the model generalizes and predicts new documents [2]. Figure 6 shows that the perplexity decreases as the number of iteration increases and converges within 200 iterations. In addition, as we increase the number of topics, the perplexity decreases. Unless mentioned specifically, all topic proportions used in our experiments are learned with LDA using 100 topics.

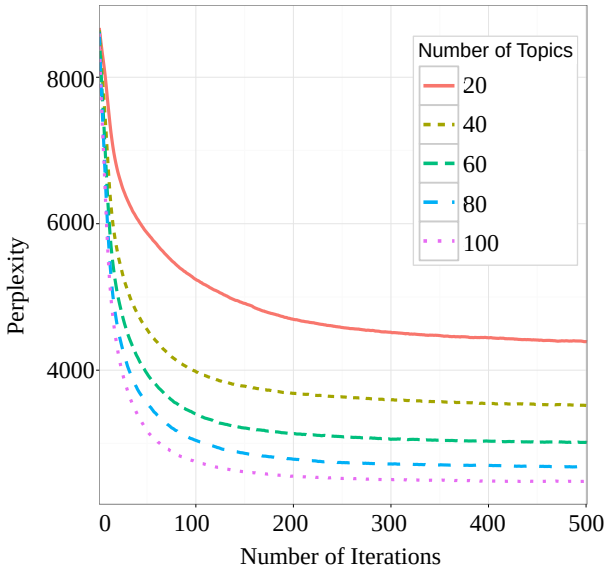
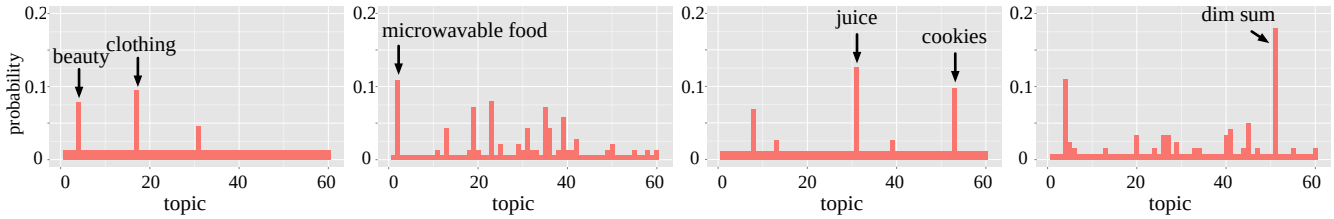


Figure 6: Perplexity as function of collapsed Gibbs sampling iterations for different number of topics used in LDA.

Figure 7 shows three example product topics learned by LDA using 100 topics. Each table shows the ten products that are most likely to be bought in that topic. Columns in the table represent the product name, the probability of the product being purchased in that topic, and the ground-truth category of the product, respectively. We see that Topic 1 is about "pasta." It contains a variety of cooked pasta and pasta sauce. Topic 18 and 53 are respectively about "bread and cakes" and "women accessories."

Figure 8 shows the topic proportions of four randomly selected users learned by LDA using 60 topics. We see that different users have distinguishable topic proportions, representing their purchasing preferences. For example, user #3617 purchased many products that are about "beauty" and "clothing" so her or his topic proportion has higher probabilities at topic 4 and topic 17. Similarly, user #39 tends to buy products in the "dim sum" category which can be represented in her or his topic proportion.



(a) User #3617 (b) User #5 (c) User #2618 (d) User #39

Figure 8: Examples illustrating learned topic proportions of four randomly selected users. For user #3617, the two most probable topics are "beauty" and "clothing".

Product	Prob.	Category
Chicken pasta (cream sauce)	0.0197	*Pasta*
Chicken pasta (pesto sauce)	0.0193	*Pasta*
Pork pasta (tomato sauce)	0.0167	*Pasta*
Pork steak	0.0155	*Meat*
Bacon pasta (cream sauce)	0.0153	*Pasta*
Spicy pasta (tomato sauce)	0.0149	*Pasta*
Clam garlic linguine	0.0146	*Pasta*
Tomato sauce pasta	0.0142	*Pasta*
German sausage sauce	0.0132	*Pasta*
Italian pasta (cooked)	0.0129	*Pasta*

(a) Topic 1, "pasta"

Product	Prob.	Category
Ham sandwich	0.0101	*Bread*
Cheese sandwich	0.0089	*Bread*
Milk bar cookie	0.0080	*Cookie*
Cherry chocolate tart	0.0078	*Cake*
Cheese roll	0.0077	*Cake*
Cheese almond tart	0.0074	*Cake*
Taro toast	0.0073	*Bread*
Creme Brulee	0.0073	*Cake*
Raisin toast	0.0071	*Bread*
Wheat ham sandwich	0.0070	*Bread*

(b) Topic 18, "bread and cakes"

Product	Prob.	Category
Knit Hat	0.0169	*Accessory*
Knit Scarf	0.0165	*Accessory*
Legging	0.0133	*Clothing*
Wool scarf	0.0120	*Accessory*
Long Pant	0.0111	*Clothing*
Cotton Socks	0.0099	*Accessory*
Wool Gloves	0.0097	*Accessory*
Facial Masks	0.0090	*Body Care*
Wool socks	0.0088	*Accessory*
Brown knit scarf	0.0081	*Accessory*

(c) Topic 53, "women accessories"

Figure 7: Illustration of product topics learned by LDA using 100 topics. Category is from ground truth.

5.3 Performance of PCIM and GICIM

We use lift charts to measure the effectiveness of PCIM and GICIM for predicting group-purchasing customers. In a lift chart, the x-axis represents the percentage of users sorted by our prediction score and the y-axis represents the cumulative percentage of the ground-truth customers we would predict. For all lift charts shown in this section, we also include two baseline models for comparison. One baseline model is to predict potential customers by *randomly sampling* from the set of users. Therefore, it is a straight line with slope 1.0 on the lift chart. Another baseline model, which we call category frequency, is to predict customers with the most frequent purchase history in a given product category. Specifically, to predict potential customers given a group-deal product category, we rank each customer in descending order of their normalized purchase frequency for the given product category.

Effect of w. We first measure the effect of the weighting parameter w in PCIM, which is shown in Figure 9. The particular w controls how much the user's own topic proportion is used in the mixture topic proportion. For example, $w = 1$ means that only the user's own topic proportion is used as the mixture topic proportion. We see that for all w values, PCIM performs much better than the baseline models between 0% and 25% of the customers predicted. For instance, PCIM is able to reach 50% of the targeted customers while the two baseline models only reach respectively 25% and 40% of the customers.

We also see that with $w = 1$, the curve first rises very fast, then flattens between 25% and 50%. It even performs worse than the baseline models starting at around 60% of the customers predicted; however this is the least interesting part of the curve. The intuition behind this behavior is that with $w = 1$, PCIM is good at predicting customers who have strong purchasing preferences that match the targeted group-deal product. On the other hand, for users without such strong purchasing preferences, the model is not able to per-

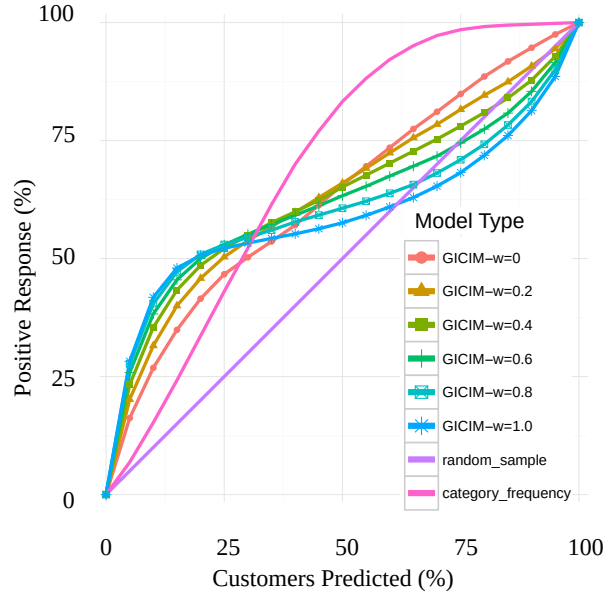


Figure 9: Lift chart of PCIM with different weighting parameter values. With $w = 1$, the model only includes the user's own topic proportion.

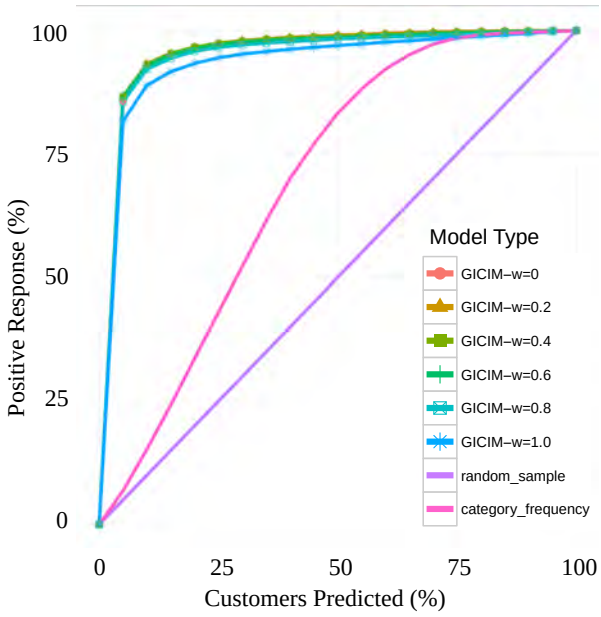


Figure 10: Lift chart of GICIM with different weighting parameter values. With $w = 1$, the model only includes the user's own topic proportion.

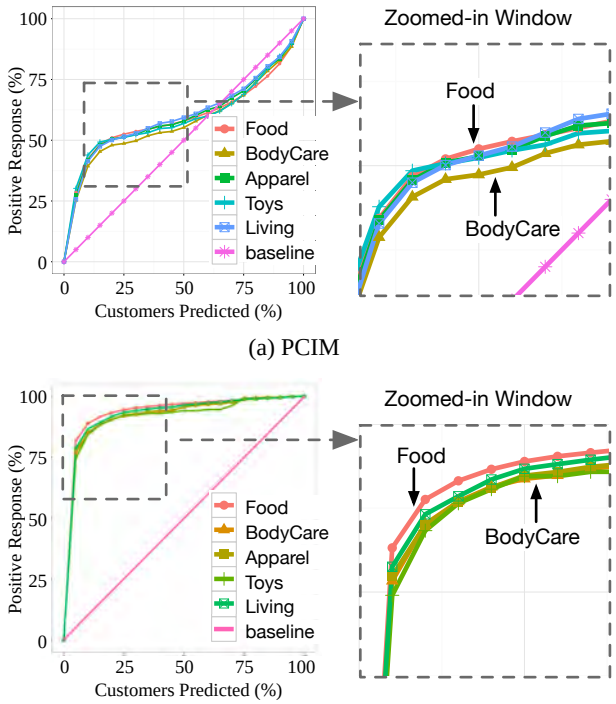


Figure 11: Lift charts of PCIM and GICIM over different product categories. The zoomed-in windows on the right show that performance is slightly better on frequently purchased items (*Food*) than on infrequently purchased items (*Body Care*).

form well. In general, by introducing the topic proportions of initiators with whom the user has co-bought products ($w < 1$), PCIM is able to reduce the flattening effect. With $w = 1$, we see that the lift curve is always above the baseline.

Figure 10 shows the effect of w in GICIM. We see that GICIM always performs better than PCIM and the baseline model even for the case where $w = 1$. In particular, for the cases where $w < 0.8$, GICIM achieves 90% positive response with only 10% of the predicted customers. The high prediction success of GICIM can be explained by the fact that whether a user chooses to join a group-purchasing event or not depends on who the group initiator is. We also note that the performance change due to different w values is not as significant as for PCIM.

Performance on different product categories. We next investigate whether frequently purchased items (e.g., drinks and food items) make PCIM and GICIM perform differently. We test on five different categories of group-deal products: *food*, *body care*, *apparel*, *toys*, and *living*. The ground-truth categories are from the dataset.

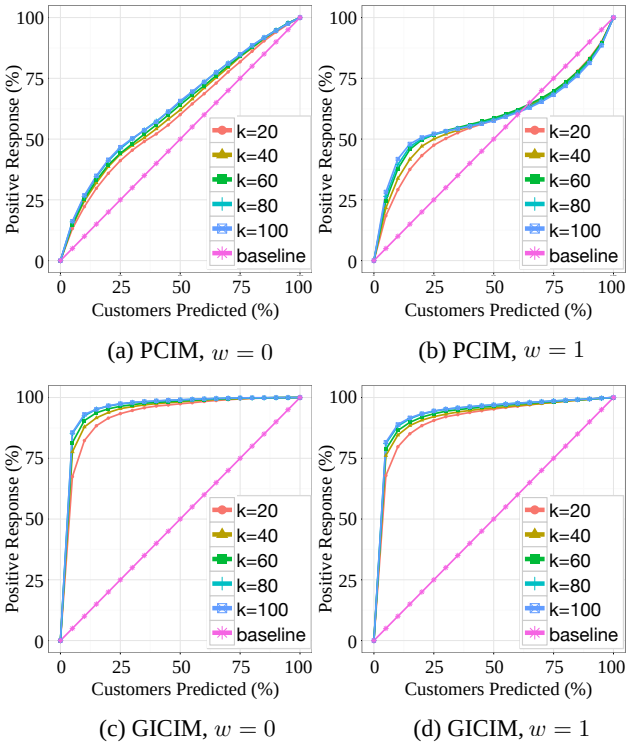


Figure 12: Lift charts of PCIM and GICIM over different number of topics.

Figure 11 shows the results. We see that, for all product categories tested, GICIM still performs better than PCIM. Moreover, from Figure 11, we see that both models are slightly better at predicting potential customers for the *food* category than for the *body care* category. We hypothesize that this may be related to the fact that purchases in the *food* category are more frequent and predictable compared to purchases in the *body care* category. A customers may buy one or more products in the *food* category repeatedly, while the same does not appear to be the case for all products in the *body care* category. For example, once someone has purchased a sunscreen spray (a product in *body care*), they are probably unlikely to buy it again, at least for the time span that our dataset covers. However, note that the differences between categories in Figure 11 are small, and developing a better understanding of them is an area of future research.

