Preface

We were pleased to present this CEUR-WS volume, the Proceedings of the 9th Bayesian Modeling Applications Workshop (BMAW-12), held in Catalina Island, California, USA, August 18th, 2012, as a workshop of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012).

Bayesian networks are now a powerful, well-established technology for reasoning under uncertainty, supported by a wide range of mature academic and commercial software tools. They are now being applied in many domains, including environmental and ecological modeling, bioinformatics, medical decision support, many types of engineering, robotics, military, financial and economic modeling, education, forensics, emergency response, surveillance, and so on. This workshop solicited submissions describing such real world applications, whether as stand-alone BNs or where the BNs are embedded in a larger software system. We suggested authors address the practical issues involved in developing real-world applications, such as knowledge engineering methodologies, elicitation techniques, defining and meeting client needs, validation processes and integration methods, as well as software tools to these support these activities.

This year we encouraged the submission of papers addressing the workshop theme of *Temporal Modeling*. Recently communities building dynamic Bayes networks (DBNs) and partially observable MDPs (POMDPs) are coming to realize that they are applying their methods to identical applications. Similarly POMDPs and other probabilistic methods are now established in the field of Automated Planning. Stochastic process models such as continuous time Bayes networks (CTBNs) should also be considered as part of this trend. Adaptive and on-line learning models also fit into this focus.

This year all submissions were full length papers peer-reviewed by at least two reviewers. Of the 18 submissions, 11 papers were accepted for oral presentation, with 10 to appear in these proceedings. The papers include an interesting range of methods, models and applications. More specifically, the proceedings contain 2 papers on dynamic fusion, with the applications being goal-based people tracking, and maritime domain awareness; two papers on dynamic topic models; a number of papers looking at discrete stage models, with applications from feedback control, oil drilling, analyzing HIV mutations and forecasting kindergarten student reading; and papers on spatial-temporal models, applied to environmental management and search query and data centre logs.

The presentations were accompanied by questions and discussions throughout, plus authors also attended a poster and demonstration session, which provided an opportunity for more detailed discussion and networking. In addition, this year the workshop concluded with a panel and session on the emerging trends in applications modelling and planning for future workshops.

We are grateful to all the program committee members for their excellent work in refereeing the papers within a very tight schedule. We also appreciate the organizational support provided by the main UAI conference. In particular, we thank the UAI 2012 conference general chair, Fabio Cozman, the workshop chair, Ruslan Salakhutdinov and the local arrangements chair, David Heckerman.

John Mark Agosta, Ann Nicholson, M. Julia Flores
Workshop Co-Chairs
August 2012

Program Committee

John Mark Agosta	Toyota
Concha Bielza	Technical University of Madrid, Spain
Dennis Buede	Innovative Decisions, Inc., USA
Javier Diez	UNED, Spain
Marek Druzdzel	U. of Pittsburgh, USA & Bialystok U. of Technology, Poland
Julia Flores	University of Castilla-La Mancha, Spain
Asela Gunawardana	Microsoft Research, USA
José Gámez	University of Castilla-La Mancha, Spain
Oscar Kipersztok	Boeing Aircraft, USA
Branislav Kveton	Technicolor Labs, USA
Helge Langseth	Norwegian University of Science and Technology, Norway
Pedro Larrañaga	Technical University of Madrid, Spain
Philippe Leray	COD/LINA Nantes University, France
Michael Mahoney	Stanford University, USA
Suzanne Mahoney	Innovative Decisions Inc, USA
Chris Meek	Microsoft Research, USA
Ole Mengshoel	Carnegie Mellon University, USA
Serafín Moral	University of Granada, Spain
Ann Nicholson	Monash University & Bayesian Intelligence, Australia
Jens Nielsen	Sheffield University, UK
Thomas Nielson	Department of Computer Science, Aalborg University, Denmark
Jose Peña	Linköping University, Sweden
Pascal Poupart	University of Waterloo, Canada
Antonio Salmerón	University of Almería, Spain
Fabio Stella	Universita' degli Studi di Milano-Bicocca, Milano, Italy
Enrique Sucar	INAOE, Mexico
Georgios Theocharous	Intel, USA

Table of Contents

Using POMDPs to Forecast Kindergarten Students Reading Comprehension

Russell G. Almond
Educational Psychology and Lerning Systems
Flordia State University
Tallahassee, FL 32306
ralmond@fsu.edu

Umit Tokac
Educational Psychology and Lerning Systems
Flordia State University
Tallahassee, FL 32306
ut08@my.fsu.edu

Stephanie Al Otaiba*
Department of Teaching and Learning
Southern Methodist University
Dallas, TX 75275
salotaiba@smu.edu

Abstract

Summative assessment of student abilities typically comes at the end of the instructional period, too late for educators to use the information for planning instruction. This paper explores the possibility of using Hierarchical Linear Models to forecast students end of year performance. Because these models are closely related to partially observed Markov decision processes (POMDPs), these should support extensions to instructional planning to meet educational goals. Despite the new notation, the POMDP models are subject to a familiar problem from the educational context: scale identifiability. This paper describes how this problem manifests itself and looks at one potential solution.

1 INTRODUCTION

There is a long tradition in education of separating instruction and assessment: summative assessment of what a student learns comes at the end of the unit/semester/year. As limited time is allocated for assessment, such assessments are typically limited in their reliability (accuracy of measurement) and content validity (coverage of the targeted knowledge, skills and ability). Because summative assessment comes at the end of instructions, instructors are not able to make changes to their instructions to maximize student learning (Almond, 2010).

Bennett (2007) suggested breaking the summative assessment into four or six periodic assessments. First, spreading the cost (student time taken away from direct instruction) over multiple measurement occasions allows for longer testing providing both greater content coverage and reliability. Moreover, a proper model for student growth allows forecasting of the students eventual status at the end of the year. Consequently, teachers and administrators can form plans for students which maximize learning outcomes and identify students for whom the goals are unreachable for special instruction. In this sense, the periodic assessments play a role somewhere between traditional summative assessment and formative assessment — assessing student learning for the purpose of improving instruction (Black & Wiliam, 1998; Wiggins, 1998; Pelligrino, Glaser, & Chudowsky, 2001).

Almond (2007) noted that the forecasting could be done using a partially observed Markov decision process (POMDP; Boutilier, Dean, & Hanks, 1999): the latent variables describing student proficiency form an unobserved Markov process, and the periodic assessments provide observable evidence about the state of those latent variables. The instructional activities chosen between time points are the measurement space, and in fact, the students response to instruction often provides important clues about their proficiency and specific learning problems (Marcotte & Hintze, 2009). Almond (2009) notes the similarity between POMDPs and other frameworks more commonly used in education, such as latent growth modeling (Singer & Willett, 2003) and hierarchical linear modeling (HLM; Raudenbush & Byrk, 2002). The principle difference is one of emphasis: in the POMDP framework, the emphasis is usually on estimating the individuals latent state for the purpose of planning. In the HLM and multilevel growth model, the emphasis is usually on estimating the effectiveness of various activities. This paper looks at the problem of forecasting using HLM models both directly and through conversion to POMDP parameterizations.

The purpose of our study is to try to fit a POMDP-based latent growth model using Bayesian methods to a set of data documenting the development of Reading skills in a number of Kindergarten students. Once the

*Some of the work took place while she was at the Florida Center for Reading Research, Tallahassee, FL

model is successfully fit, we will use it to predict the end-of-year status of the students.

2 THE DATA

This study uses longitudinal data about reading development originally collected by the Florida Center for Reading Research (Al Otaiba et al., 2011). The reading skills for this initial cohort of students was measured three times (Fall, Winter and Spring) during Kindergarten, and follow-up measurements were taken at the end of 1st, 2nd and 3rd grade. There were 247 students in the initial sample, but only 224 were still in the area at the end of the first year.

During Kindergarten, children rapidly develop in Reading and pre-Reading skills (e.g., oral vocabulary and letter identification). Consequently, not all measures are appropriate for all time points. Consequently, different measures were collected at different time points. Table 1 shows the measures that were collected during Kindergarten:

Table 1: Measures Collected By Occasion

Measure	Fall	Winter	Spring
LW	X	X	X
PV	X	X	X
ISF	X	X	
PSF		X	X
NWF		X	X
LNF	X	X	X

The measures are taken from the Woodcock-Johnson III Cognitive Test (WJ-III; Woodcock, McGrew, & Mather, 2001) and the Dynamic Indicators of Basic Early Literacy Skills (DIBELS; Good & Kaminkski, 2002). The measures used were:

LW – Letter-Word Identification (WJ-III)

PV – Picture Vocabulary (WJ-III)

LNF – Letter Naming Fluency (DIBELS)

ISF – Initial Sound Fluency (DIBELS)

PSF – Phoneme Segmentation Fluency (DIBELS)

NWF – Nonsense Word Fluency (DIBELS)

The Woodcock-Johnson measures are available in several forms. We used the "W" scale (which is scaled to an item response theory model), as it showed more variation than the scale scores.

Additionally, teacher and school identifiers are available for each child. For this cohort teachers were not given special instructions nor a prescribed curriculum, although most of them used the same curriculum.

3 THE POMDP FRAMEWORK

Almond (2007) provides a generalized model for how a POMDP can represent measurement of a developing proficiency across multiple time points (Figure 1).