Effect of different number of topics. We ran PCIM and GICIM on different number of topics used in LDA. Results are given in Figure 12. We find that increasing the number of topics increases prediction accuracy for both models. This agrees with the above perplexity analysis that higher number of topics results in better performance.

6 Conclusion

In this paper, we study group-purchasing patterns with social information. We analyze a real-world group-purchasing dataset (5,602 users, 26,619 products, and 13,609 events) from ihergo.com. To the best of our knowledge, we are the first to analyze the group-purchasing scenario where each group-purchasing event is started by an initiator. Under this kind of social group-purchasing framework, each user builds up social ties with a set of group initiators. Our analysis of the dataset shows that a user usually joins group-purchasing events initiated by a certain and relatively small number of initiators. That is, if a user has co-bought a group-deal product with a group initiator, he or she is more likely to join a group-purchasing event started by that initiator again.

We develop two models to predict which users are most likely to join a group-purchasing event. Experimental results show that by including the weighted topic proportions of the initiators, we achieve higher prediction accuracy. We also find that whether a user decides to join a group-purchasing event is strongly influenced by who the group initiator of that event is.

Our model can be further improved in several ways. First, we can use Labeled LDA [16] by exploiting the ground-truth category of the products or user profile from the dataset. Second, we can incorporate other information such as the geographical and demographic information of users, and the seasonality of products in a more complex topic model. We are also interested in investigating the model to deal with *cold start*, where a new user or group-deal product is added to the system.

Acknowledgments

We want to thank Kuo-Chun Chang from ihergo.com for his cooperation and the use of ihergo dataset. This material is based, in part, upon work supported by NSF award CCF0937044 to Ole Mengshoel.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6), June 2005.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3, Mar. 2003.

[3] J. W. Byers, M. Mitzenmacher, M. Potamias, and G. Zervas. A month in the life of Groupon. *CoRR*, 2011.

[4] J. W. Byers, M. Mitzenmacher, and G. Zervas. Daily deals: Prediction, social diffusion, and reputational ramifications. *CoRR*, abs/1109.1530, 2011.

[5] Y. Cha and J. Cho. Social-network analysis using topic models. SIGIR '12, 2012.

[6] B. Edelman, S. Jaffe, and S. D. Kominers. To groupon or not to groupon: The profitability of deep discounts. Harvard business school working papers, Harvard Business School, Dec. 2010.

[7] S. Guo, M. Wang, and J. Leskovec. The role of social networks in online shopping: Information passing, price of trust, and consumer choice. *CoRR*, abs/1104.0942, 2011.

[8] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. SIGIR '03, 2003.

[9] T. Iwata, S. Watanabe, T. Yamada, and N. Ueda. Topic tracking model for analyzing consumer purchase behavior. In *IJCAI*, 2009.

[10] T. Iwata, T. Yamada, and N. Ueda. Modeling social annotation data with content relevance using a topic model. In *In NIPS*, 2009.

[11] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1), May 2007.

[12] T. Lu and C. E. Boutilier. Matching models for preference-sensitive group purchasing. EC '12. ACM, 2012.

[13] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. SIGIR '09. ACM, 2009.

[14] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. WSDM '11. ACM, 2011.

[15] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. KDD '08. ACM, 2008.

[16] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. EMNLP '09. Association for Computational Linguistics, 2009.

[17] H. M. Wallach. Topic modeling: beyond bag-of-words. ICML '06. ACM, 2006.

[18] X. Yang, Y. Guo, and Y. Liu. Bayesian-inference based recommendation in online social networks. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints), 2012.

[19] M. Ye, C. Wang, C. Aperjis, B. A. Huberman, and T. Sandholm. Collective attention and the dynamics of group deals. *CoRR*, abs/1107.4588, 2011.

[20] L. Yu, R. Pan, and Z. Li. Adaptive social similarities for recommender systems. RecSys '11. ACM, 2011.

An Object-oriented Spatial and Temporal Bayesian Network for Managing Willows in an American Heritage River Catchment

Lauchlin Wilkinson
Faculty of IT,
Monash Univ., AUS

Yung En Chee
School of Botany,
Univ. of Melbourne, AUS

Ann E. Nicholson
Faculty of IT,
Monash Univ., AUS

Pedro Quintana-Ascencio
Department of Biology,
Univ. of Central Florida, USA

Abstract

Willow encroachment into the naturally mixed landscape of vegetation types in the Upper St. Johns River Basin in Florida, USA, impacts upon biodiversity, aesthetic and recreational values. To control the extent of willows and their rate of expansion into other extant wetlands, spatial context is critical to decision making. Modelling the spread of willows requires spatially explicit data on occupancy, an understanding of seed production, dispersal and how the key life-history stages respond to environmental factors and management actions. Nicholson et al. (2012) outlined the architecture of a management tool to integrate GIS spatial data, an external seed dispersal model and a state-transition dynamic Bayesian network (ST-DBN) for modelling the influence of environmental and management factors on temporal changes in willow stages. That paper concentrated on the knowledge engineering and expert elicitation process for the construction and scenario-based evaluation of the prototype ST-DBN. This paper extends that work by using object-oriented techniques to generalise the knowledge organisational structure of the willow ST-DBN and to construct an object-oriented spatial Bayesian network (OOSBN) for modelling the neighbourhood spatial interactions that underlie seed dispersal processes. We present an updated architecture for the management tool together with algorithms for implementing the dispersal OOSBN and for combining all components into an integrated tool.

1 INTRODUCTION

The highly-valued Upper St. Johns River in Florida, USA has been the focus of considerable restoration investment (Quintana-Ascencio et al., 2013). However, woody shrubs, primarily Carolina willow (*Salix caroliniana* Michx.), have invaded areas that were historically herbaceous marsh (Kinser et al., 1997). This change to the historical composition of mixed vegetation types is considered undesirable, as extensive willow thickets detract from biodiversity, aesthetic and recreational values. Overabundance of willows reduces local vegetation heterogeneity and habitat diversity. People also prefer open wetlands that offer a viewshed, navigable access and scope for recreation activities such as wildlife viewing, fishing and hunting.

Managers seek to control the overall extent of willows, their rate of expansion into other extant wetland types and encroachment into recently restored floodplain habitats. Spatial context is critical to decision-making as areas differ in terms of biodiversity, aesthetic and recreational value, "invasibility" and applicable interventions. For instance, vegetation communities that are intact or distant from willow populations (seed sources) are less susceptible to invasion. With respect to interventions, mechanical clearing is restricted to areas where the substrate can support heavy machinery; prescribed fire depends on water levels and the quantity of "burnable" understorey vegetation.

Modelling willow spread requires spatially explicit data on willow occupancy, an understanding of seed production, dispersal, germination and survival, and how the key life-history stages respond to environmental factors and management actions. Data and knowledge on these pieces of the puzzle are available from ecological and physiological theory, surveys, field and laboratory experiments and domain experts.

State-transition (ST) models are a convenient means of organising information and synthesising knowledge to represent system states and transitions that are of

management interest. We build on recent studies that combine ST models with BNs to incorporate uncertainty in hypothesised states and transitions, and enable sensitivity, diagnostic and scenario analysis for decision support in ecosystem management (e.g. Bashari et al., 2009; Rumpff et al., 2011). Our approach uses the template described by Nicholson and Flores (2011) to explicitly model temporal changes in willow stages.

Nicholson et al. (2012) outlined the architecture of a management tool that would integrate GIS spatial data, a seed dispersal model and a state-transition dynamic Bayesian network (ST-DBN) for modelling the influence of environmental and management factors on temporal changes in willow stages. That paper described the knowledge engineering and expert elicitation process for the construction and scenario-based evaluation of the prototype ST-DBN. This paper extends that work, using object-oriented techniques to generalise the knowledge organisational structure of the willow ST-DBN and to construct an object-oriented spatial Bayesian network (OOSBN) for modelling the neighbourhood spatial interactions that underlie seed dispersal processes. We present an updated architecture for the management tool that incorporates GIS data and the new ST-OODBN and OOSBN structures, together with algorithms for implementing the dispersal OOSBN and for combining all components into an integrated tool.

2 BACKGROUND

Dynamic Bayesian Networks (DBNs) are a variant of ordinary BNs (Dean and Kanazawa, 1989; Nicholson, 1992) that allow explicit modelling of changes over time. A typical DBN has nodes for N variables of interest, with copies of each node for each *time slice*. Links in a DBN can be divided into those between nodes in the same time slice, and those in the next time slice. While DBNs have been used in some enviromental applications (e.g. Shihab, 2008), their uptake has been limited.

State-and-transition models (STMs) have been used to model changes over time in ecological systems that have clear transitions between distinct states (e.g., in rangelands and woodlands, see Bestelmeyer et al., 2003; Rumpff et al., 2011). Nicholson and Flores (2011) proposed a template for state-transition dynamic Bayesian networks (ST-DBNs) which formalised and extended Bashari et al.'s model, combining BNs with the qualitative STMs.

The influence of environmental and management factors on the main willow stages of management interest and their transitions is shown in our updated version of the Nicholson et al. (2012) ST-DBN (see Figure 1). For each cell (spatial unit), data on attributes such as soil, vegetation type and information about landscape position and context is supplied from GIS data. This data provides inputs to parameterise the ST-DBN and dispersal model. A cell size of 100m x 100m (1 ha) was chosen to represent a modelling unit. This reflects the resolution of available spatial data for environmental attributes, makes the computational demand associated with seed dispersal modelling feasible, and is a reasonable scale with respect to candidate management actions. A time step of one year was considered appropriate given the willow's growth and seed production cycle (Nicholson et al., 2012).

Seed production depends on the size and number of reproductive (adult) stems *within* each cell. However, *Seed Availability*, the amount of seed available for germination within a cell, depends on willow seed production and dispersal from *surrounding* cells. As these processes are not accounted for in the ST-DBN (Figure 1), a key focus of this paper is the development and integration of an object-oriented spatial Bayesian network (OOSBN) to model the neighbourhood spatial interactions that underlie this process.

The purpose of the integrated tool is to synthesise current understanding and quantify important sources of uncertainty to support decisions on *where*, *when* and *how* to control willows most effectively. The ST-DBN models willow state transitions and characteristics in response to environmental and management factors within a single spatial unit and time step. For coherent, effective and well-coordinated landscape-scale management however, we want to be able to predict willow response across *space* (at every cell) in the target area and across *time* frames of management interest (e.g. 10-20 years). Such predictions can then be mapped and also aggregated across the target area to produce evaluation metrics for managers. Such a tool would enable managers to "test", visually compare and quantitatively evaluate different candidate management strategies.

This real-world management problem is naturally described in terms of hierarchies of components that include similar, repetitive structures. Object-oriented (OO) modelling has obvious advantages in this context. We apply OO techniques to generalise the knowledge organisational structure of the willow ST-DBN and design and construct the seed production and dispersal spatial network.

3 AN ST-OODBN FOR WILLOWS

Various authors have advocated the use of OO modelling techniques to: a) help manage BN complexity via abstraction and encapsulation, b) facilitate the

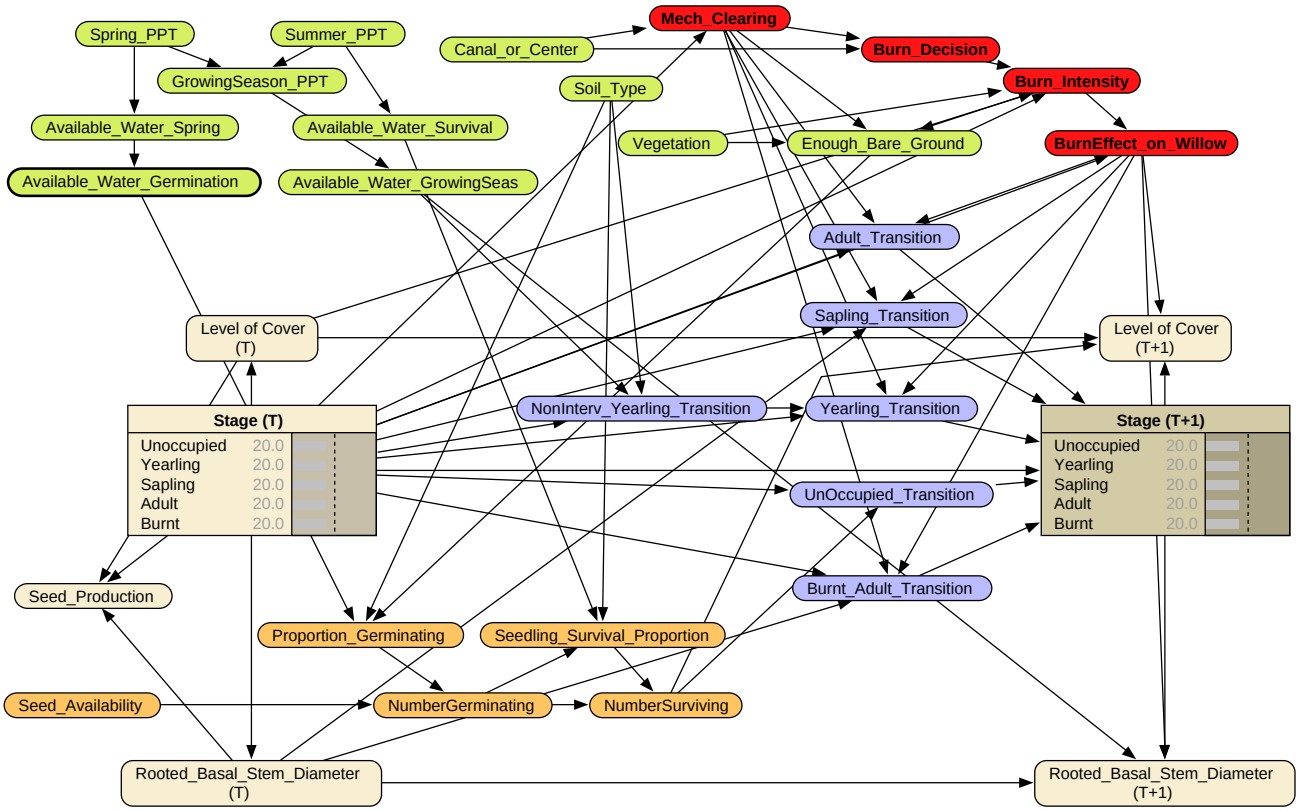


Figure 1: State-and-transition dynamic Bayesian network (ST-DBN) for modelling the response of key willow stages to environmental factors and management actions within a single spatial unit. The willow stages of management interest are: unoccupied, yearling, sapling, adult and burnt adult. Colours indicate: (i) aspects of willow state in tan; (ii) seed availability, germination and seedling survival processes in orange; (iii) environmental factors in green; (iv) management options in red; and (v) willow state-transitions in purple.

construction of classes of objects that are internally cohesive and potentially more reuseable, and c) formalise interfaces prior to integration (Koller and Pfeffer, 1997; Neil et al., 2000; Kjærulff and Madsen, 2008; Korb and Nicholson, 2010; Molina et al., 2010). However, examples in ecological and environmental management are scant (Molina et al., 2010; Carmona et al., 2011; Johnson and Mengersen, 2012).