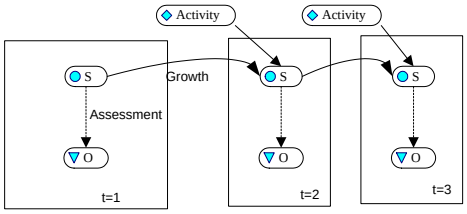


Figure 1: Measurement across time as POMDP

In this figure, the nodes marked **S** represent the latent student proficiency as it evolves over time. At each time slice, there is generally some kind of measurement of student progress represented by the observable outcomes **O**. Note that these may be different for different time slices (c.f., Table 1). Following the terminology of evidence-centered assessment design (ECD; Mislevy, Steinberg, & Almond, 2003) we call this an *evidence model*. In general, both the proficiency variables at Measurement Occasion m, $\mathbf{S}_m$, and the observable outcome variables on that occasion, $\mathbf{O}_m$ are multivariate.

Extending the ECD terminology, Almond (2007) calls the model for the $\mathbf{S}_m$'s, the *proficiency growth model*. Following the normal logic of POMDPs this is expressed with two parts: the first is the initial proficiency model, which gives the population distribution for proficiency at the first measurement occasion. The second is an action model, which gives a probability distribution for change in proficiency over time that depends on the instructional activity chosen between measurement occasions.

3.1 PROFICIENCY GROWTH MODEL

For the data from the Al Otaiba et al. (2011) study, the latent proficiency is obviously Reading. The question immediately arises as to how many dimensions to use to represent the reading construct. As the students are entering the study in Kindergarten, components of the reading skill, such as oral vocabulary and phonemic awareness are less tightly correlated than they are with older children. (In the fall of the Kindergarten year the correlation between the LW and PV scores in the Al Otaiba et al. study was $r = .46$, $n = 247$, while in the spring it had increased to $r = .56$, $n = 224$.) As

a starting point, we will fit a unidimensional model of Reading, representing it with a single continuous variable: R_{nm} the reading ability of Individual n on Measurement Occasion m.

3.1.1 Model for Growth

In the first cohort of the Al Otaiba et al. (2011) study, teachers were not given specific instructions about curriculum or activity between the time points. We therefore do not have a dependency on activity to measure here. However we do expect there to be some classroom-to-classroom differences, so we will make the growth parameters dependent on the classroom (The teacher effect is part of the classroom effect, however aspects of the peer group and environment are captured as well). Let $c(n)$ be the classroom to which Student n belongs. Note also that classrooms are nested within schools, so school effects are considered part of the general classroom effect.

Following this logic, for Measurement Occasion $m > 1$, define:

$$\begin{aligned} R_{nm} &= R_{n(m-1)} + (\gamma_{c(n)m} + \gamma_{0m})\Delta T_{nm} + \eta_{nm} \\ \eta_{nm} &\sim \mathcal{N}(0, \sigma_{c(n)m}\sqrt{\Delta T_{nm}}) \end{aligned} \tag{1}$$

Here ΔT_{nm} is the time between Measurement Occasions m and $m-1$ for individual n. Here γ_{0m} is an average growth rate, and γ_{cm} is a classroom specific growth rate. Note that the residual standard deviation depends on both a classroom specific rate, σ_{cm}, and the time elapsed between measurements. This is consistent with the model that student ability is growing according to a nonstationairy Wiener (Brownian motion) process.

3.1.2 Model For Initial Proficiency

Children entering Kindergarten have very diverse language and early literacy backgrounds. There are considerable differences in the amount of experience with print material the child experiences at home, breadth and depth of vocabulary used with the child, as well as a wide variety of preschool experiences. As a child's preschool and early home experiences are at least partially dependent on their parents' social and economic status, and within-school socio-economic status tends to be more homogeneous than across school status, we model the initial status as dependent on the school. Let $s(n)$ be the school attended (during Kindergarten) for Student n.

There is also a considerable variation in the age at entry. In the Al Otaiba et al. (2011) study, 95% of the children were between the ages of 5 years 2 months and 6 years 4 months at the time of the first testing (with a few students 7 years or older). This represents a considerable variation in maturity, and potentially in initial ability.

We define the following model for Measurement Occasion 1:

$$R_{n1} \sim \mathcal{N}(\mu_{s(n)}, \upsilon_{s(n)}) \tag{2}$$

3.2 EVIDENCE MODELS

Because we are assuming that Reading proficiency is unidimensional, we do not need to specify which of the measures in Table 1 are relevant to which proficiencies. Thus, the evidence model is a collection of simple regressions, for each observation Y_{nmi} for Individual n at Measurement Occasion m on Instrument i, we have:

$$Y_{nmi} = a_i + b_i R_{nm} + \epsilon_{nmi} \tag{3}$$

$$\epsilon_{nmi} \sim \mathcal{N}(0, \omega_i) \tag{4}$$

Note that the slope parameter b_i actually encodes a relative importance for the various measures.

One advantage of this structure is that we we do not need to explicitly specify the data collection structure (Table 1). Instead, we can simply set the values of measures not recorded in each wave to missing values.

Because each of the instruments are well established (Woodcock et al., 2001; Good & Kaminkski, 2002), we know some of their critical psychometric properties. In particular, the *reliability* of Instrument i, ρ_i is documented in the handbooks for the measures. In classical test theory, the reliability is the squared correlation between the true score of an examinee and the observed score. With a bit of algebra, this definition is is equivalent to:

$$\rho_i = 1 - \frac{\mathrm{Var}_n(\epsilon_{nmi})}{\mathrm{Var}_n(Y_{nmi})} \ . \tag{5}$$

Here the notation $\mathrm{Var}_n(\cdot)$ indicates that the variance is taken over individuals (with measurement occasion and instrument held constant). Solving Equation 5 for $\mathrm{Var}_n(\epsilon_{nmi})$ yields an estimate for ω_i^2 for each measurement occasion. We took the median of the three estimates as our base estimate for ω_i^2, $\tilde{\omega}_i$.

One drawback of the classical test theory concept of reliability is that it is dependent on the population being measured. Thus, as the sample in the Al Otaiba et al. (2011) is slightly different from the norming samples used in the development of the WJ-III and DIBELS measures, we expect our observed reliability will differ slightly from the published values. What we do is set up priors for ω_i using $\tilde{\omega}_i$ as the prior mean. In particular,

$$1/\omega_i^2 \sim \mathrm{Gamma}(\alpha, \alpha\tilde{\omega}_i^2) \ , \tag{6}$$

where Gamma(α, β) is a gamma distribution with shape parameter α and rate parameter β. We note that any gamma distribution with $\beta = \alpha\tilde{\omega}_i^2$ will have the proper mean. The shape parameter α is then effectively a tuning parameter giving the strength the prior distribution, or equivalently the relative weight of the published reliabilities and the observed error distribution. We initially chose a value of $\alpha = 100$ weights the prior knowledge as equivalent to 100 observations, but later increased it to 1000 when we were experiencing convergence problems.

3.3 SCALE IDENTIFICATION

A problem that frequently arises in educational models using latent variables is the identifiability of the scale. In particular, suppose we replaced R_{nmi} with $R'_{nmi} = R_{nmi} + c$ for an arbitrary constant c, and replaced a_i with $a'_i = a_i - b_i c$. The likelihood of the observed data Y_{nmi} (implicit in Equation 3) would be identical. A similar problem arises if we replace R_{nmi} with $R''_{nmi} = cR_{nmi}$ and b_i with $b''_i = b_i/c$. Additional constraints must be added to the model to identify the scale and location of the latent variable R.

A frequently used convention in psychometrics is to identify the scale and location of the latent variable by assuming that the population mean and variance for the latent variable is 0 and 1 (i.e., that the latent variable has an approximately unit normal distribution). In this case we can identify the scale for R_{n1} by constraining $\sum_s \mu_s = 0$ and $\frac{1}{S}\sum_s \upsilon_s = 1$, where S is the total number of schools in the study.

Because this is a temporal model, there exists another complication. We need to identify the scale of R_{nm} for $m > 1$. In particular, the mean and variance of the innovations γ_{0m} and σ_{tm} can cause similar identifiability to the scale and location for R_{nm} that the initial mean and variance caused for R_{nm}. In this case we apply a different solution. We assume that the properties of the instruments, and their relationships to the latent reading proficiency do not vary across time (at least for the time points they are in use). Note that in Equation 3, the slope, b_i and intercept, a_i do not vary across time. This establishes a common scale for all time points.

Our initial thinking was that this would be enough to identify the model. Unfortunately, because of the structural missing data additional constraints are needed. These are described below.

Bafumi, Gelman, Park, and Kaplan (2005) present a different approach to enforcing identifiability. They let the model be unidentified while fitting the data, but then transform the estimates when evaluating the data (i.e., they enforce the constraint by manipulating the samples in R and coda R Development Core Team, 2007; Plummer, Best, Cowles, & Vines, 2006 rather than in BUGS or JAGS). For example, rather than constraining $\sum \mu_s = 0$, they would estimate μ_s freely, but post hoc would adjust the sample from the rth cycle, $\mu_s^{(r)'} = \mu_s^{(r)} - \sum \mu_s^{(r)}$, making appropriate adjustments to the other parameters. They claim that the resulting model mixes better, however, there is some difficulty in figuring out how the post hoc adjustments will affect other parameters in the model.

4 PROBLEMS WITH MODEL FITTING

We attempted to fit the model described in the previous section with MCMC using JAGS (Plummer, 2012).[1] After some initial difficulties we removed the teacher and school effects (intending to add them again after we fit the simpler model). This also allowed us to restrict the prior distribution for R_{n1} to be a unit normal distribution (zero mean, variance one). This is a common identifiability constraint imposed in psychometric models.

4.1 FIVE MEASURE MODEL

Our initial experiments involved five of the six measures (the PSF measure was left out due to a mistake in the model setup). We ran three Markov chains using random starting positions and found that the models did not converge. Or more properly, the evidence model parameters (a_i, b_i, and ω_i) for the DIBELS NWF (nonsense word fluency) measure did not converge. Table 2 shows the posterior mean of the evidence model parameters for the five measures (because the MCMC chain did not reach the stationary state, this may not be the true posterior).

Note in Table 2 that the estimated residual variance is extremely low, indicating a nearly perfect correlation between the latent Reading variable and the NWF measure. In this case, the MCMC chain looks like it is somehow using that measure to identify the scale of the latent variable. Furthermore, the slope for that variables is twice as high as the slope for other variables in

[1] Actually, we did some of our early model fitting using WinBUGS (D. J. Lunn, Thomas, Best, & Spiegelhalter, 2000). Some of the identification problems we were having in WinBUGS we are not having in JAGS. JAGS may be using slightly better samplers which may take care of issues that occur when the predictor variables in regressions are not centered (Plummer, 2012). Similar improvements may have been made in OpenBUGS (D. Lunn, Spiegelhalter, Thomas, & Best, 2009), the successor to WinBUGS, but we have not tested this model using OpenBUGS.

Table 2: Evidence Model Parameters, 5 Measure Model

	LW	PV	LNF	ISF	NWF
a	105.37	99.90	25.76	13.97	-4.27*
b	0.15	0.05	0.49	0.32	0.87*
ω	6.15	4.92	6.31	4.38	0.09

* indicates parameter did not converge

the model. Table 3 shows some of the difficulty. The NWF measure is the only one showing a large increase between the Winter and Spring testing periods. So naturally, there is a tendency to track that measure.

Table 3: Mean Scores on Each Measure at Each Administration

	LW	PV	LNF	ISF	NWF
Fall	108.5	100.6	27.3	14.2	
Winter	110.8	102.3	42.9	25.5	27.9
Spring	111.2	101.7	51.3		43.2

Trace plots of the evidence models show the problem. Figure 2 shows an example of extremely slow mixing, that is characteristic of identifiability problems. Depending on the values of the other variables in the system (particularly the latent reading variables) higher or lower slopes may be sensible. Looking at the trace plots of R_{nm} for several students show similar poor mixing for $m > 1$. We would expect similar problems with the trace plots for γ_{0m}, but the mixing looks good on those chains.

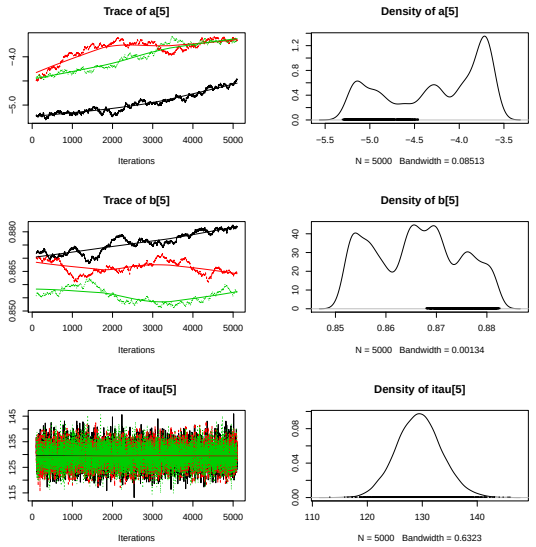


Figure 2: Trace plots of evidence model parameters for measure NWF

It is likely that the problem is some complex interaction between using γ_{0m} and the b's to identify mean growth, or the a's which define the starting point for growth. Note that the problematic measure, NWF, was not measured at the first time point. Thus the constraint on the distribution of R_{n0} will not define its scale in the second or third measurement occasions.

4.2 THREE MEASURE MODEL

As the problematic measure may be the ones which were not recored at all three time points, we ran the model again, dropping the ISF and NWF measures (the ones not observed at the first or third measurement occasion). The new model also did not converge, although the focus of the problem has now moved from the NWF measure to the LNF measure.

Table 4 shows the new estimates from the unconverged posterior. Again, the variance for the measure that did not converge is substantially smaller than that of the other measures, and the slope is substantially higher. Again the trace plots (Figure 3) show poor mixing, as do similar plots for the R_{nm} measures for $m > 1$. There is also an indication of a trend that indicates that the chains have not covered the whole of the posterior distribution.

Table 4: Evidence Model Parameters, 3 Measure Model

	LW	PV	LNF
a	103.74	98.95	23.02*
b	1.59	0.64	4.55*
ω	5.55	4.78	0.11

* indicates parameter did not converge

4.3 MISSING IDENTIFICATION CONSTRAINT

Looking back to the problems in the model fit in Section 4.1, note that the lack of fit could be explained by the interaction between a_5 (the intercept for the NWF measure) and γ_1 (the average proficiency change between the first and second time points). As NWF is not measured in the first time point, any arbitrary change between the first and second time point can be created by changing a_5, b_5 and γ_1. The other four measures were all collected in the fall, so in these cases, a_i should have been fixed by the constraint that $E[R_{n1}] = 0$.

What is required is a method for fixing the value of a_i for measures that were not collected at the initial time point. One possible way to do this would be to simply set $a_i = 0$. This is not unreasonable, if all of

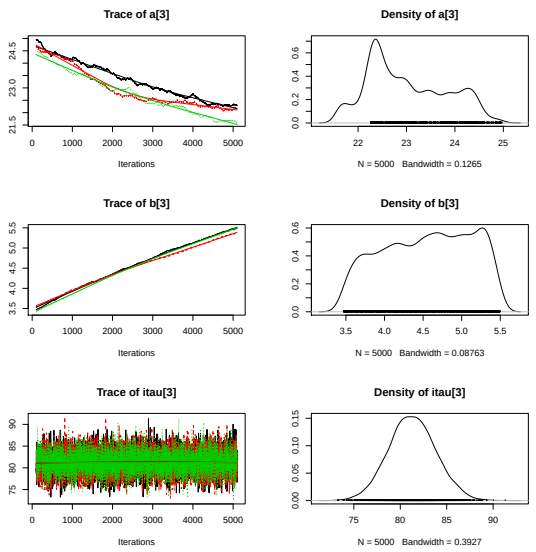


Figure 3: Trace plots of evidence model parameters for measure LNF, Three Measure Model

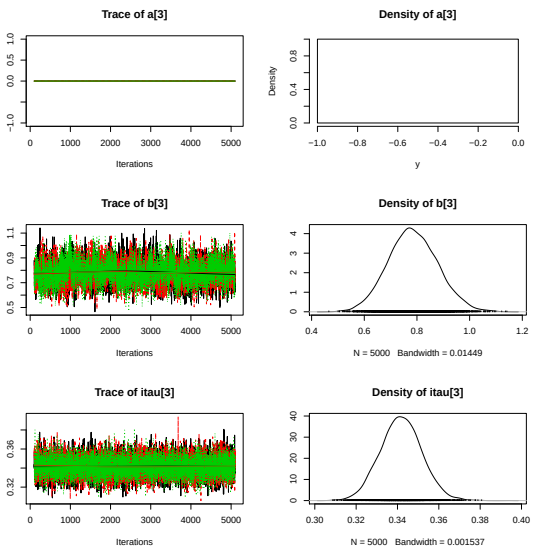


Figure 4: Trace plots of evidence model parameters for measure NWF, Six Measure Model with $a_i = 0$

the variables are on a standardized scale: it implies that the average trajectory of the average student will pass through the average of the scores.

This required that the scores all be on the same scale (especially problematic with the WJ-III and DIBELS scores based on different development and norming sample. Fortunately, for these data all six measures were collected in the winter time period. Subtracting the mean of the Winter scores and dividing by the standard deviation for each measure produced standardized scores. This standardization together with the constraint $a_i = 0$ caused the models to converge.

4.4 SIX MEASURE MODEL

Using the standardized data and the additional constraint of $a_1 = 0$, we again fit the model using MCMC. This time, we got convergence on all of the evidence model parameters (Figure 4).

Table 5 shows the mean of the latent Reading variable for the first five students in the sample. This appears to be well behaved with all of the students showing growth across the three time points.

Table 5: Mean values for Reading for first five students, Six Measure model with $a_i = 0$

	S1	S2	S3	S4	S5
F	-0.394	-0.351	-0.375	-1.556	-0.773
W	0.029	0.104	0.031	-1.278	-0.415
S	0.864	1.016	0.852	-0.514	0.419

5 FUTURE DIRECTIONS AND CHALLENGES

The key to getting this model to converge was the standardization of the measure scales. Fortunately, this data set had a time period where all six measures were applied to the same population. Consequently, standardizing the scales at this time point put the measures on a comparable scale, which then made the fixed intercept constraint meaningful.