We follow the definition of OOBNs used in Kjærulff and Madsen (2008), and implemented in the Hugin BN software package. A standard BN is made up of ordinary nodes, representing random variables. An OOBN class is made up of both nodes, and objects, which are instances of other classes. Thus an object may *encapsulate* multiple sub-networks, giving a composite and hierarchical structure. Objects are connected to other nodes via some of its own ordinary nodes, called its *interface* nodes. The rest of the nodes are not visible to the outside world, thus hiding information detail, another key OO concept. A class can be thought of as a self-contained 'template' for an OOBN object, described by its name, its interface and its hidden part. Finally, interface nodes are divided into input nodes and output nodes. Input nodes are the root nodes within an OOBN class, and when an object (instance) of that class becomes part of another class, each input node may be mapped to a single node (with the same state space) in the encapsulating class. The output nodes are the only nodes that may become parents of nodes in the encapsulating class. When displaying an OOBN, we show Hugin[1] screen shots, where input nodes are indicated with a dotted and shaded outline, and output nodes with a bold and shaded outline.

We converted the ST-DBN (Figure 1) into a ST-OODBN as follows. Using the five conceptual categories of nodes from the original network as a guide, the network was split into two abstract class types. The first represents abstract influencing factors and consists of three sub-classes that define *Environmental Conditions*, *Management Options* and the germination and seedling survival *Processes Factors*. The second type represents state transitions of willows

[1]Note that Hugin OOBNs do not support inheritance, so there are no super-classes or sub-classes.

(see Figure 2) and defines five sub-classes, *TransitionFromUnoccupied*, *TransitionFromYearling*, *TransitionFromSapling*, *TransitionFromAdult* and *TransitionFromBurntAdult*. Figure 3 illustrates the definition of the *TransitionFromYearling* class.

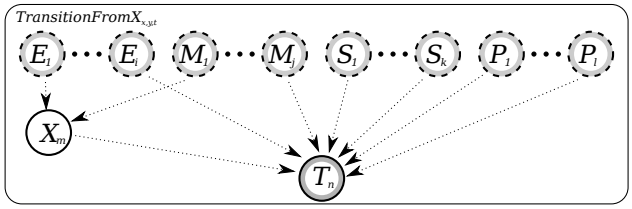


Figure 2: An abstract OOBN class for state transitions. Each implementation of a state transition output node T_n is defined by a combination of environmental conditions $E_{1...i}$, management options $M_{1...j}$, previous state variables $S_{1...k}$, process factors $P_{1...l}$ and any number of X_m hidden nodes. The only required input is the node that defines the previous state, all others are optional and are based on the implementing class. Dotted arrows indicate possible connections between input, hidden and output nodes.

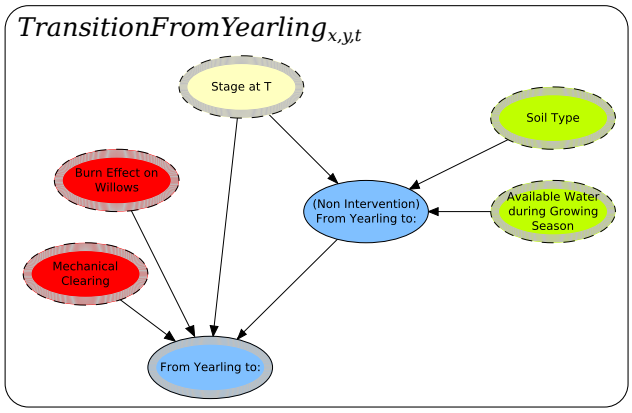


Figure 3: The *TransitionFromYearling* implementation of the abstract *TransitionFromX* type showing *Management Options* in red, *State Variables* in yellow and *Environmental Conditions* in green. The output *Transition* node *Transition from Yearling to:* is shown in blue with a bold and shaded outline.

These classes are instantiated as objects within a ST-OODBN class and when integrated with the seed dispersal OOSBN (described below) defines the complete ST model over a single time step (Figure 4). Recasting the network as a ST-OODBN makes the knowledge organisational structure explicit, whilst allowing network complexity to be hidden and integration efforts to focus on the interfaces between components of the network. Note that in the ST-OODBN (Figure 4) *Seed Availability* is an input node. *Seed Production* and its spatial dispersal is modelled by a separate OO network, which we present next.

4 AN OOSBN FOR SEED PRODUCTION AND DISPERSAL

S.caroliana flowers in early spring and produces very large numbers of small seeds (∼165,000 per average adult)that disperse by wind and water. Seed production is modelled by the *Willow Seed Production* OOBN (Figure 5), which is embedded in a broader seed dispersal model described below.

The number of seeds produced by an adult is given by the product of the number of *Inflorescences*, the number of *Fruits per inflorescence* and the number of *Seeds per fruit*. *Fruits per inflorescence* and *Seeds per fruit* are defined by distributions estimated from empirical data. The number of *Inflorescences* increases as a function of adult size (represented by *Rooted Basal Stem Diameter*) and this relationship has also been estimated from empirical data.

Cover is the percentage of a 1 hectare cell that is occupied by willows and *Average Canopy Area* is modelled as a function of *Rooted Basal Stem Diameter*. Together these two variables provide an estimate of the number of reproductive stems. Overall seed production within a cell, *Seeds per Hectare*, is then simply the product of the seed production per stem, by the number of reproductive stems. This *Willow Seed Production* OOBN models seed production processes explicitly rather than implicitly as in the ST-DBN prototype (Figure 1); an example of iterative and incremental knowledge engineering.

Willow seeds do not exhibit dormancy and have only a short period of viability – those that fail to germinate in the year they are produced are lost. The amount of seed available for germination within a cell depends on seed production and dispersal from surrounding cells. Thus, neighbourhood seed production and dispersal in combination with environmental and management factors determines patterns of willow spread and colonization.

Our approach to modelling seed dispersal is phenomenological rather than mechanistic. Wind-mediated seed dispersal is calculated using the Clark et al. (1999) dispersal kernel:

$$SD_{x,y}^{x',y'} = SP_{x',y'} \times \frac{1}{2\pi\alpha^2} e^{-(\frac{d}{\alpha})} \quad (1)$$

where $SD_{x,y}^{x',y'}$ is the number of seeds arriving at cell (x, y) from those produced at a cell (x', y'); it is the product of seed produced $SP_{x',y'}$ and an exponential

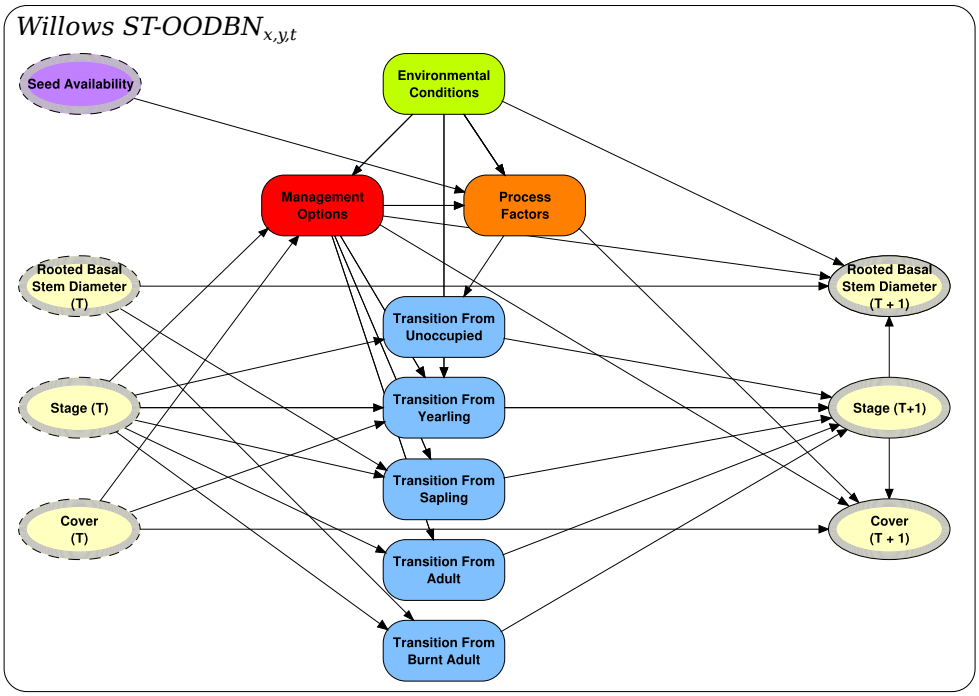


Figure 4: The resultant ST-OODBN class showing the four input nodes (*Seed Availability*, *Rooted Basal Stem Diameter (T)*, *Stage (T)* and *Cover (T)*) along with the *Environmental Conditions*, *Management Options*, *Process Factors* and five *TransitionFromX* objects that define the outputs (*Stage (T+1)*, *Rooted Basal Stem Diameter (T+1)* and *Cover (T+1)*) after one time step. Input nodes are illustrated with a dotted and shaded outline, instances of OOBN classes as round cornered boxes and output nodes with a bold and shaded outline.

kernel where d is the distance between cells (x, y) and (x', y'), and α is a distance parameter. To simulate stochasticity in dispersal events, α is a random variable that can be sampled from distributions designed to reflect the expected nature of dispersal (e.g. short versus long distance dispersal) (Fox et al., 2009). This seed dispersal model is captured within the *WindDispersalKernel*$_{x',y',t}$ OOBN (Figure 5), where the input *Distance* node is set for the particular (x, y) and α is set as a discretised normal distribution with a mean of 1 and a variance of 0.25.

For our purposes, we want to compute the seed availability $SA_{x,y}$ for a target cell (x, y), which is the sum of the seeds dispersed to it from every cell in the study area:

$$SA_{x,y} = \sum_{x',y' \in Area} SD_{x,y}^{x',y'} \qquad (2)$$

A naive BN model of this additive function would mean a *Seed Availability* node with all the *Seeds Dispersed* nodes (one for every cell) as its parents! For a study area with width w cells, height h cells, a *Seed Availability* node discretized to n states, and the *Seeds Dispersed* node discretized to m states, the CPT for *Seed Availability* would include $n \times m^{w \times h}$ probabilities –clearly infeasible.

From an ecological perspective, however, not *all* cells within a study area are expected to contribute towards final *Seed Availability* at any given cell. Indeed, because the number of seeds dispersed from a seed producing cell declines exponentially with increasing distance from that cell, we can make a simplifying assumption that after a certain distance, the number of seeds dispersed is effectively negligible. As a starting point we assume a circular region of influence for the target cell, defined by a dispersal mask with radius r. So for instance, a radius of eight cells (800 meters) implies $\pi 8^2$, or $\sim$201 cells providing parents to the final *Seed Availability* node. This is still far too many, particularly as we are using a standard BN software package with discrete nodes and exact inference.

However, since *Seed Availability* is a simple additive function, we use the simple modelling trick of adding the *Seed Availability* from each cell to the cumulative seed availability so far (via node *Cumulative Seed Availability*). This is equivalent to repeatedly divorcing parents to reduce the size of the state space. We can think of this is as sequentially scanning over the spatial dimension in a similar way to rolling out a DBN over time. We call this a spatial Bayesian network (SBN), and the object-oriented variety an OOSBN.