It is difficult to see how this generalizes to cases in which there is not a single time point in which all measures are collected. This is a problem with the cohort examined in this study when we look at the data gathered in first and second grades. As the students reading abilities develop, new and more difficult measures of reading become appropriate. Linking these back to the old scale is a difficult problem. This problem is well known in the educational literature under the name "vertical scaling" (von Davier, Carstensen and von Davier, 2006, provide a review of the literature).

Now that the model without teacher or school effects converges, the next step is to add those back into the model. Also, we should use cross-validation to evaluate how well the model predicts students scores. The Al Otaiba et al. (2011) data set has long term follow-up for a substantial portion of the students, so we can see how well the model can predict First and Second grade reading scores as well. Finally, we can look at the rules for classification in to special instruction, to see whether integrating the data across multiple measures

provides a better picture of the student than looking at one measure alone.

Acknowledgments

We would like to thank the Florida Center for Reading Research for allowing us access to the data used in this paper. The data were originally collected as part of a larger National Institute of Child Health and Human Development Early Child Care Research Network study.

References

Almond, R. G. (2007). Cognitive modeling to represent growth (learning) using Markov decision processes. *Technology, Instruction, Cognition and Learning (TICL)*, *5*, 313–324. Available from `http://www.oldcitypublishing.com/TICL/TICL.html`

Almond, R. G. (2009). *Estimating parameters of periodic assessment models* (Research Report No. To appear). Educational Testing Service.

Almond, R. G. (2010). Using evidence centered design to think about assessments. In V. J. Shute & B. J. Becker (Eds.), *Innovative assessment for the 21st century: Supporting educational needs.* (pp. 75–100). Springer.

Al Otaiba, S., Folsom, J. S., Schatschnneider, C., Wanzek, J., Greulich, L., Meadows, J., et al. (2011). Predicting first-grade reading performance from kindergarten response to tier 1 instruction. *Exceptional Children*, *77*(4), 453–470.

Bafumi, J., Gelman, A., Park, D. K., & Kaplan, N. (2005). Practial issues in implementing and understanding bayesian ideal point estimation. *Political Analysis*, *13*, 171-187.

Bennett, R. E. (2007, May). *Assessment of, for, and as learning: Can we have all three?* Paper presented at the Institute of Educational Assessors National Conference, London, England. Available from `http://www.ioea.org.uk/Home/news_and_events/annual_conference/day1/randy_bennett.aspx`

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy, and Practice*, *5*(1), 7–74.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1-94. Available from `citeseer.ist.psu.edu/boutilier99decisiontheoretic.html`

Good, R. H., & Kaminkski, R. A. (Eds.). (2002). Dynamic indicators of basic early literacy skills (6th ed.) [Computer software manual]. Available from `https://dibels.uoregon.edu/`

Lunn, D., Spiegelhalter, D., Thomas, A., & Best, N. (2009). The BUGS project: Evolution, critique and future directions (with discussion). *Statistics in Medicine*, *28*, 3049–3082.

Lunn, D. J., Thomas, A., Best, N., & Spiegelhalter, D. (2000). WinBUGS – a Bayesian modeling framework: concepts, structure, and extensibility. *Statistics and Computing*, *10*, 325–337.

Marcotte, A. M., & Hintze, J. M. (2009). Incremental and predictive utility of formative assessment methods of reading comprehension. *Journal of School Psychology*, *47*, 315-335.

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.

Pelligrino, J., Glaser, R., & Chudowsky, N. (Eds.). (2001). *Knowing what students know: The science and design of educational assessment.* National Research Council.

Plummer, M. (2012, May). JAGS version 3.2.0 user manual (3.2.0 ed.) [Computer software manual]. Available from `http://mcmc-jags.sourceforge.net/`

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). coda: Output analysis and diagnostics for MCMC [Computer software manual]. (R package version 0.10-7)

R Development Core Team. (2007). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Available from `http://www.R-project.org`

Raudenbush, S. W., & Byrk, A. S. (2002). *Hierarchical linear models* (second edition ed.). Sage Publications.

Singer, J. D., & Willett, J. B. (2003). *Applied longitudinal data analysis: Modeling change and event occurrence* (1st ed.). Oxford University Press, USA.

von Davier, M., Carstensen, C. H., & von Davier, A. A. (2006). *Linking competencies in educational settings and measuring growth* (Research Report No. RR-06-12). ETS.

Wiggins, G. P. (1998). *Educative assessment: Designing assessments to inform and improve student performance.* Jossey-Bass.

Woodcock, R. W., McGrew, K. S., & Mather, N. (2001). Wj-iii tests of cognitive abilities and achievement [Computer software manual].

High-Level Information Fusion with Bayesian Semantics

Paulo C. G. Costa, Kathryn Laskey, Kuo-Chu Chang, Wei Sun, Cheol Park, Shou Matsumoto
Center of Excellence in C4I & Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030
(pcosta,klaskey,kchang)@gmu.edu; wsun@c4i.gmu.edu; cparkf@gmu.edu; smatsum2@masonlive.gmu.edu

Abstract

In an increasingly interconnected world information comes from various sources, usually with distinct, sometimes inconsistent semantics. Transforming raw data into high-level information fusion (HLIF) products, such as situation displays, automated decision support, and predictive analysis, relies heavily on human cognition. There is a clear lack of automated solutions for HLIF, making such systems prone to scalability issues. In this paper, we propose to address this issue with the use of highly expressive Bayesian models, which can provide a tighter link between information coming from low-level sources and the high-level information fusion systems, and allow for greater automation of the overall process. We illustrate our ideas with a naval HLIF system, and show the results of a preliminary set of experiments.

1 INTRODUCTION

Information fusion is defined as:

> "... the synergistic integration of information from different sources about the behavior of a particular system, to support decisions and actions relating to the system."[1]

A distinction is commonly made between *low-level* and *high-level* fusion. Low-level fusion combines sensor reports to identify, classify, or track individual objects. High-level fusion combines information about multiple objects, as well as contextual information, to characterize a complex situation, draw inferences about the intentions of actors, and support process refinement.

In current information fusion systems, lower-level data fusion is typically accomplished by stove-piped systems that feed information directly to human users. Subsequent generation of high-level information fusion (HLIF) products, such as situation displays, automated decision support, and predictive analysis, relies heavily on human cognition. The tacit underlying assumption is that humans are still the most efficient resource for translating low-level fusion products into decision-relevant knowledge. While the current approach works well for many purposes, it cannot scale as the data influx grows. Automated assistance for HLIF tasks is urgently needed to mitigate cognitive overload and achieve the necessary throughput.

Stove-piped systems can be extremely efficient at exploiting a specific technology applied to a limited and well defined set of problems. Air Traffic Control Systems, for instance, employ radar technology in a very effective way to provide reliable situation awareness for radar controllers via sophisticated low-level information fusion (LLIF) techniques. The synthetic radar screen shown to traffic controllers in a sector of an Area Control Center (ACC) fuses multiple radar tracks. Data association algorithms infer whether geographically close signals captured by various radars are coming from a single or multiple aircraft. Despite the sophistication of its low-level fusion components, the ATC system relies heavily on humans for HLIF products. For instance, controllers rely on their own understanding of the overall picture to decide how to drive their tracks; area coordinators rely on their knowledge to decide whether the outbound traffic to a given airport should be redirected due to an upcoming storm; and so on.

The ATC system is a good example of a highly sophisticated stove-piped system that relies on human cognition for its major purpose: to ensure that thousands of airplanes in the US can share the airspace in a safe and effective way. As the volume of airplanes increases, more humans are needed to perform HLIF tasks. After a point, the overhead of transferring between ever-smaller control sectors becomes a major

[1] The International Society for Information Fusion, http://isif.org

scalability issue. That is, cognitive limitations (each human can control only airplanes at once) together with the added complexity of adding extra cognitive units (a.k.a. traffic controllers) become a major obstacle to growth. This scalability problem is common to HLIF systems in other domains as well.

In this paper, we propose to address the issue with the use of highly expressive Bayesian models. Such systems provide a tighter link between low-level and high-level information fusion systems. Because they are sufficiently expressive to reason about high-level information, they provide a coherent framework for expressing and reasoning with uncertain information that spans low and high level systems.

This paper describes our approach by way of a case study in information fusion for Maritime Domain Awareness. Section 2 motivates the use of explicit probabilistic semantics and explains the main concepts behind our approach. Section 3 introduces the Maritime Domain Ontology we used in our experiments. The experiments are described in Section 4. Section 5 concludes with a discussion.

2 Semantics in HLIF

Humans are more effective than computers at gathering various pieces of information and correlating them into a coherent picture, but still have a high error rate. For example, an intelligence analyst can correlate images and videos of a road with observers reports that a convoy has passed in the early afternoon, and conclude that this was the same convoy that participated in a terrorist attack 10 miles down that road. These conclusions are based on an implicit understanding of how trucks and cars are represented in each type of media (video, imagery, human reports), as well as the temporal and spatial relationships between cars, roads, convoys, etc. For a computer program to perform the same inferences from the same set of sources, it must possess the same kind of knowledge. Conveying such knowledge to a computer program requires a means to make the humans tacit knowledge explicit and formal, so it can be retrieved and used when needed.

Ontologies are the current paradigm for specifying domain knowledge in an explicit and formal way. One of the most cited definitions of ontologies is the specification of a conceptualization. [1] To perform automated fusion in the above example, concepts such as cars, roads, convoys, people, etc., as well as their relationships, must be formalized in a way that computers can store, retrieve, and use. Not surprisingly, ontologies have been widely considered in the domain of information fusion as a means to enable automated systems to perform HLIF tasks (e.g. [2, 3, 4]).

Most languages for expressing ontologies, such as the most popular variant of the W3C Recommendation OWL [5], are based on Description Logic [6]. Other ontology languages such as KIF[2] and the ISO Standard Common Logic[3] , are based on first-order logic. Classical logic has no standard means to represent uncertainty. This is a major drawback for HLIF systems, which must operate in environments in which uncertainty is pervasive. Inputs from LLIF systems come with uncertainty, as do the high-level domain relationships that analysts use to draw inferences about a complex situation.

Although LLIF algorithms often have a basis in probability theory, it is common to report results according to a threshold rule without any confidence qualifier. This is often justified by cognitive limitations of human decision makers. As an example, suppose a video analysis report assigns 86% probability of person Joe being inside a car driving towards place A. If the threshold for the input source was 85%, a LLIF system might simply report the statement without qualification, and a HLIF system might treat this as a true statement. Such threshold rules lose uncertainty information. Other information sources, each with its own internal processing and threshold rules, might provide additional reports about Joe, A, and other aspects of the situation relevant to inferences about Joes destination. Without uncertainty qualifiers, it is difficult for the HLIF system to draw sound inferences about Joes destination. Other limitations of HLIF with respect to the handling of uncertainty are discussed within the context of the International Society of Information Fusions working group on Evaluation of Technologies for Uncertainty Reasoning (ETURWG)[4] [7, 8].

Representing uncertainty with ontologies is an active area of research, especially in the area of the Semantic Web (e.g., [9, 10]). HLIF requires reasoning with uncertain information about complex situations with many interacting objects, actors, events and processes. Automating HLIF therefore requires expressive representation formalisms that can handle uncertainty. Probabilistic ontologies [11, 12], extend traditional ontologies to capture both domain semantics and associated uncertainty about the domain. The probabilistic ontology language PR-OWL [12] is based on multi-entity Bayesian Networks [13].

2.1 Multi-Entity Bayesian Networks

MEBNs represent the world as a collection of interrelated entities and their respective attributes. Knowl-

[2] http://www-ksl.stanford.edu/knowledge-sharing/kif/
[3] http://www.iso-commonlogic.org/
[4] http://eturwg.c4i.gmu.edu/

edge about attributes of entities and their relationships is represented as a collection of repeatable patterns, known as *MEBN Fragments* (MFrags). A set of MFrags that collectively satisfies constraints ensuring a unique joint probability distribution is a *MEBN Theory* (MTheory).

An MFrag is a parametrized fragment of a directed graphical probability model. It represents probabilistic relationships among uncertain attributes of and relationships among domain entities. MFrags are templates that can be instantiated to form a joint probability distribution involving many random variables. Such a ground network is called a *situation-specific Bayesian network* (SSBN).

MEBN provides a compact way to represent repeated structures in a Bayesian Network. There is no fixed limit on the number of random variable instances, which can be dynamically generated as needed. The ability to form a consistent composition of parametrized model fragments makes MEBN well suited for knowledge fusion applications [14]. MEBN inference can be performed by instantiating relevant MFrags and assembling them into SSBNs to reason about a given situation. As evidence arrives, it is fused into the SSBN to provide updated hypotheses with associated levels of confidence. These are very convenient features for representing diverse information coming from various sensors, which make MEBN attractive as a logical basis for probabilistic ontologies.

2.2 PR-OWL Probabilistic Ontologies

There are basically three aspects that must be addressed for a representational and reasoning framework in support of effective higher-level knowledge fusion:

1. A rigorous mathematical foundation,
2. The ability to represent intricate patterns of uncertainty, and
3. Efficient and scalable support for automated reasoning.

Current ontology formalisms deliver a partial answer to items 1 and 3, but lack a principled, standardized means to represent uncertainty. This has spurred the development of palliative solutions in which probabilities are simply inserted in an ontology as annotations (e.g. marked-up text describing some details related to a specific object or property). These solutions address only part of the information that needs to be represented, and too much information is lost to the lack of a good representational scheme that captures structural constraints and dependencies among probabilities. A true probabilistic ontology must be capable of properly representing those nuances. More formally:

> Definition 1 (from [11]): A probabilistic ontology (PO) is an explicit, formal knowledge representation that expresses knowledge about a domain of application. This includes:
>
> - Types of entities that exist in the domain;
> - Properties of those entities;
> - Relationships among entities;
> - Processes and events that happen with those entities;
> - Statistical regularities that characterize the domain;
> - Inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge related to entities of the domain; and
> - Uncertainty about all the above forms of knowledge;
>
> where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application. ■

POs provide a principled, structured, sharable formalism for describing knowledge about a domain and the associated uncertainty and could serve as a formal basis for representing and propagating fusion results in a distributed system. They expand the possibilities of standard ontologies by introducing the requirement of a proper representation of the statistical regularities and the uncertain evidence about entities in a domain of application. POs can be implemented using PR-OWL[5], a Probabilistic Web Ontology Language that extends OWL with constructs for expressing first-order Bayesian theories. PR-OWL structures map to MEBN structures, so PR-OWL provides a means to express MEBN theories in OWL.

2.3 The UnBBayes MEBN/PR-OWL Plugin

In order to develop and use POs, we have developed a MEBN/PR-OWL plugin to the graphical probabilistic package UnBBayes[6], an open source, Java™-based application developed at the University of Brasilia. The plugin provides both a GUI for building probabilistic ontologies and a reasoner based on the MEBN/PR-OWL framework [15, 16]. Reasoning in the UnBBayes MEBN/PR-OWL plugin involves SSBN construction, which can be seen type of propositionalization, and the subsequent inferential process over the resulting SSBN. Figure 1 shows a screenshot of the UnBBayes MEBN/PR-OWL plugin.

[5] http://www.pr-owl.org
[6] http://unbbayes.sourceforge.net

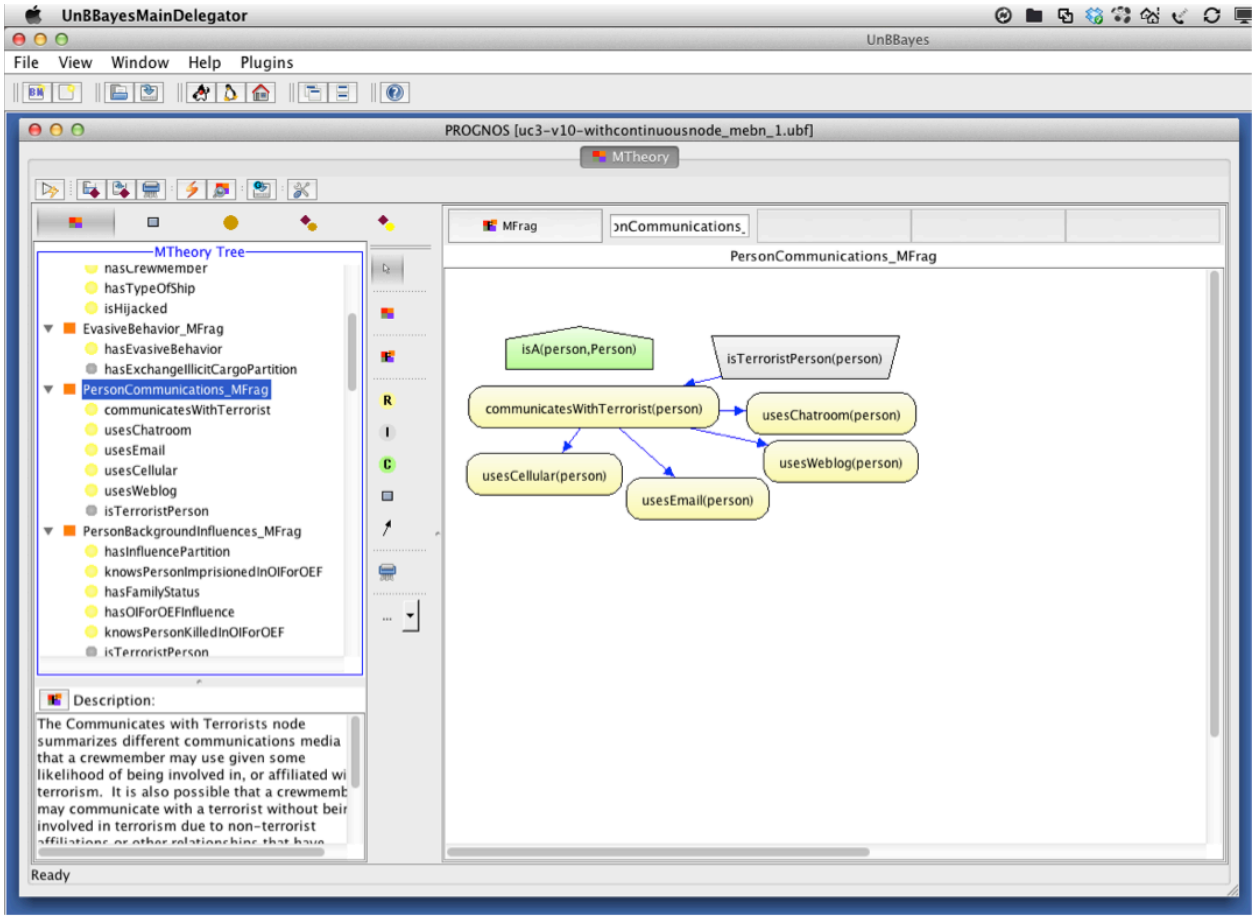


Figure 1: The UnBBayes MEBN/PR-OWL plugin

Many HLIF problems involve spatio-temporal entities, and require reasoning with discrete and continuous, possibly non-Gaussian, variables. To support this requirement, a capability for hybrid MTheories was added to the UnBBayes MEBN/PR-OWL plugin. The plugin can handle MTheories in which continuous variables can have discrete or continuous parents, but no discrete variable is allowed to have a continuous parent. To specify hybrid models, constructs were added to the local distribution scripting language for continuous distributions. The SSBN construction algorithm for building the ground model is basically unchanged except that local distributions can be continuous.