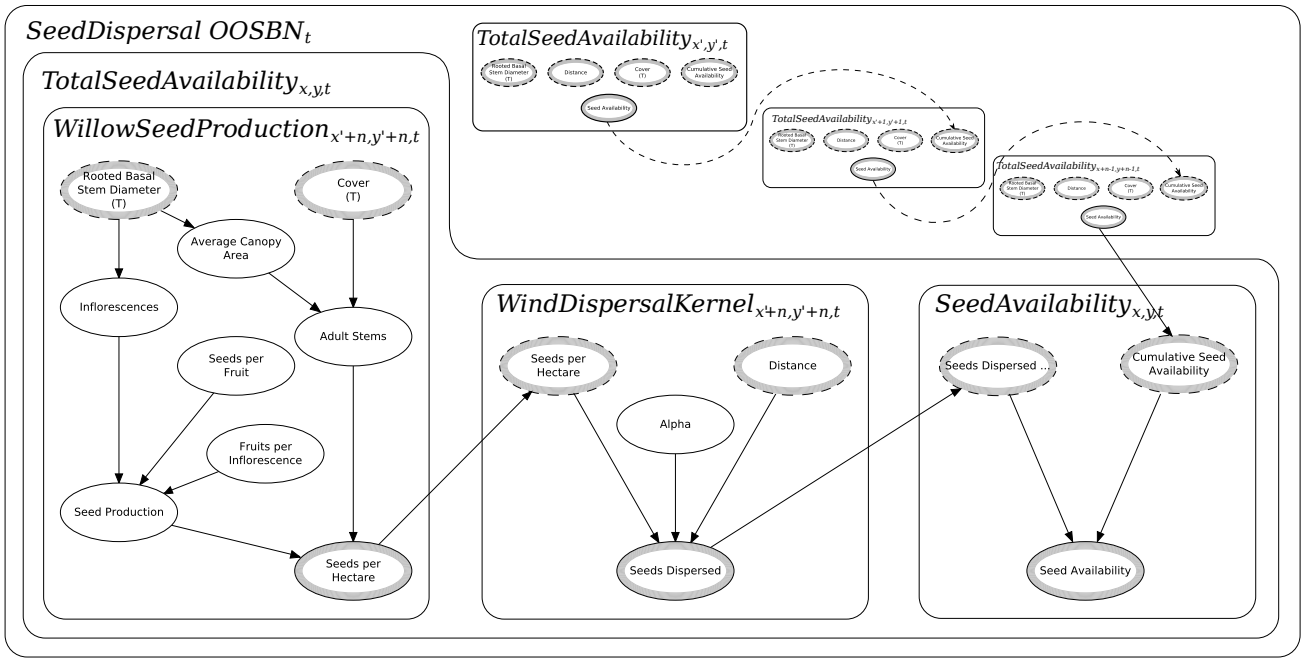


Figure 5: The OOSBN architecture showing how *Cover*, *Rooted Basal Stem Diameter*, *Cumulative Seed Availability* and *Distance* at locations (x', y') to $(x'+n, y'+n)$ are combined to provide total seed availability for each cell at location (x, y) at time t. Dashed arcs indicate that nodes are connected via the seed availability *PTLayer*. Multiple TotalSeedAvailability objects are illustrated within the SeedDispersal OOSBN reflecting that the total seeds available at a given (x, y) is dependant on seeds being produced in multiple locations. These synthetic TotalSeedAvailabilty objects are shown as collapsed objects, hiding their private nodes, and showing just the four input nodes that define the seeds dispersed from each producing cell to the target cell.

Our approach to integrating the seed dispersal makes use of such an OOSBN, that is run πr^2 times (i.e. once for every cell (x', y') within dispersal range) for each cell (x, y) in the study area to disperse and then sum the seed availability at each cell. The dispersal mask is flexible and can be designed to take on different shapes to reflect potentially important influences on wind dispersal such wind direction, wind strength and terrain characteristics. However, in the results given Section 6, we use a radius of eight cells.

Overall, our approach here is to mitigate the problem of large CPT sizes by turning the problem in to one of computation time. This has the added benefit of potentially being able to be computed easily in parallel and thus regaining some computation efficiency.

5 Integrating the ST-OODBN with the OOSBN

Figure 6 is an abstract representation of the system architecture, specifically, the interactions between the GIS layers and the OO networks as the tool is used for prediction at yearly intervals over the required management time frame.

For each cell in the GIS, there is conceptually one state-transition network (*ST-OODBN*) and one seed production and dispersal spatial network, (*OOSBN*). In practice, we do not store all these as separate networks, but rather re-use a single network structure, whose input nodes are re-parameterised for each cell, for each time step.

For the first time step, the system takes GIS data, which represents the initial conditions of the study area. These are stored in an internal data structure, *PTLayer*, that combines the spatial structure of the GIS, with distributions for the (discrete) nodes in the networks. *PTLayers* are used to store and pass the spatially referenced prior distributions of input nodes and posterior distributions of output nodes for both the *OOSBN* and *ST-OODBN* (Figure 6). Each *PTLayer* contains a number of fields, one for each of the node states of the linked input and output nodes. Each field stores the probability mass of the corresponding node state.

The *PTLayer* distributions are used to set the priors for the input nodes at each time step t. Then inference (belief updating) is performed within the OO network, producing new posterior probability distri-

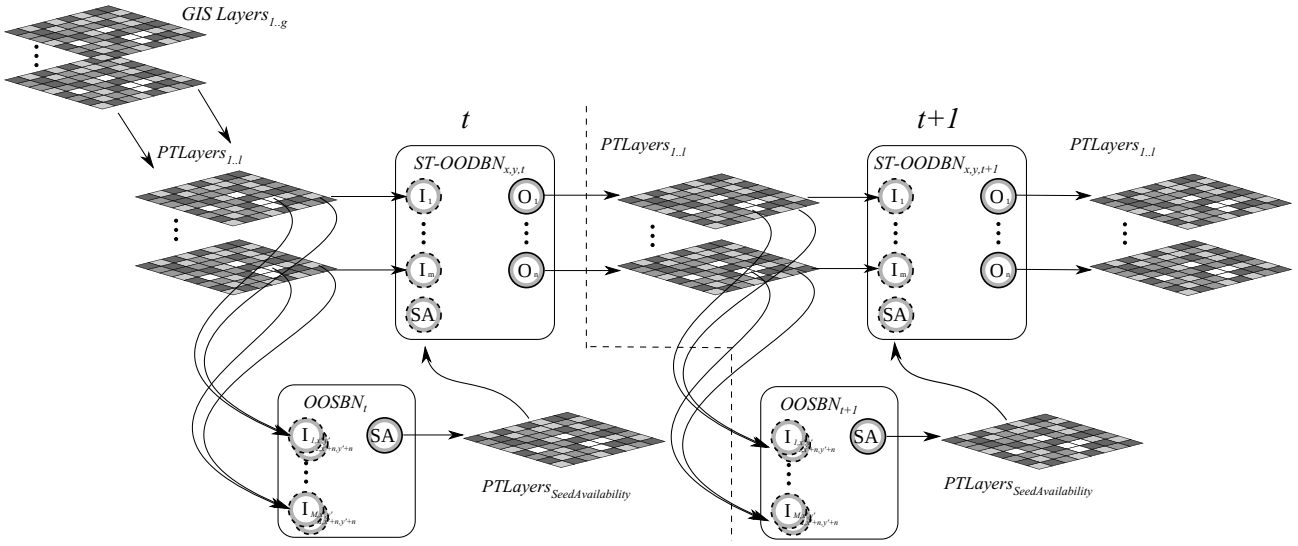


Figure 6: An abstract representation of the management tool architecture.

butions from the output nodes for each OO network, which are then transferred back to the corresponding *PTLayer*. This is equivalent to the standard "roll-out" followed by "roll-up" steps done in prediction with two time slice DBNs (Boyen and Koller, 1998) to avoid the computational complexity of rolling out a DBN over a large number of time steps. But here, in addition, the *PTLayers* are being used as an intermediate storage *across* the spatial grid, with the inputs for the next time $t+1$ coming not only from both the network for the same cell, but from outputs from networks for *other* cells. This is done via the seed dispersal OOSBN, which uses the *Seed Availability PTLayer* to accumulate the seed availability arising from seed production in other cells within the dispersal mask. In effect, the *PTLayer* replaces both the temporal arcs if the network was rolled-out over many time steps, and the spatial arcs between the networks for different cells, which are essentially the cross-network arcs from seed production in one place to seed availability in another. Note that this method is limited to prediction only –we cannot use the model for diagnosis, or to identify the starting states and management actions to achieve a preferred end-state.

More formally, using the notation from Figure 6, at time t and at each cell location (x, y), $PTLayers_{1...l}$ are used to initialise the priors of nodes $I_{1...m}$ and $I_{1...M}$ of $ST\text{-}OODBN_{x,y,t}$ and $OOSBN_t$ respectively. After propagation within $OOSBN_t$, beliefs from the output node SA are stored in $PTLayer_{SeedAvailabilty}$ and then used to update the priors of input node SA of $ST\text{-}OODBN_{x,y,t}$. Then belief updating is done for the $ST\text{-}OODBN_{x,y,t}$ for each cell(x, y) and the beliefs from output nodes $O_{1...n}$ propagated back to $PTLayers_{1..l}$ at time $t+1$. Although the mapping between the set of $PTLayers_{1...l}$ and input nodes $I_{1...m}$ is one-to-one, there may be cases where there are no GIS layers available for an input, in which case a prior distribution is used. Finally, with respect to OOSBN propagation, the range of locations $x', y' \ldots x'+n, y'+n$ (i.e. neighbourhood cells) that are included is defined by the dispersal mask described in the previous section.

Algorithm 1 An algorithm for propagating a GIS coupled ST-OODBN with spatial OOSBN subnetworks

```
1:  function PROPAGATE(ST− OODBN, OOSBN, ptlayers, t)
2:      I ← I(ST-OODBN)
3:      O ← O(ST-OODBN)
4:      SA ← getLayer(ptlayers, SeedAvailability)
5:      for t := 0 to t do
6:          PROPAGATE(OOSBN,ptlayers,dispMask)
7:          // computes all Seed Availabilities
8:          for all (x, y) ∈ Area do
9:              p(SeedAvailability) ← SA_{x,y}
10:             for all I_i ∈ I do
11:                 L_j ← getLayer(ptlayers, I_i)
12:                 p(I_i) ← L_j(x, y)
13:             end for
14:             update beliefs in ST-OODBN
15:             for all O_i ∈ O do
16:                 L_j ← getLayer(ptlayers, O_i)
17:                 L_j(x, y) ← Bel(O_i)
18:             end for
19:         end for
20:     end for
21: end function
```

The process is detailed in Algorithm 1, as a function $PROPAGATE$ which takes the ST-OODBN (shown in Fig. 4), and OOSBN (shown in Fig. 5) networks, a list of *ptlayers* (previously initialised from the GIS layers) whose cells correspond to the area under consideration, and the number of time steps T over which to propagate the network. In the algorithm, we use

$I(OOBN)$ (respectively $O(OOBN)$) to denote a function that returns the input (resp. output) nodes of the interface of the OOBN, and *getLayer(ptlayers,V)* to denote a function that returns the *PTLayer* corresponding to a node V.

Algorithm 2 An algorithm for dispersing seeds by wind using an OOSBN class

```
1: function PROPAGATE(OOSBN, ptlayers, dispMask)
2:     SA ← getLayer(ptlayers, SeedAvailability)
3:     for all (x, y) ∈ Area do
4:         P(SA_{x,y} = none) ← 1
5:     end for
6:     for all (x, y) ∈ Area do
7:         for all I_i ∈ I_{x,y}(OOSBN) do
8:             L_j ← getLayer(ptlayers, I_i)
9:             p(I_i) ← L_j(x, y)
10:        end for
11:        for all (x', y') ∈ dispMask do
12:            p(CumSeedAvail) ← SA_{x,y}
13:            d ← √((x − x')² + (y − y')²)
14:            p(Distance = d) ← 1
15:            update beliefs in OOSBN
16:            SA_{x,y} ← Bel(SeedAvailability)
17:        end for
18:    end for
19: end function
```

First, seed dispersal is done with the OOSBN using Algorithm 2. Then for each cell (x, y) in the area, for each input node I_i, the distribution for that cell from its corresponding *ptlayer* is set as the prior of the input node. Belief updating of the ST-OODBN is done, propagating the new priors through to updated posterior distributions for the output nodes. Finally the beliefs for each output node ($Bel(O_i)$) are copied back to the distribution at the (x, y) cell for the corresponding *ptlayer* (i.e. into $SA_{x,y}$).

Algorithm 2 details the propagation process using the *SeedDispersal OOSBN* class illustrated in Figure 5. The algorithm starts by taking the OOSBN, a list of *PTLayers* whose cells correspond to the area under consideration, and a dispersal mask (*dispMask*). Before starting the dispersal process, the provided *Seed Availability PTLayer* is initialised with no seeds available at all (x, y) co-ordinates. It then loops through each cell (x, y) in the area, setting the distribution for each input node I_i (i.e., *Cover*, *Rooted Basal Stem Diameter* and *Cumulative Seed Availability*) from its corresponding *PTlayer* cell. The algorithm then enters a second loop for every cell that is a possible seed source based on the dispersal mask. The *Distance* node is set using the Euclidean distance between the current cell and the target co-ordinates. Finally, belief updating is done within the OOSBN (Figure 5) and the beliefs (i.e. the posterior probability distribution) from the *Seed Availability* node at each point are transferred into the *Cumulative Seed Availability* for the subsequent cell at each iteration. After all the cells (x', y') within the dispersal mask have been visited, the *Seed Availability PTLayer*, $SA_{x,y}$ contains the overall seed availability in that cell, which is used for modelling germination in the ST-OODBN in Algorithm 1.

6 PRELIMINARY RESULTS

To implement the software architecture described we chose to use Hugin Researcher 7.7(2013) to develop the ST-OODBN and dispersal model OOSBN, the Hugin Researcher Java API 7.7 (2013) to provide programmatic access to the developed networks, the ImageIO-ext (2013) java library to provide access to GIS raster layer formats, and the Java programming language to implement the algorithms tying the components together. Hugin was chosen as the OOBN development platform as it currently has one of the most complete OOBN implementations. Java was chosen as the implementing language as it is platform independent and provides for a well established and understood OO development environment. We implemented the tool as a standalone program allowing preprocessing of GIS data to be performed in whatever program the end user was most familiar with. In our case we used a combination of ArcGIS (2013), Quantum GIS (2013)and SAGA GIS(2013).

To demonstrate our working implementation, we ran the model for the Blue Cypress Marsh Conservation Area (138 x 205 cells) within the Upper St. Johns River basin. We used a simplified (and unrealistic) management rule set that says if a cell is next to a canal, mechanical clearing is carried out, otherwise for landlocked cells, burning is prescribed (with a probability of 0.1). Maps of willow cover and seed production were generated at yearly intervals for a 25 year prediction window. This took about 8 hours of computation on a 64bit machine with a 2.8GHz processor. Figure 7 shows seed availability across the study area using output from the *SeedAvailability* nodes in the OOSBN at 5 yearly intervals. To produce the maps, the seed availability interval with the highest posterior probability distribution is used to produce a grayscale value where *zero* seeds is black, and 10^{12} is white. In the run shown, seed availability decreases over time as the level of willow cover is reduced by the management regime.

7 DISCUSSION AND FUTURE WORK

For coherent, coordinated and effective landscape-scale decision support, managers need the capability to predict willow state changes across *space* and *time*. We have tackled the challenges of this real-world problem by synthesising ideas and techniques from object-oriented knowledge engineering, dynamic BNs, GIS-

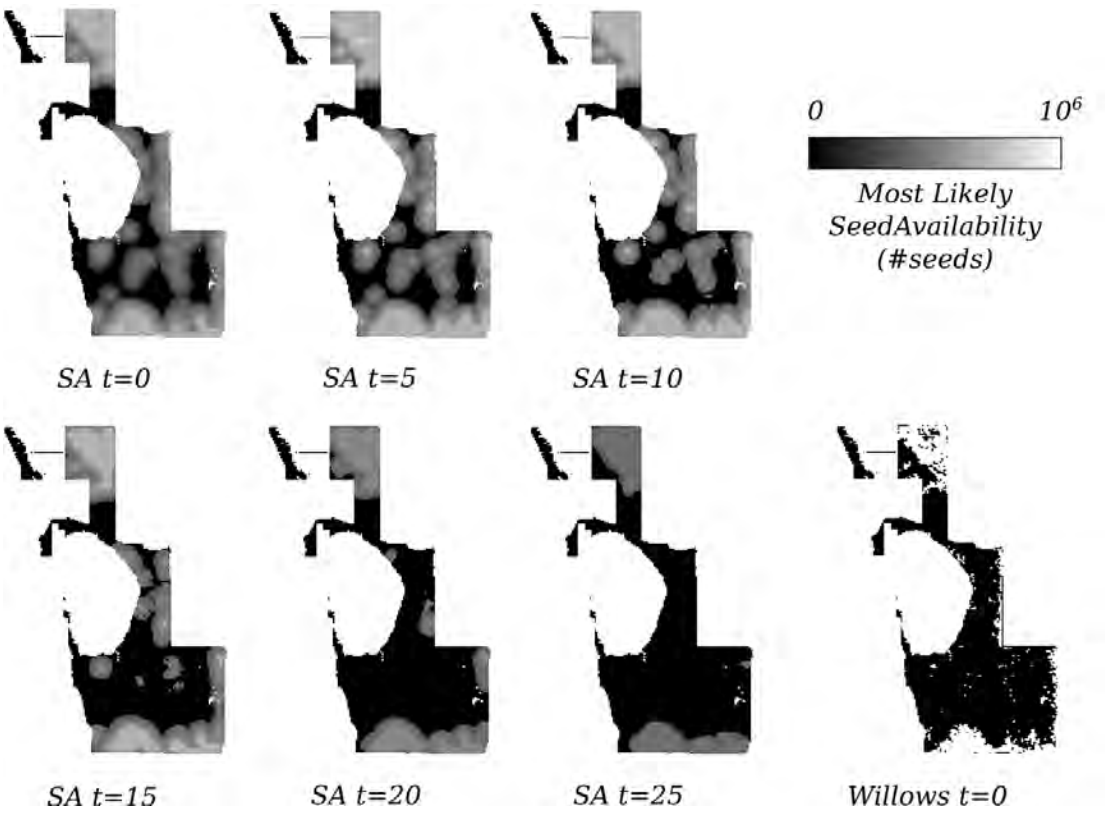


Figure 7: Seed availability predicted by the Willows ST-OODBN at $t = 0, 5, 10, 15, 20, 25$ years, across the Blue Cypress Marsh Conservation Area (138 x 208 cells). Adult willow occupancy at $t = 0$ is shown in the bottom right panel; black indicates absence, grey presence.

coupled BNs and dispersal modelling. To our knowledge, this is the first environmental management application in which OOBNs are used to model spatially-explicit process interactions.

Further work in the development of this management tool includes developing a water dispersal model and updating the parametrisation of the *ST-OODBN* and *OOSBN* using judgements from a larger pool of domain experts, together with specific empirical data where available. In addition, we will work with managers and domain experts to identify: i) realistic management scenarios, ii) useful summary descriptors for the various model outputs and iii) desirable features for a tool interface.

Throughout the research and integration process we encountered challenges with the development, management and use of OOBNs. While there have been advances in OOBN software, they still lack a lot of the useful features available in other development tools. For instance, modern software engineering IDEs provide easy to use re-factoring, documentation and integration with version control tools. The tools we used to design and implement the underlying OOBNs for our tool still lack powerful refactoring, making the management of object interface changes a time consuming and error prone task. Integrated source control is non-existent and documentation tools rudimentary. Improvements in these areas would make working with OOBNs far more accessible to the type of user that wishes to make use of OOBNs for natural resource management.

With respect to spatialising the ST-OOBN with the use of OOSBNs, there is currently no graphical tool up to the task of facilitating the integration of the required components. This means that anyone wanting to replicate our work would need to make use of the available APIs and this constitutes a barrier to usage by people with no or little programming background.

Acknowledgements

This work was supported by ARC Linkage Project LP110100304 and the Australian Centre of Excellence for Risk Analysis. John Fauth (UCF), and Dianne Hall, Kimberli Ponzio and Ken Snyder (SJWRMD) provided critical information about the St Johns River ecosystem, and knowledge about willow invasion.

References

Bashari, H., Smith, C., and Bosch, O. (2009). Developing decision support tools for rangeland management by combining state and transition models and Bayesian belief networks. *Agricultural Systems*, 99(1):23–34.

Bestelmeyer, B. T., Brown, J. R., Havstad, K. M., Alexander, R., Chavez, G., and Herrick, J. E. (2003). Development and use of state-and-transition models for rangelands. *Journal of Range Management*, (2):114–126.

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 33–42, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Carmona, G., Molina, J., and Bromley, J. (2011). Object-Oriented Bayesian networks for participatory water management: two case studies in Spain. *Journal of Water Resources Planning and Management*, 137(4):366–376.

Clark, J. D., Silman, M., Kern, R., Macklin, E., and Hille Ris Lambers, J. (1999). Seed dispersal near and far: patterns across temperate and tropical forests. *Ecology*, 80(5):1475–1494.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

Esri (2013). ArcGIS (Version 10.1) [Software]. `http://www.esri.com/software/arcgis`.

Fox, J. C., Buckley, Y. M., Panetta, F. D., Bourgoin, J., and Pullar, D. (2009). Surveillance protocols for management of invasive plants: modelling Chilean needle grass (*Nassella neesiana*) in Australia. *Diversity and Distributions*, 15(4):577–589.

GeoSolutions (2013). Image-IO-ext Java Library (Version 1.1.7) [Software]. `http://github.com/geosolutions-it/`.

Hugin Expert A/S (2013). Hugin Researcher (Version 7.7) [Software]. `http://www.hugin.com`.

Johnson, S. and Mengersen, K. (2012). Integrated Bayesian network framework for modeling complex ecological issues. *Integrated Environmental Assessment and Management*, 8(3):480–90.

Kinser, P., Lee, M. A., Dambek, G., Williams, M., Ponzio, K., and Adamus, C. (1997). Expansion of Willow in the Blue Cypress Marsh Conservation Area, Upper St. Johns River Basin. Technical report, St. Johns River Water Management District, Palatka, Florida.

Kjærulff, U. B. and Madsen, A. (2008). *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer Verlag, New York.

Koller, D. and Pfeffer, A. (1997). Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, UAI'97, pages 302–313, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Korb, K. B. and Nicholson, A. (2010). *Bayesian artificial intelligence*. Chapman&Hall/CRC, Boca Raton, FL, 2nd edition.

Molina, J., Bromley, J., García-Aróstegui, J., Sullivan, C., and Benavente, J. (2010). Integrated water resources management of overexploited hydrogeological systems using Object-Oriented Bayesian Networks. *Environmental Modelling & Software*, 25(4):383–397.

Neil, M., Fenton, N., and Nielson, L. (2000). Building large-scale Bayesian networks. *The Knowledge Engineering Review*, 15(3):1–33.

Nicholson, A., Chee, Y., and Quintana-Ascencio, P. (2012). A state-transition DBN for management of willows in an american heritage river catchment. In Nicholson, A., Agosta, J. M., and Flores, J., editors, *Ninth Bayesian Modeling Applications Workshop at the Conference of Uncertainty in Artificial Intelligence*, Catalina Island, CA, USA, http://ceur-ws.org/Vol-962/paper07.pdf.

Nicholson, A. E. (1992). *Monitoring Discrete Environments using Dynamic Belief Networks*. PhD thesis, Department of Engineering Sciences, Oxford.

Nicholson, A. E. and Flores, M. J. (2011). Combining state and transition models with dynamic Bayesian networks. *Ecological Modelling*, 222(3):555–566.

Quantum GIS Development Team (2013). Quantum GIS (Version 1.8.0) [Software]. `http://www.qgis.org/`.

Quintana-Ascencio, P. F., Fauth, J. E., Castro Morales, L. M., Ponzio, K. J., Hall, D., and Snyder, K. (2013). Taming the beast: managing hydrology to control Carolina Willow (*Salix caroliniana*) seedlings and cuttings. *Restoration Ecology*.

Rumpff, L., Duncan, D., Vesk, P., Keith, D., and Wintle, B. (2011). State-and-transition modelling for adaptive management of native woodlands. *Biological Conservation*, 144(4):1224–1236.

SAGA Development Team (2013). SAGA GIS (Version 2.0.8) [Software]. `http://www.saga-gis.org/`.

Shihab, K. (2008). Dynamic modeling of groundwater pollutants with Bayesian networks. *Applied Artificial Intelligence*, 22(4):352–376.

Exploring Multiple Dimensions of Parallelism in Junction Tree Message Passing

Lu Zheng
Electrical and Computer Engineering
Carnegie Mellon University

Ole J. Mengshoel
Electrical and Computer Engineering
Carnegie Mellon University

Abstract

Belief propagation over junction trees is known to be computationally challenging in the general case. One way of addressing this computational challenge is to use node-level parallel computing, and parallelize the computation associated with each separator potential table cell. However, this approach is not efficient for junction trees that mainly contain small separators. In this paper, we analyze this problem, and address it by studying a new dimension of node-level parallelism, namely *arithmetic parallelism*. In addition, on the graph level, we use a clique merging technique to further adapt junction trees to parallel computing platforms. We apply our parallel approach to both marginal and most probable explanation (MPE) inference in junction trees. In experiments with a Graphics Processing Unit (GPU), we obtain for marginal inference an average speedup of 5.54x and a maximum speedup of 11.94x; speedups for MPE inference are similar.

1 INTRODUCTION

Bayesian networks (BN) are frequently used to represent and reason about uncertainty. The junction tree is a secondary data structure which can be compiled from a BN [2, 4, 5, 9, 10, 19]. Junction trees can be used for both marginal and most probable explanation (MPE) inference in BNs. Sum-product belief propagation on junction tree is perhaps the most popular exact marginal inference algorithm [8], and max-product belief propagation can be used to compute the most probable explanations [2, 15]. However, belief propagation is computationally hard and the computational difficulty increases dramatically with the density of the BN, the number of states of each network node, and the treewidth of BN, which is upper bounded by the generated junction tree [13]. This computational challenge may hinder the application of BNs in cases where real-time inference is required.

Parallelization of Bayesian network computation is a feasible way of addressing this computational challenge [1,6,7,9,11,12,14,19,20]. A data parallel implementation for junction tree inference has been developed for a cache-coherent shared-address-space machine with physically distributed main memory [9]. Parallelism in the basic sum-product computation has been investigated for Graphics Processing Units (GPUs) [19]. The efficiency in using disk memory for exact inference, using parallelism and other techniques, has been improved [7]. An algorithm for parallel BN inference using pointer jumping has been developed [14]. Both parallelization based on graph structure [12] as well as node level primitives for parallel computing based on a table extension idea have been introduced [20]; this idea was later implemented on a GPU [6]. Gonzalez et al. developed a parallel belief propagation algorithm based on parallel graph traversal to accelerate the computation [3].

A parallel message computation algorithm for junction tree belief propagation, based on the cluster-sepset mapping method [4], has been introduced [22]. Cluster-sepset based node level parallelism (denoted *element-wise parallelism* in this paper) can accelerate the junction tree algorithm [22]; unfortunately the performance varies substantially between different junction trees. In particular, for small separators in junction trees, element-wise parallelism [22] provides limited parallel opportunity as explained in this paper.

Our work aims at addressing the small separator issue. Specifically, this paper makes these contributions that further speed up computation and make performance more robust over different BNs from applications:

- We discuss another dimension of parallelism, namely *arithmetic parallelism* (Section 3.1). Inte-

grating arithmetic parallelism with *element-wise parallelism*, we develop an improved parallel sum-product propagation algorithm as discussed in Section 3.2.

- We also develop and test a parallel max-product (Section 3.3) propagation algorithm based on the two dimensions of parallelism.

- On the graph level, we use a clique merging technique (Section 4), which leverages the two dimensions of parallelism, to adapt the various Bayesian networks to the parallel computing platform.

In our GPU experiments, we test the novel two-dimensional parallel approach for both regular sum-propagation and max-propagation. Results show that our algorithms improve the performance of both kinds of belief propagation significantly.

Our paper is organized as follows: In Section 2, we review BNs, junction trees parallel computing using GPUs, and the small-separator problem. In Section 3 and Section 4, we describe our parallel approach to message computation for belief propagation in junction trees. Theoretical analysis of our approach is in Section 5. Experimental results are discussed in Section 6, while Section 7 concludes and outlines future research.

2 BACKGROUND

2.1 Belief Propagation in Junction Trees

A BN is a compact representation of a joint distribution over a set of random variables $\mathcal{X}$. A BN is structured as a directed acyclic graph (DAG) whose vertices are the random variables. The directed edges induce dependence and independence relationships among the random variables. The evidence in a Bayesian network consists of instantiated random variables.