For inference in a hybrid SSBN, the plugin implements the direct message passing (DMP) algorithm [17] to compute, propagate, and integrate messages. DMP combines the unscented transformation [18] and the traditional message-passing algorithm to deal with arbitrary, not necessary linear, functional relationships between continuous variables in the network. DMP gives exact results for polytree conditional linear Gaussian (CLG) networks and approximate results for networks with loops (via loopy propagation), networks with non-linear relationships (via the unscented transformation) and networks with non-Gaussian variables (via mixtures of Gaussians). Mixtures of Gaussian distributions are used to represent continuous messages. The number of mixture components can be as large as the size of the joint state space of all discrete parents. To achieve scalability, the algorithm can restrict the number of mixture components in the messages to satisfy a predefined error bound [19].

Specifically, without loss of generality, suppose a typical hybrid model involving a continuous node X with a discrete parent node D and a continuous parent node U. As shown in Figure 2, messages sent between these nodes are: (1) π message from D to X, denoted as $\pi_X(D)$; (2) π message from U to X, denoted as $\pi_X(U)$; (3) λ message from X to D, denoted as $\lambda_X(D)$; and (4) λ message from X to U, denoted as $\lambda_X(U)$.

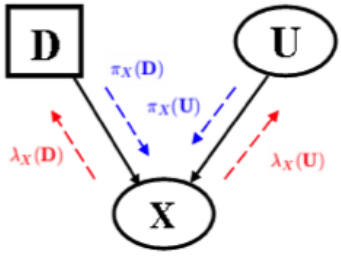


Figure 2: Example hybrid model

In general, for a polytree network, any node X *d–*separates evidence into $\{\mathbf{e}^+, \mathbf{e}^-\}$, where $\mathbf{e}^+$ and $\mathbf{e}^-$ are evidence from the sub-network "above" X and "below" X respectively. The λ and π message maintained in each node are defined as,

$$\lambda(X) = P(\mathbf{e}_X^-|X) \tag{1}$$

and

$$\pi(X) = P(X|\ \mathbf{e}_X^+) \tag{2}$$

With the two messages, it is straightforward to see that the belief of a node X given all evidence is just the normalized product of λ and π values, namely,

$$\begin{aligned} BEL(X) &= P(X|\mathbf{e}) = P(X|\mathbf{e}_X^+, \mathbf{e}_X^-) \\ &= \alpha\lambda(X)\pi(X) \end{aligned} \tag{3}$$

where α is a normalizing constant. It can be shown that for a hybrid network, the π message can be recursively computed as,

$$\begin{aligned} \pi(X) &= \sum_D \int_U P(X|D,U)\pi_X(D)\pi_X(U)dU \\ &= \sum_D \left[\pi_X(D) \int_U P(X|D,U)\pi_X(U)dU\right] \end{aligned} \tag{4}$$

where the integral of $P(X|D = d, U)\pi_X(U)$ over U is equivalent to a functional transformation of $\pi_X(U)$, which is a continuous message in the form of a Gaussian mixture.

Similarly, the λ message for the discrete parents can be obtained as

$$\lambda_X(D = d) = \int_X \lambda(X) \int_U P(X|D = d, U)\pi_X(U)dU dX \tag{5}$$

where $\int_U P(X|D = d, U)\pi_X(U)dU$ is a functional transformation of a distribution over U to X.

On other hand, the λ message for continuous parent U can be computed as

$$\begin{aligned} \lambda_X(U) &= \int_X \lambda(X) \sum_D P(X|D,U)\pi_X(D)\pi_X(D)dX \\ &= \sum_D \left[\pi_X(D) \int_X \lambda(X)P(X|D,U)dX\right] \end{aligned} \tag{6}$$

Equations (3) to (6) form a baseline for computing direct messages between mixed variables.

As mentioned earlier, with the unscented transformation, this method can be modified for arbitrary nonlinear non-Gaussian hybrid models. In addition, the algorithm is scalable by combining the mixture components in the messages with any given error bound

3 Maritime Domain Awareness PO

In 2008, the Department of Defense issued a directive to establish policy and define responsibilities for Maritime Domain Awareness (MDA).[7] The directive defines MDA as the "effective understanding of the global maritime domain and its impact on the security, safety, economy, or environment of the United States." The ability to automatically integrate information and recommendations from multiple intelligence sources in a complex and ever-changing environment to produce a dynamic, comprehensive, and accurate battlespace picture is a critical capability for MDA. This section reports on a prototype probabilistic ontology for maritime domain awareness (MDA-PO). This model, which is depicted in Figure 3, was developed as part of the PROGNOS project [20, 21], with the assistance of two retired Navy officers who served as subject-matter experts.

Figure 4 depicts one of the MFrags of the MDA-PO, the *AggressiveBehavior* MFrag. As the name implies, this is a chunk of knowledge that captures some of the concepts and relationships that are useful to infer whether a ship is displaying aggressive behavior. The three different types of MFrag nodes can be seen: *Context*, *Input*, and *Resident* nodes.

Resident nodes are the random variables that form the core subject of an MFrag. The MFrag defines a local distribution for each resident node as a function of the parents of the resident node in the fragment graph. They can be discrete or continuous. There are three discrete nodes in this MFrag, which are depicted as yellow rounded rectangles in the picture, and five continuous nodes, depicted as rounded rectangles with double lines.

As an example of how the representation works, reports on the propeller turn count of a ship will be an indicator of whether the ship speed is changing or not. Also, there will be different probability distributions for *speedChange(ship)* if the ship is behaving aggressively or not (i.e. if the state of node *hasAggressiveBehavior(ship)* is true or false).

Input nodes, depicted as gray trapezoids in the figure, serve as "pointers" referring to resident nodes in other MFrags. Input nodes influence the local distributions of resident, but their own distributions are defined in the MFrags in which they are resident.

In a complete MTheory, every input node must point to a resident node in some MFrag. For instance, the *hasBombPortPlan(ship)* input node influences the distribution of all the *hasAggressiveBehavior(ship)* nodes