The junction tree algorithm propagates beliefs (or posteriors) over a derived graph called a junction tree. A junction tree is generated from a BN by means of moralization and triangulation [10]. Each vertex C_i of the junction tree contains a subset of the random variables and forms a clique in the moralized and triangulated BN, denoted by $\mathcal{X}_i \subseteq \mathcal{X}$. Each vertex of the junction tree has a potential table $\phi_{\mathcal{X}_i}$. With the above notations, a junction tree can be defined as $J = (\mathbb{T}, \Phi)$, where $\mathbb{T}$ represents a tree and Φ represents all the potential tables associated with this tree. Assuming C_i and C_j are adjacent, a separator S_{ij} is induced on a connecting edge. The variables contained in S_{ij} are defined to be $\mathcal{X}_i \cap \mathcal{X}_j$.

The junction tree size, and hence also junction tree computation, can be lower bounded by *treewidth*, which is defined to be the minimal size of the largest junction tree clique minus one. Considering a junction tree with a treewidth t_w, the amount of computation is lower-bounded by $O(\exp(c*t_w))$ where c is a constant.

Belief propagation is invoked when we get new evidence $\boldsymbol{e}$ for a set of variables $\mathcal{E} \subseteq \mathcal{X}$. We need to update the potential tables Φ to reflect this new information. To do this, belief propagation over the junction tree is used. This is a two-phase procedure: evidence collection and evidence distribution. For the evidence collection phase, messages are collected from the leaf vertices all the way up to a designated root vertex. For the evidence distribution phase, messages are distributed from the root vertex to the leaf vertices.

2.2 Junction Trees and Parallelism

Current emerging many-core platforms, like the recent Graphical Processing Units (GPUs) from NVIDIA and Intel's Knights Ferry, are built around an array of processors running many threads of execution in parallel. These chips employ a Single Instruction Multiple Data (SIMD) architecture. Threads are grouped using a SIMD structure and each group shares a multithreaded instruction unit. The key to good performance on such platforms is finding enough parallel opportunities.

We now consider opportunities for parallel computing in junction trees. Associated with each junction tree vertex $\mathcal{C}_i$ and its variables $\mathcal{X}_i$, there is a potential table $\phi_{\mathcal{X}_i}$ containing non-negative real numbers that are proportional to the joint distribution of $\mathcal{X}_i$. If each variable contains s_j states, the minimal size of the potential table is $|\phi_{\mathcal{X}_i}| = \prod_{j=1}^{|\mathcal{X}_i|} s_j$, where $|\mathcal{X}_i|$ is the cardinality of $\mathcal{X}_i$.

Message passing from $\mathcal{C}_i$ to an adjacent vertex $\mathcal{C}_k$, with separator $\mathcal{S}_{ik}$, involves two steps:

1. **Reduction step**. In sum-propagation, the potential table $\phi_{\mathcal{S}_{ik}}$ of the separator is updated to $\phi^*_{\mathcal{S}_{ik}}$ by reducing the potential table $\phi_{\mathcal{X}_i}$:

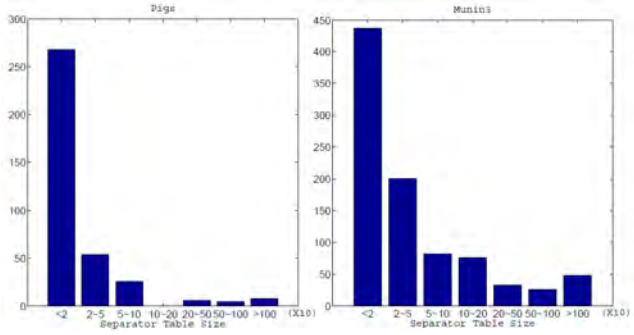
$$\phi^*_{\mathcal{S}_{ik}} = \sum_{\mathcal{X}_i / \mathcal{S}_{ik}} \phi_{\mathcal{X}_i}. \tag{1}$$

2. **Scattering step**. The potential table of $\mathcal{C}_k$ is updated using both the old and new table of $\mathcal{S}_{ik}$:

$$\phi^*_{\mathcal{X}_k} = \phi_{\mathcal{X}_k} \frac{\phi^*_{\mathcal{S}_{ik}}}{\phi_{\mathcal{S}_{ik}}}. \tag{2}$$

We define $\frac{0}{0} = 0$ in this case, that is, if the denominator in (2) is zero, then we simply set the corresponding $\phi^*_{\mathcal{X}_k}$ to zeros.

Figure 1: Histograms of the separator potential table sizes of junction trees *Pigs* and *Munin3*. For both junction trees, the great majority of the separator tables contain 20 or fewer elements.



Equation (1) and (2) reveal two dimensions of parallelism opportunity. The first dimension, which we return to in Section 3, is arithmetic parallelism. The second dimension is element-wise parallelism [22].

Element-wise parallelism in junction trees is based on the fact that the computation related to each separator potential table cell are independent, and takes advantage of an index mapping table, see Figure 2. In Figure 2, this independence is illustrated by the white and grey coloring of cells in the cliques, the separator, and the index mapping tables. More formally, an *index mapping table* $\mu_{\mathcal{X},\mathcal{S}}$ stores the index mappings from $\phi_{\mathcal{X}}$ to $\phi_{\mathcal{S}}$ [4]. We create $|\phi_{\mathcal{S}_{ik}}|$ such mapping tables. In each mapping table $\mu_{\mathcal{X}_i,\phi_{\mathcal{S}_{ik}}(j)}$ we store the indices of the elements of $\phi_{\mathcal{X}_i}$ mapping to the j-th separator table element. Mathematically,

$$\mu_{\mathcal{X}_i,\phi_{\mathcal{S}_{ik}}(j)} = \{r \in [0, |\phi_{\mathcal{X}_i}| - 1] \mid \phi_{\mathcal{X}_i}(r) \text{ is mapped to } \phi_{\mathcal{S}_{ik}}(j)\}.$$

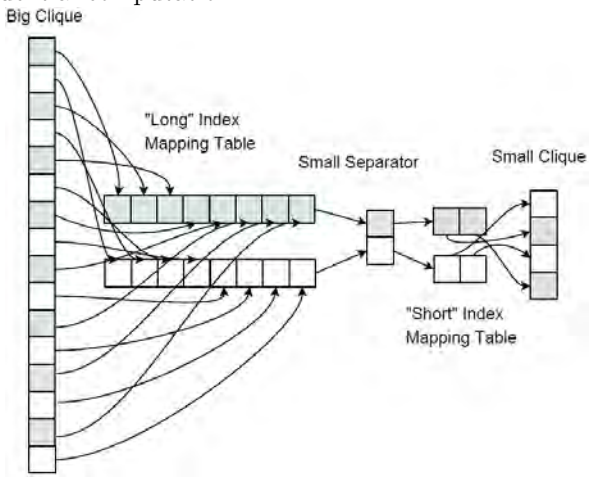
With the index mapping table, element-wise parallelism is obtained by assigning one thread per mapping table of a separator potential table as illustrated in Figure 2 and Figure 3. Consequently, belief propagation over junction trees can often be sped up by using a hybrid CPU/GPU approach [22].

2.3 Small Separator Problem

Figure 1 contains the histograms of two real-world BNs *Pigs* and *Munin3*.[1] We see that most separators in these two BNs are quite small and have a potential table size of less than 20.

In general, small separators can be found in these three scenarios: (i) due to two small neighboring cliques (we call it the *S-S-S* pattern); (ii) due to a small intersection set of two big neighboring cliques (the *B-S-B* pattern); and (iii) due to one small neighboring clique and one big neighboring clique (the *B-S-S* pattern).[2] Due to parallel computing issues, detailed next, these three patterns characterize what we call the *small-separator problem*.

Figure 2: Due to the small separator in the *B-S-S* pattern, a long index mapping table is produced. If only element-wise parallelism is used, there is just one thread per index mapping table, resulting in slow sequential computation.



Unfortunately, element-wise parallelism may not provide enough parallel opportunities when a separator is very small and the mapping tables to one or both cliques are very long. A small-scale example, reflecting the *B-S-S* pattern, is shown in Figure 2.[3] While the mapping tables may be processed in parallel, the long mapping tables result in a significant amount of sequential computation within each mapping table.

A state of the art GPU typically supports more than one thousand concurrent threads, thus message passing through small separators will leave most of the GPU resources idle. This is a major bottleneck for the performance, which we address next.

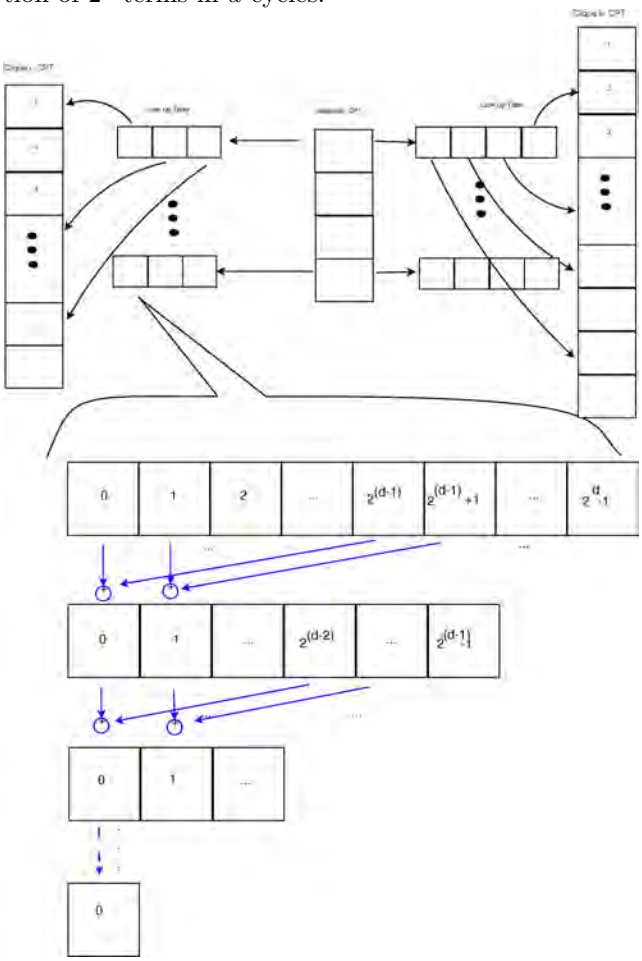
In this paper, to handle the small separator problem, we use clique merging to eliminate small separators (see Section 4) resulting from the *S-S-S* pattern and arithmetic parallelism (see Section 3) to attack the *B-*

[1]These BNs can be downloaded at `http://bndg.cs.aau.dk/html/bayesian_networks.html`

[2]These patterns, when read left-to-right, describe the size of a clique, the size of a neigboring separator, and the size of a neigboring clique (different from the first) as found in a junction tree. For example, the pattern *B-S-S* describes a big clique, a small separator, and a small clique.

[3]In fact, both the "long" and "short" index mapping tables have for presentation purposes been made short–in a realistic junction tree a "long" table can have more than 10,000 entries.

Figure 3: Example of data structure for two GPU cliques and their separator. Arithmetic parallelism (the reverse pyramid at the bottom) is integrated with element-wise parallelism (mapping tables at the top). The arithmetic parallelism achieves parallel summation of 2^d terms in d cycles.



S-S and B-S-B patterns.

3 PARALLEL MESSAGE PASSING IN JUNCTION TREES

In order to handle the small-separator problem, we discuss another dimension of parallelism in addition to *element-wise parallelism*, namely *arithmetic parallelism*. Arithmetic parallelim explores the parallel opportunity in the sum of (1) and in the multiplication of (2). By considering also arithmetic parallelism, we can better match the junction tree and the many-core GPU platform by optimizing the computing resources allocated to the two dimensions of parallelism.

Mathematically, this optimization can be modeled as a computation time minimization problem:

$$\begin{array}{c} \min_{p_e,p_a} T(p_e,p_a,\mathcal{C}_i,\mathcal{C}_j,\Phi), \\ \text{subject to}: p_e + p_a \leq p_{tot} \end{array} \quad (3)$$

where $T(\cdot)$ is the time consumed for a message passing from clique $\mathcal{C}_i$ to clique $\mathcal{C}_j$; p_e and p_a are the number of parallel threads allocated to the element-wise and arithmetic dimensions respectively; p_{tot} is the total number of parallel threads available in the GPU; and Φ is a collection of GPU-related parameters, such as the cache size, etc. Equation (3) is a formulation of optimizing algorithm performance on a parallel computing platform. Unfortunately, traditional optimization techniques can typically not be applied to this optimization problem. This is because the analytical form of $T(\cdot)$ is usually not available, due to the complexity of the hardware platform. So in our work we choose p_e and p_a empirically for our implementation.

In the rest of this section, we will describe our algorithm design, seeking to explore both element-wise and arithmetic parallelism.

3.1 Arithmetic Parallelism

Arithmetic parallelism needs to be explored in different ways for reduction and scattering, and also integrated with element-wise parallelism, as we will discuss now.

For reduction, given a certain fixed element j, Equation (1) is essentially a summation over all the clique potential table $\phi_{\mathcal{X}_i}$ elements indicated by the corresponding mapping table $\mu_{\mathcal{X}_i,\phi_{\mathcal{S}_{ik}}(j)}$. The number of sums is $|\mu_{\mathcal{X}_i,\phi_{\mathcal{S}_{ik}}(j)}|$. We compute the summation in parallel by using the approach illustrated in Figure 3. This improves the handling of long index mapping tables induced, for example, by the B-S-B and B-S-S patterns. The summation is done in several iterations. In each iteration, the numbers are divided into two groups and the corresponding two numbers in each

group are added in parallel. At the end of this recursive process, the sum is obtained, as shown in Algorithm 1. The input parameter d is discussed in Section 3.2; the remaining inputs are an array of floats.