[7] www.dtic.mil/whs/directives/corres/pdf/200502p.pdf

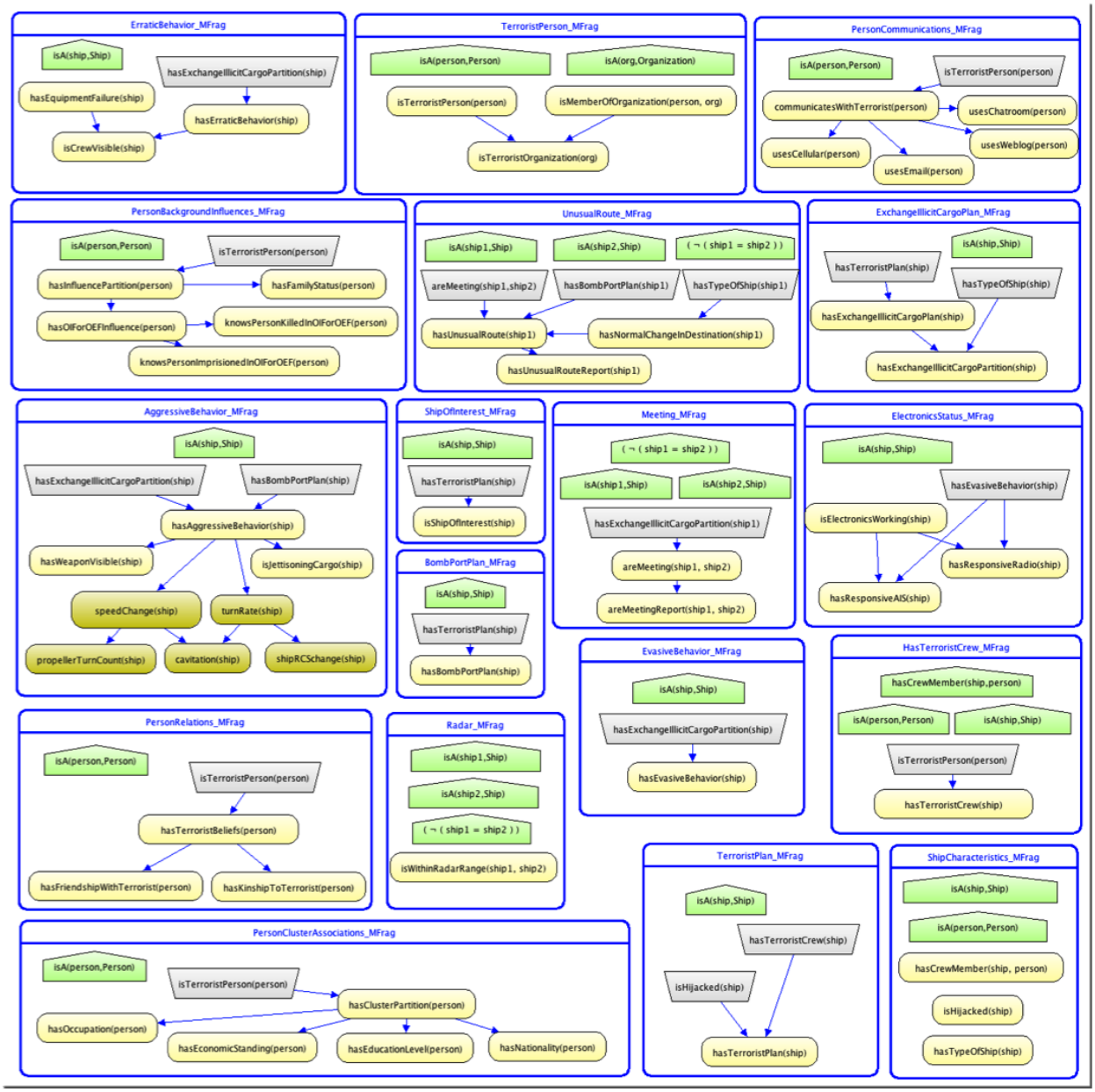


Figure 3: The MDA-PO

that would be instantiated in an SSBN construction process.

Context nodes are Boolean (i.e., true/false) random variables representing conditions that must be satisfied for the probability distribution of an MFrag to apply. Like input nodes, context nodes also have distributions defined in other MFrags.

By allowing uncertainty on context nodes, MEBN can represent several types of sophisticated uncertainty patterns, such as relational uncertainty or existence uncertainty. There is only one context node in the *AggressiveBehavior* MFrag, seen in the figure as a green pentagon.

The MDA-PO is described in detail in [22]. In PROGNOS, the MDA-PO was also used to build the model used to run the test and evaluation process, which we explain in the next Section.

4 Experimental Results

The main objective of the set of experiments presented in this paper was to assess the accuracy, scalability, and overall performance of the SSBN construction and DMP algorithms combined. As a benchmark, we used the UnBBayes implementation of the Junction Tree (JT) algorithm, which is a well-known belief propagation method for Bayesian networks [23] and the fo-

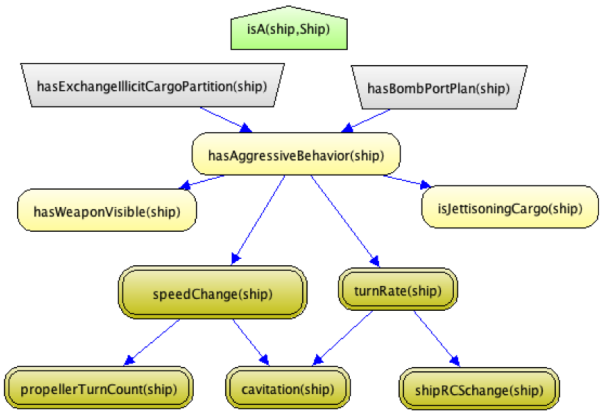


Figure 4: The Aggressive Behavior MFrag

cus of various efforts on algorithm optimization (e.g. [24, 25]).

4.1 Setup and Metrics

Obtaining a real data set for maritime HLIF was not an option for our team. Therefore, we generated various synthetic datasets through an agent-based simulation module, depicted in Figure 5. The module generates simulated scenarios, including entities (e.g., ships, people) and their features. The simulated scenarios serve as the ground truth for evaluating performance. The simulation module also generates reports of the kind the eventual operational system is expected to receive, thus exercising the interfaces and the reasoning module in a realistic manner [21].

The simulation is based on maritime activities (regular and suspicious) with the objective of prevention and disruption of terrorist attacks, sabotage, espionage, or subversive acts. Therefore, the agents on the simulation tool simulate commercial, fishing, recreational, and other types of ships in their normal and suspicious behaviors. Suspicious behaviors are characterized by ships that do not follow their regular or most probable routes according to their origin and destination, by ships that meet in the middle of the ocean for no apparent reason, etc.

For the experiments, the simulation engine generated 9 types of scenarios with different combinations of the number of ships and the associated entities (e.g. organizations, people, etc.). The maximum size of the dataset was limited to 10K entities. After generating each dataset, we added noise as to assess robustness to model misspecification. The scenario for the experiments emulates a U.S. Navy destroyer conducting Maritime Security Operations in a relatively busy area. This is a "needle in a haystack" type of problem, in which the destroyer has information about dozens of ships within a certain radius, but can only verify a few of them. In this case, the HLIF system must integrate the data coming from the ship sensors with information coming from other sources, such as intelligence reports, signals intelligence, HUMINT, and others. We emulate this in the experiments with "area queries" in which all ships within a 60NM radius are queried by the system. More precisely, information on all n known ships within that radius trigger the instantiation of MFrags storing pertinent knowledge, including the one containing the *shipOfInterest(ship)* node. The system would then query the n *shipOfInterest(ship)* nodes that were instantiated.

All experiments were performed in a dedicated computer with an Intel quad-core i7^{TM}processor with 8 GB of RAM, and running MS Windows 7^{TM}64bit.

Accuracy is assessed by the quadratic scoring rule [26]:

$$B(r,i) = \sum_{j=1}^{C}(y_j - r_j)^2$$

where $y_j = 1$ when the j^{th} event is correct and 0 otherwise. C is the number of classes. This is a proper scoring rule, i.e., the score is minimized when the assessed probability is equal to the actual frequency.

To assess scalability, we measured the computation time as a function of the generated SSBN size and the number of ships involved in a query.

4.2 Preliminary Results

The results for accuracy are depicted in Table 1. From the obtained scores, it is clear that the Hybrid system performed better in capturing both the cases in which the *shipOfInterest(ship)* node state was true ($\approx$ 10.36% better) in the ground truth, as well as those in which the node state was false ($\approx$ 14.02% better).

Table 1: Results for Accuracy

Queries on the "Ship of Interest" node	Hybrid System	Discrete System
Ship of Interest = true (ground truth)	0.88203	0.79947
Ship of Interest = false (ground truth)	0.89439	0.78439

These results are consistent with expectations, given the inherent inaccuracies in discretizing the continuous random variables in the MDA-PO. The 10 to 14% improvement from the hybrid with respect to the discrete model was consistent all over the 9 datasets. However, since the datasets were all generated from the same model, it is difficult to assess robustness with this run of experiments. In any case, more complex relationships between nodes are likely to increase the difference

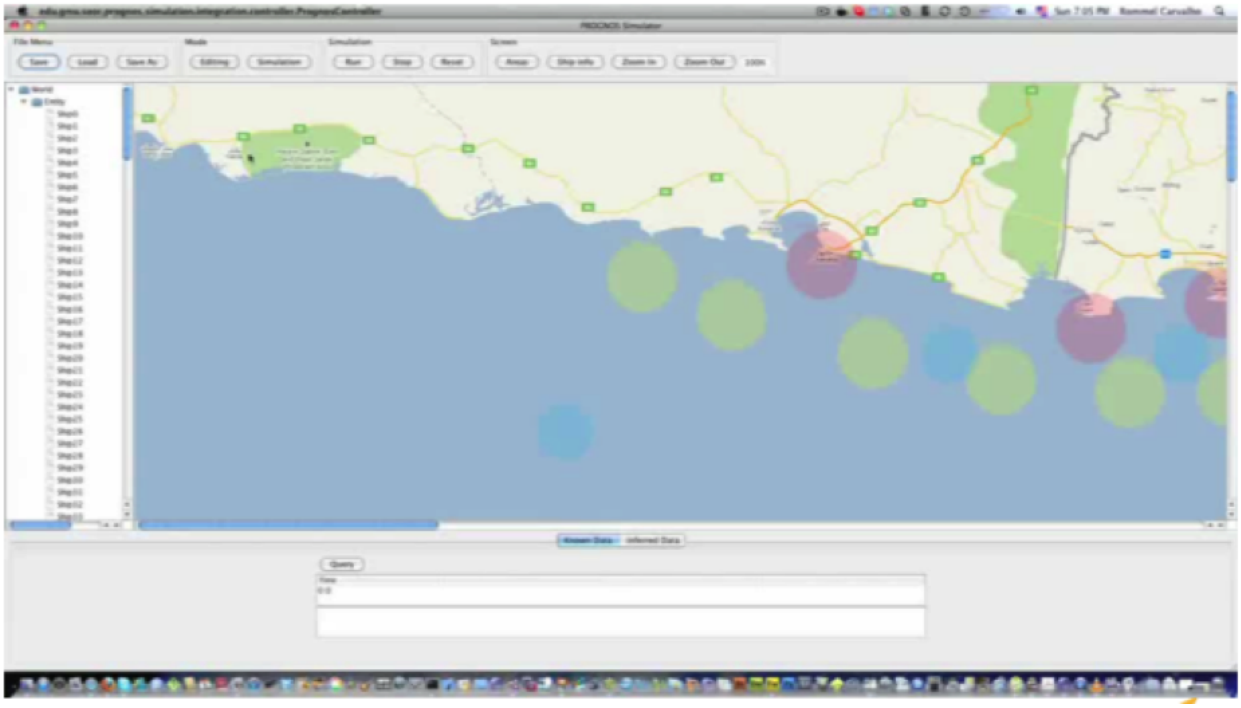


Figure 5: PROGNOS simulation module

in accuracy between the JT and the DMP systems.