Algorithm 1 ParAdd$(d, op(0), \ldots, op(2^d - 1))$

Input: $d, op(0), \ldots, op(2^d - 1)$.
sum = 0
for $i = 1$ *to* d **do**
  for $j = 0$ *to* $2^{d-i} - 1$ **in parallel do**
    $op(j) = op(j) + op(j + 2^{d-i})$
  end for
end for
return $op(0)$

For scattering, Equation (2) updates the elements of $\phi_{\mathcal{X}_k}$ independently despite that $\phi_{\mathcal{S}_{ik}}$ and $\phi^*_{\mathcal{S}_{ik}}$ are re-used to update different elements. Therefore, we can compute each multiplication in (2) with a single thread. The parallel multiplication algorithm is given in Algorithm 2. The input parameter p is discussed in Section 3.2; the remaining inputs are an array of floats.

Algorithm 2 ParMul$(p, op1, op2(0), \ldots, op2(p-1))$

Input: $p, op1, op2(0), \ldots, op2(p-1)$.
sum = 0
for $j = 0$ *to* $p - 1$ **in parallel do**
  $op2(j) = op1 * op2(j)$
end for

3.2 Parallel Belief Propagation

Combining element-wise and arithmetic parallelism, we design the reduction and scattering operations as shown in Algorithm 3 and Algorithm 4. Both of these algorithms take advantage of arithmetic parallelism. In Algorithm 3, the parameter d is the number of cycles used to compute the reduction (summation), while in Algorithm 4, p is the number of threads operating in parallel. Both p and d are parameters that determine the degree of arithmetic parallelism. They can be viewed as a special form of p_a in (3). Based on Algorithm 3 and Algorithm 4, junction tree message passing can be written as shown in Algorithm 5.

Belief propagation can be done using both breadth-first and depth-first traversal over a junction tree. We use the Hugin algorithm [5], which adopts depth-first belief propagation. Given a junction tree J with root vertex C_{root}, we first initialize the junction tree by multiplying together the Bayesian network potential tables (CPTs). Then, two phase belief propagation is adopted [10]: collect evidence and then distribute evidence [22].

Algorithm 3 Reduction$(d, \phi_{\mathcal{X}_i}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_i,\mathcal{S}_{ik}})$

Input: $d, \phi_{\mathcal{X}_i}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_i,\mathcal{S}_{ik}}$.
for $n = 1$ *to* $|\phi_{\mathcal{S}_{ik}}|$ **in parallel do**
  for $j = 0$ *to* $\lceil |\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}|/2^d \rceil$ **do**
    sum = sum + ParAdd$(d, \phi_{\mathcal{X}_i}(\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}(j * 2^d)), \ldots, \phi_{\mathcal{X}_i}(\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}((j+1) * 2^d - 1)))$
  end for
end for

Algorithm 4 Scattering$(p, \phi_{\mathcal{X}_k}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_k,\mathcal{S}_{ik}})$

Input: $p, \phi_{\mathcal{X}_k}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_k,\mathcal{S}_{ik}}$.
for $n = 1$ *to* $|\phi_{\mathcal{S}_{ik}}|$ **in parallel do**
  for $j = 0$ *to* $\lceil |\phi_{\mathcal{S}_{ik}}|/p \rceil$ **do**
    sum = sum + ParMul$(p, \frac{\phi^*_{\mathcal{S}_{ik}}(n)}{\phi_{\mathcal{S}_{ik}}(n)}, \phi_{\mathcal{X}_k}(\mu_{\mathcal{X}_k,\mathcal{S}_{ik}(n)}(j * p)), \ldots, \phi_{\mathcal{X}_k}(\mu_{\mathcal{X}_k,\mathcal{S}_{ik}(n)}((j+1) * p - 1)))$
  end for
end for

3.3 Max-product Belief Propagation

In this paper, we also apply our parallel techniques to max-product propagation (or in short, max-propagation), which is also referred as the Viterbi algorithm. Max-propagation solves the problem of computing a most probable explanation. For max-propagation, the $\sum$ in (1) is replaced by max [2,15]. In this paper we use sum-product propagation to explain our parallel algorithms; the explanation can generally be changed to discuss max-propagation by replacing add with max.

4 CLIQUE MERGING FOR JUNCTION TREES

The performance of our parallel algorithm is to a large extent determined by the degree of parallelism available in message passing, which intuitively can be measured by the separator size $|\phi_{\mathcal{S}_{ik}}|$, which determines the element-wise parallelism, and the mapping table size $|\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}|$ which upper bounds the arithmetic parallelism. In other words, the larger $|\phi_{\mathcal{S}_{ik}}|$ and $|\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}|$ are, the greater is the parallelism opportunity. Therefore, message passing between small cliques (the *S-S-S* pattern), where $|\phi_{\mathcal{S}_{ik}}|$ and $|\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}|$ are small, is not expected to have good performance. There is not enough parallelism to make full use of

Algorithm 5 PassMessage$(p, d, \phi_{\mathcal{X}_i}, \phi_{\mathcal{X}_k}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_i,\mathcal{S}_{ik}})$

Input: $p, d, \phi_{\mathcal{X}_i}, \phi_{\mathcal{X}_k}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_i,\mathcal{S}_{ik}}$.
Reduction$(d, \phi_{\mathcal{X}_i}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_i,\mathcal{S}_{ik}})$
Scattering$(p, \phi_{\mathcal{X}_k}, \phi_{\mathcal{S}_{ik}}, \mu_{\mathcal{X}_k,\mathcal{S}_{ik}})$

a GPU's computing resources.

In order to better use the GPU computing power, we propose to remove small separators (that follow the S-S-S pattern) by selectively merging neighboring cliques. This increases the length of mapping tables, however the arithmetic parallelism techniques introduced in Section 3 can handle this. Clique merging can be done offline according to this theorem [8].

Theorem 4.1 *Two neighboring cliques C_i and C_j in a junction tree J with the separator S_{ij} can be merged together into an equivalent new clique C_{ij} with the potential function*

$$\phi(x_{C_{ij}}) = \frac{\phi(x_{C_i})\phi(x_{C_j})}{\phi(x_{S_{ij}})}, \tag{4}$$

while keeping all the other part of the junction tree unchanged.

The result of merging cliques is three-fold: (i) it produces larger clique nodes and thus longer mapping tables; (ii) it eliminates small separators; and (iii) it reduces the number of cliques. Larger clique nodes will result in more computation and therefore longer processing time for each single thread, but getting rid of small separators will improve utilization of the GPU and reduce computation time. We have to manage these two conflicting objectives to improve the overall performance of our parallel junction tree algorithm.

Our algorithm for clique merging is shown in Algorithm 6. It uses two heuristic thresholds, the separator size threshold τ_s and the index mapping table size threshold τ_μ, to control the above-mentioned two effects. We only merge two neighboring cliques C_i and C_j into a new clique C_{ij} when $|\phi_{\mathcal{S}_{ij}}| < \tau_s$ and $|\mu_{\mathcal{X}_j,\phi_S}| < \tau_\mu$.

Algorithm 6 MergeCliques(J, τ_s, τ_μ)

```
merge_flag = 1
while merge_flag do
  merge_flag = 0
  for each adjacent clique pair (C_i, C_j) in J do
    if φ_S < τ_s and |μ_{X_j,φ_S}| < τ_μ then
      Merge (J, C_i, C_j)
      merge_flag = 1
    end if
  end for
end while
```

Given an S-S-S-S-S pattern, Algorithm 6 may merge two S cliques and produce an S-B-S pattern. Here, B is the merged clique. Note that the B-S sub-pattern creates a long index mapping table, which is exactly what arithmetic parallelism handles. There is in other words potential synergy between clique merging and arithmetic parallelism, as is further explored in experiments in Section 6.

5 ANALYSIS AND DISCUSSION

In this section, we analyze the theoretical speedup for our two-dimensional parallel junction tree inference algorithm under the idealized assumption that there is unlimited parallel threads available from the many-core computing platform.

The degree of parallelism opportunity is jointly determined by the size of the separators' potential table, $|\phi_{\mathcal{S}}|$, and the size of the index mapping table $|\mu_{\mathcal{X},\phi_S}|$. Consider a message passed from $\mathcal{C}_i$ to $\mathcal{C}_k$. Since we employ separator table element-wise parallelism in our algorithm, we only need to focus on the computation related to one particular separator table element. With the assumption of unlimited parallel threads, we can choose $d = \lceil \log |\mu_{\mathcal{X}_i,\phi_S}| \rceil$. The time complexity for the reduction is then $\lceil \log |\mu_{\mathcal{X}_i,\phi_S}| \rceil$, due to our use of summation parallelism.[4] Note since $|\mu_{\mathcal{X}_i,\phi_S}| = \frac{|\phi_{\mathcal{X}_i}|}{|\phi_S|}$, the time complexity can be written as $\lceil \log |\phi_{\mathcal{X}_i}| - \log |\phi_{\mathcal{S}}| \rceil$. For the scattering phase, we choose $p = |\mu_{\mathcal{X}_k,\phi_S}|$ and the time complexity is given by $|\mu_{\mathcal{X}_k,\phi_S}|/p + 1 = 2$ due to the multiplication parallelism. Thus the overall time complexity of the two-dimensional belief propagation algorithm is:

$$\lceil \log |\phi_{\mathcal{X}_i}| - \log |\phi_{\mathcal{S}}| \rceil + 2, \tag{5}$$

which is the theoretical optimal time complexity under the assumption of an infinite number of threads. Nevertheless, this value is hard to achieve in practice since the value of d and p are subject to the concurrency limit of the computing platform. For example, in the above-mentioned BN *Pigs*, some message passing requires $p = 1120$ while the GTX460 GPU supports at most 1024 threads per thread block.

Belief propagation is a sequence of messages passed in a certain order [10], for both CPU and GPU [22]. Let $Ne(\mathcal{C})$ denote the neighbors of $\mathcal{C}$ in the junction tree. The time complexity for belief propagation is

$$\sum_i \sum_{k \in Ne(\mathcal{C}_i)} \left(\lceil \log |\phi_{\mathcal{X}_i}| - \log |\phi_{\mathcal{S}}| \rceil + 2 \right),$$

Kernel invocation overhead, incurred each time Algorithm 5 is invoked, turns out to be an important performance factor. If we model the invocation overhead for each kernel call to be a constant τ, then the time

[4] We assume, for simplicity, sum-propagation. The analysis for max-propagation is similar.

complexity becomes

$$\sum_i d_i\tau + \sum_i \sum_{k\in Ne(\mathcal{C}_i)} (\lceil \log|\phi_{\mathcal{X}_i}| - \log|\phi_{\mathcal{S}}| \rceil + 2)\,,$$

where d_i is the degree of a node $\mathcal{C}_i$. In a tree structure, $\sum d_i = 2(n-1)$. Thus the GPU time complexity is

$$2(n-1)\tau + \sum_i \sum_{k\in Ne(\mathcal{C}_i)} (\lceil \log|\phi_{\mathcal{X}_i}| - \log|\phi_{\mathcal{S}}| \rceil + 2)\,.$$

From this equation, we can see that the junction tree topology impacts GPU performance in at least two ways: the total invocation overhead is proportional to the number of nodes in the junction tree, while the separator and clique table sizes determine the degree of parallelism.

The overall speedup of our parallel belief propagation approach is determined by the equation

$$Speedup = \frac{\sum_i \sum_{k\in Ne(\mathcal{C}_i)} (|\phi_{\mathcal{X}_i}| + |\phi_{\mathcal{X}_k}|)}{2(n-1)\tau + \sum_i \sum_{k\in Ne(\mathcal{C}_i)} \left(\left\lceil \log \frac{|\phi_{\mathcal{X}_i}|}{|\phi_{\mathcal{S}}|} \right\rceil + 2 \right)}.$$

Clearly, the speedup depends on the distribution of the sizes of the separators' and cliques' potential tables. That is the reason we propose the clique merging technique. Using clique merging, we change the number of nodes in the junction tree and distribution of the size of the separators' and cliques' potential table as well, adapting the junction tree for the specific parallel computing platform.

From the equations above, we can estimate the overall belief propagation speedup. However, taking into account that the CPU/GPU platform incurs invocation overhead and the long memory latency when loading data from slow device memory to fast shared memory, the theoretical speedup is hard to achieve in practice. We take an experimental approach to study how the structure of the junction trees affects the performance of our parallel technique on the CPU/GPU setting in Section 6.

6 EXPERIMENTAL RESULTS

In experiments, we study Bayesian networks compiled into junction trees. We not only want to compare the two-dimensional parallel junction tree algorithm to the sequential algorithm, but also study how effective the arithmetic parallelism and clique merging methods are. Consequently, we experiment with different combinations of element-wise parallelism (EP), arithmetic parallelism (AP), and clique merging (CM).

6.1 Computing Platform

We use the NVIDIA GeForce GTX460 as the platform for our implementation. This GPU consists of seven multiprocessors, and each multiprocessor consists of 48 cores and 48K on-chip shared memory per thread block. The peak thread level parallelism achieves 907GFlop/s. In addition to the fast shared memory, a much larger but slower off-chip global memory (785 MB) that is shared by all multiprocessors is provided. The bandwidth between the global and shared memories is about 90 Gbps. In the junction tree computations we are using single precision for the GPU and the thread block size is set to 256.

6.2 Methods and Data

For the purpose of comparison, we use the same set of BNs as used previously [22] (see `http://bndg.cs.aau.dk/html/bayesian_networks.html`). They are from different domains, with varying structures and state spaces. These differences lead to very different junction trees, see Table 1, resulting in varying opportunities for element-wise and arithmetic parallelism. Thus, we use clique merging to carefully control our two dimensions of parallelism to optimize performance. The Bayesian networks are compiled into junction trees and merged offline and then junction tree propagation is performed.

6.3 GPU Optimization: Arithmetic Parallelism

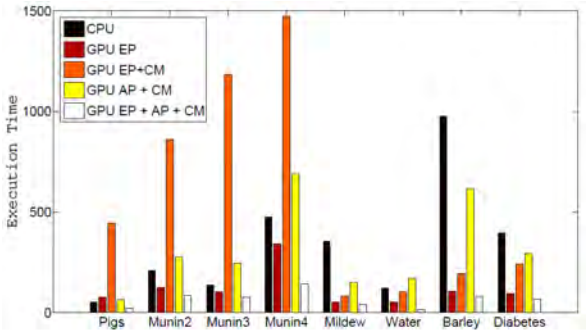
Arithmetic parallelism gives us more freedom to match the parallelism in message passing and the concurrency provided by a GPU: when there is not enough potential table element-wise parallelism available, we can increase the degree of arithmetic parallelism. The number of threads assigned to arithmetic parallelism affects the performance significantly. The parameter p in parallel scattering and the parameter 2^d in the parallel reduction should be chosen carefully (see Algorithm 1 and 2). Since the GPU can provide only limited concurrency, we need to balance the arithmetic parallelism and the element-wise parallelism for each message passing to get the best performance.

Consider message passing between big cliques, for example according to the *B-S-B* pattern. Intuitively, the values of the arithmetic parallelism parameters p and d should be set higher than for the message passing between smaller cliques. Thus, based on extensive experimentation, we currently employ a simple heuristic parameter selection scheme for the scattering parameter p

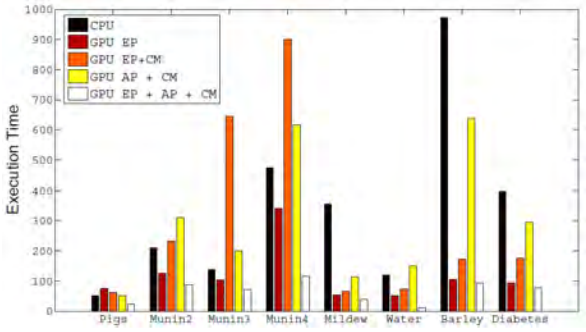
$$p = \begin{cases} 4 & \text{if} |\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}| \le 100 \\ 128 & \text{if} |\mu_{\mathcal{X}_i,\mathcal{S}_{ik}(n)}| > 100 \end{cases} \quad (6)$$

Dataset	Pigs	Munin2	Munin3	Munin4	Mildew	Water	Barley	Diabetes
# of original JT nodes	368	860	904	872	28	20	36	337
# of JT nodes after merge	162	553	653	564	22	18	35	334
Avg. CPT size before merge	1,972	5,653	3,443	16,444	341,651	173,297	512,044	32,443
Avg. CPT size after merge	5,393	10,191	7,374	26,720	447,268	192,870	527,902	33,445
Avg. SPT size before merge	339	713	533	2,099	9,273	26,065	39,318	1,845
Avg. SPT size after merge	757	1,104	865	3,214	11,883	29,129	40,475	1,860
GPU time (sum-prop) [ms]	22.61	86.40	74.99	141.08	41.31	16.33	81.82	68.26
GPU time (max-prop) [ms]	22.8	86.8	72.6	114.9	38.6	12.1	94.3	78.3
CPU time (sum-prop) [ms]	51	210	137	473	355	120	974	397
CPU time (max-prop) [ms]	59	258	119	505	259	133	894	415
Speedup (sum-prop)	**2.26x**	**2.43x**	**1.82x**	**3.35x**	**8.59x**	**7.35x**	**11.94x**	**5.81x**
Speedup (max-prop)	**2.58x**	**2.97x**	**1.64x**	**4.39x**	**6.71x**	**10.99x**	**9.48x**	**5.30x**

Table 1: Junction tree (JT) statistics and belief propagation (BP) performance for various junction trees, with speedup for our GPU approach (GPU EP + AP + CM) compared to CPU-only in the two bottom rows. The row "CPU time (sum-prop)" gives previous results [22].



(a) Junction tree sum-propagation



(b) Junction tree max-propagation

Figure 4: Comparison of combinations of junction tree optimization techniques CM, AP, and EP for (a) sum- and (b) max-propagation. Best performance is achieved for GPU EP + AP + CM.

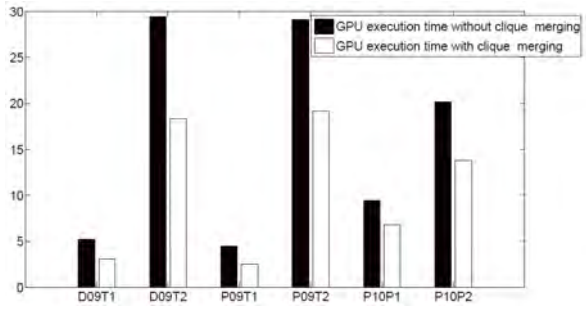


Figure 5: GPU execution times with CM (GPU EP + AP + CM) and without CM (GPU EP + AP) for junction trees compiled from sparse BNs representing electrical power systems.

and the reduction parameter d

$$d = \begin{cases} 2 & \text{if} |\mu_{\mathcal{X}_i, \mathcal{S}_{ik}(n)}| \leq 100 \\ 7 & \text{if} |\mu_{\mathcal{X}_i, \mathcal{S}_{ik}(n)}| > 100 \end{cases} \quad (7)$$

We compare the execution time when using element-wise parallelism alone and the case when element-wise parallelism is used in combination with arithmetic parallelism. Results, for both sum-propagation and max-propagation, are shown in Figure 4(a) and Figure 4(b). In all cases, the GPU EP + AP + CM outperforms all the other approaches.[5]

6.4 GPU Optimization: Clique Merging

Clique merging is based on the observation that many junction trees mostly consist of small cliques. This lack of parallelism opportunity will hinder the efficient

[5]The GPU EP + CM combination is included for completeness, but as expected it often performs very poorly. The reason for this is that CM, by merging cliques, creates larger mapping tables that EP is not equipped to handle.

use of the available computing resources, since a single message passing will not be able to occupy the GPU. Merging neighboring small cliques, found in the *S*-*S*-*S* pattern, can help to increase the average size of separators and cliques. Clique merging also reduces the total number of nodes in the junction tree, which in turn reduces the invocation overhead.

We use two parameters to determine which cliques should be merged–one is τ_s, the threshold for separators' potential table size and the other is τ_μ, the threshold for the size of the index mapping table. These parameters are set manually in this paper, however in a companion paper [21] this parameter optimization process is automated by means of machine learning.

In the experiments, we used both arithmetic parallelism and element-wise parallelism. This experiment presents how much extra speedup can be obtained by using clique merging and arithmetic parallelism. The experimental results can be found in Table 1, in the rows showing the GPU execution times for both sum-propagation and max-propagation. In junction trees *Pigs*, *Munin2*, *Munin3* and *Munin4*, a considerable fraction of cliques (and consequently, separators) are small, in other words the *S*-*S*-*S* pattern is common. By merging cliques, we can significantly increase the average separators' and cliques' potential size and thus provide more parallelism.

Comparing GPU EP + AP with GPU EP + AP + CM, the speedup for these junction trees ranged from 10% to 36% when clique merging is used. However, in junction trees *Mildew*, *Water*, *Barley* and *Diabetes*, clique merging does not help much since they mainly consist of large cliques to start with.

Another set of experiments were performed with junction trees that represent an electrical power system ADAPT [16–18]. These junction trees contain many small cliques, due to their underlying BNs being relatively sparsely connected.[6] The experimental results are shown in Figure 5. Using clique merging, the GPU execution times are shortened by 30%-50% for these BNs compared to not using clique merging.

6.5 Performance Comparison: CPU

As a baseline, we implemented a sequential program on an Intel Core 2 Quad CPU with 8MB cache and a 2.5GHz clock. The execution time of the program is comparable to that of GeNie/SMILE, a widely used C++ software package for BN inference.[7] We do not directly use GeNie/SMILE as the baseline here, because we do not know the implementation details of GeNie/SMILE.

In Table 1, the bottom six rows give the execution time comparison for our CPU/GPU hybrid versus a traditional CPU implementation. The CPU/GPU hybrid uses arithmetic parallelism, element-wise parallelism and clique merging. The obtained speedup for sum-propagation ranges from 1.82x to 11.94x, with an arithmetic average of 5.44x and a geometric average of 4.42.

The speedup for max-propagation is similar to, but different from sum-propagation in non-trivial ways. The performance is an overall effect of many factors such as parallelism, memory latency, kernel invocation overhead, etc. Those factors, in turn, are closely correlated with the underlying structures of the junction trees. The speedup for max-propagation ranges from 1.64x to 10.99x, with an arithmetic average of 5.51x and a geometric average of 4.61x.

6.6 Performance Comparison: Previous GPU

We now compare the GPU EP + AP + CM technique introduced in this paper with our previous GPU EP approach [22]. From results in Table 1, compared with the GPU EP approach [22], the arithmetic average cross platform speedup increases from 3.38x (or 338%) to 5.44x (or 544%) for sum-propagation. For max-propagation[8] the speedup increases from 3.22x (or 322%) to 5.51x (or 551%).

7 CONCLUSION AND FUTURE WORK

In this paper, we identified small separators as bottlenecks for parallel computing in junction trees and developed a novel two-dimensional parallel approach for belief propagation over junction trees. We enhanced these two dimensions of parallelism by careful clique merging in order to make better use of the parallel computing resources of a given platform.

In experiments with a CUDA implementation on an NVIDIA GeForce GTX460 GPU, we explored how the performance of our approach varies with different junction trees from applications and how clique merging can improve the performance for junction trees that contains many small cliques. For sum-propagation, the average speedup is 5.44x and the maximum speedup is 11.94x. The average speedup for max-propagation is 5.51x while the maximum speedup is 10.99x.

In the future, we would like to see research on parameter optimization for both clique merging and message

[6] `http://works.bepress.com/ole_mengshoel/`
[7] `http://genie.sis.pitt.edu/`
[8] We implemented max-propagation based on the approach developed previously [22].

passing. It would be useful to automatically change the merging parameters for different junction trees based on the size distribution of the cliques and separators. In addition, we also want to automatically change the kernel running parameters for each single message passing according to the size of a message. In fact, we have already made progress along these lines, taking a machine learning approach [21].

Acknowledgments

This material is based, in part, upon work supported by NSF awards CCF0937044 and ECCS0931978.

References

[1] R. Bekkerman, M. Bilenko, and J. Langford, editors. *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[2] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[3] J. Gonzalez, Y. Low, C. Guestrin, and D. O'Hallaron. Distributed parallel inference on large factor graphs. In *Proc. of the 25th Conference in Uncertainty in Artificial Intelligence (UAI-09)*, 2009.

[4] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, (3):225–263, 1994.

[5] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic network by local computations. *Computational Statistics*, pages 269–282, 1990.

[6] H. Jeon, Y. Xia, and V. K. Prasanna. Parallel exact inference on a CPU-GPGPU heterogenous system. In *Proc. of the 39th International Conference on Parallel Processing*, pages 61–70, 2010.

[7] K. Kask, R. Dechter, and A. Gelfand. BEEM: bucket elimination with external memory. In *Proc. of the 26th Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 268–276, 2010.

[8] D. Koller and N. Friedman, editors. *Probablistic graphic model: principle and techniques*. The MIT Press, 2009.

[9] A. V. Kozlov and J. P. Singh. A parallel Lauritzen-Spiegelhalter algorithm for probabilistic inference. In *Proc. of the 1994 ACM/IEEE conference on Supercomputing*, pages 320–329, 1994.

[10] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[11] M. D. Linderman, R. Bruggner, V. Athalye, T. H. Meng, N. B. Asadi, and G. P. Nolan. High-throughput Bayesian network learning using heterogeneous multicore computers. In *Proc. of the 24th ACM International Conference on Supercomputing*, pages 95–104, 2010.

[12] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. Hellerstein. GraphLab: A new framework for parallel machine learning. In *Proc. of the 26th Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 340–349, 2010.

[13] O. J. Mengshoel. Understanding the scalability of Bayesian network inference using clique tree growth curves. *Artificial Intelligence*, 174:984–1006, 2010.

[14] V. K. Namasivayam and V. K. Prasanna. Scalable parallel implementation of exact inference in Bayesian networks. In *Proc. of the 12th International Conference on Parallel and Distributed System*, pages 143–150, 2006.

[15] D. Nilsson. An efficient algorithm for finding the m most probable configurations in probablistic expert system. *Statistics and Computing*, pages 159–173, 1998.

[16] B. W. Ricks and O. J. Mengshoel. The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems. In *Proc. of the 20th International Workshop on Principles of Diagnosis (DX-09)*, pages 415–422, Stockholm, Sweden, 2009.

[17] B. W. Ricks and O. J. Mengshoel. Methods for probabilistic fault diagnosis: An electrical power system case study. In *Proc. of Annual Conference of the PHM Society, 2009 (PHM-09)*, San Diego, CA, 2009.

[18] B. W. Ricks and O. J. Mengshoel. Diagnosing intermittent and persistent faults using static bayesian networks. In *Proc. of the 21st International Workshop on Principles of Diagnosis (DX-10)*, Portland, OR, 2010.

[19] M. Silberstein, A. Schuster, D. Geiger, A. Patney, and J. D. Owens. Efficient computation of sum-products on GPUs through software-managed cache. In *Proc. of the 22nd ACM International Conference on Supercomputing*, pages 309–318, 2008.

[20] Y. Xia and V. K. Prasanna. Node level primitives for parallel exact inference. In *Proc. of the 19th International Symposium on Computer Architechture and High Performance Computing*, pages 221–228, 2007.

[21] L. Zheng and O. J. Mengshoel. Optimizing parallel belief propagation in junction trees using regression. In *Proc. of 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-13)*, Chicago, IL, August 2013.

[22] L. Zheng, O. J. Mengshoel, and J. Chong. Belief propagation by message passing in junction trees: Computing each message faster using GPU parallelization. In *Proc. of the 27th Conference in Uncertainty in Artificial Intelligence (UAI-11)*, pages 822–830, Barcelona, Spain, 2011.

Author Index