Figure 6 below shows the results of the area query experiments. The x-axis conveys the number of nodes generated by each query, which tends to be correlated with the number of ships. However, it was not uncommon to see a few ships generating a large network or vice-versa. The y-axis depicts query time in milliseconds.

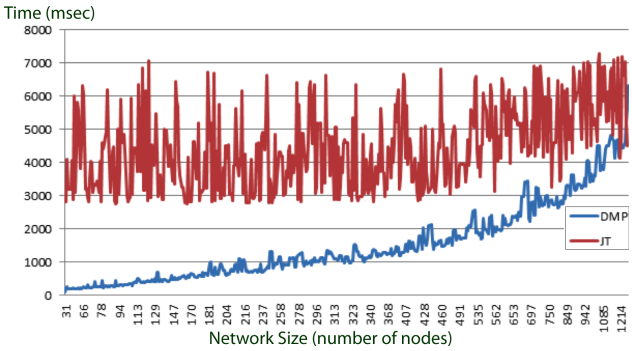


Figure 6: Area Query Time vs. Network Size

Regarding performance, most of the generated networks were between 100 and 500 nodes, and generally yielded a query time below 2 seconds for DMP and 5 seconds for JT. The maximum query time for JT was 7.3 seconds, while the DMP system worst case was 6.4 seconds for a query. The results also show lower variance for DMP query times for a given network size.

Figure 7 shows results for query time vs. number of ships within the 60NM area.

Figure 7: Query Time vs. Number of Ships

This was a different run of experiments, in which the focus was on keeping a controlled number of ships within the query area. This allowed an assessment of how each system reacted to the controlled increase in that number. The performance of JT stayed relatively steady, while the DMP system performed much better for simpler problems but approached the performance of JT when the number of ships was above twenty.

These results were also expected, since our implementation of DMP was not as optimized as the JT implementation in UnBBayes. More specifically, the DMP algorithm was initially implemented in MATLAB^{TM}, and the translation to Java^{TM}was not tuned for performance. Yet, the graph suggests a linear increase in the range considered.

5 Discussion

The experiments were meant to simulate an HLIF system within a relatively simple scenario. In spite of the overall size of the experiments and the fact that it was conducted within a controlled environment, the performance figures are promising. There remain many ways to improve the efficacy of the algorithm. As previously mentioned, the main objective of the testing and evaluation was to assess the gains in accuracy, which clearly lived up to our expectations.

The results also show promise for the feasibility of using probabilistic ontologies as a driver for HLIF systems. Our future steps towards this goal are to continue the optimization of the algorithms, and to seek out new forms of knowledge acquisition techniques. The latter involves automated learning, which has been the subject of our latest research efforts. We also plan to address the research on rare events, and to work with other datasets.

Acknowledgements

The PROGNOS project was partially funded by the Office of Naval Research. The authors acknowledge Richard Haberlin and Michael Lehocky who served as subject-matter experts for developing the MDA-PO, and Rommel Carvalho, for his various contributions to the PROGNOS project.

References

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–200, 1993.

[2] D. McGuinness, "Ontologies for information fusion," in *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, vol. 1, pp. 650 – 657, 2003.

[3] E. G. Little and G. L. Rogova, "Designing ontologies for higher level fusion," *Information Fusion*, vol. 10, pp. 70–82, Jan. 2009.

[4] E. Blasch, E. Dorion, P. Valin, E. Bosse, and J. Roy, "Ontology alignment in geographical hard-soft information fusion systems," in *Information Fusion (FUSION), 2010 13th Conference on*, pp. 1 –8, July 2010.

[5] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL web ontology language semantics and abstract syntax." http://www.w3.org/TR/owl-semantics/, Feb. 2004. W3C Recommendation.

[6] F. Baader, I. Horrocks, and U. Sattler, "Description logics as ontology languages for the semantic web," in *Mechanizing Mathematical Reasoning*, pp. 228–248, 2005.

[7] E. Blasch, J. Llinas, D. Lambert, P. Valin, S. Das, C. Chong, M. Kokar, and E. Shahbazian, "High level information fusion developments, issues, and grand challenges: Fusion 2010 panel discussion," in *Information Fusion (FUSION), 2010 13th Conference on*, pp. 1 –8, July 2010.

[8] P. C. G. Costa, R. N. Carvalho, K. B. Laskey, and C. Y. Park, "Evaluating uncertainty representation and reasoning in HLF systems," in *Proceedings of the Fourteenth International Conference on Information Fusion*, (Chicago, Illinois, USA), July 2011.

[9] K. Laskey and K. Laskey, "Uncertainty reasoning for the world wide web: Report on the URW3-XG incubator group," URW3-XG, W3C, 2008.

[10] L. Predoiu and H. Stuckenschmidt, "Probabilistic extensions of semantic web languages - a survey," in *The Semantic Web for Knowledge and Data Management: Technologies and Practices*, Idea Group Inc, 2008.

[11] P. C. G. Costa, *Bayesian semantics for the Semantic Web*. PhD dissertation, George Mason University, Fairfax, VA, USA, July 2005. Brazilian Air Force.

[12] P. C. G. Costa and K. B. Laskey, "PR-OWL: a framework for probabilistic ontologies," in *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS 2006)* (B. Bennet and F. Christiane, eds.), vol. 150 of *Frontiers in Artificial Intelligence and Applications*, (Baltimore, MD, USA), pp. 237–249, IOS Press, Nov. 2006.

[13] K. B. Laskey, "MEBN: a language for first-order Bayesian knowledge bases," *Artificial Intelligence*, vol. 172, pp. 140–178, Feb. 2008.

[14] P. C. G. Costa, K. Chang, K. B. Laskey, and R. N. Carvalho, "High level fusion and predictive situational awareness with probabilistic ontologies," (George Mason University, Fairfax, VA, USA), May 2010.

[15] R. N. Carvalho, L. L. Santos, M. Ladeira, and P. C. G. Costa, "A GUI tool for plausible reasoning in the semantic web using MEBN," (Los Alamitos, CA, USA), pp. 381–386, IEEE Computer Society, Oct. 2007.

[16] P. Costa, M. Ladeira, R. N. Carvalho, K. Laskey, L. Santos, and S. Matsumoto, "A first-order Bayesian tool for probabilistic ontologies," in *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS 2008)*, (Coconut Grove, FL, USA), pp. 631–636, AAAI Press, May 2008.

[17] W. Sun and K. Chang, "Direct message passing for hybrid Bayesian network and its performance analysis," in *SPIE Defense and Security Symposium*, (Orlando, FL, USA), Apr. 2010.

[18] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the American Control Conference*, vol. 6, p. 45554559, 2002.

[19] H. Chen, K. Chang, and C. J. Smith, "Constraint optimized weight adaptation for gaussian mixture reduction," in *SPIE Defense and Security Symposium*, (Orlando, FL, USA), Apr. 2010.

[20] P. C. G. Costa, K. B. Laskey, and K. Chang, "PROGNOS: applying probabilistic ontologies to distributed predictive situation assessment in naval operations," in *Proceedings of the Fourteenth International Command and Control Research and Technology Conference*, (Washington, DC, USA), CCRP Publications, June 2009. Best paper award of the Collaborative Technologies for Network-Centric Operations Track.

[21] R. N. Carvalho, P. C. G. Costa, K. B. Laskey, and K. Chang, "PROGNOS: predictive situational awareness with probabilistic ontologies," (Edinburgh, UK), July 2010.

[22] R. Carvalho, R. Haberlin, P. Costa, K. Laskey, and K. Chang, "Modeling a probabilistic ontology for maritime domain awareness," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pp. 1 –8, July 2011.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1 ed., Sept. 1988.

[24] L. Zheng, O. J. Mengshoel, and J. Chong, "Belief propagation by message passing in junction trees: Computing each message faster using GPU parallelization," *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 2011.

[25] A. L. M. F. V. Jensen, A. L. Madsen, A. L. Madsen, F. V. Jensen, and F. V. Jensen, "Lazy propagation in junction trees," in *In Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, p. 362369, Morgan Kaufmann Publishers, 1998.

[26] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, pp. 1–3, Jan. 1950.

Goal-Based Person Tracking Using a First-Order Probabilistic Model

Thomas Geier, Stephan Reuter, Klaus Dietmayer and Susanne Biundo
Faculty of Engineering and Computer Science
Ulm University, Germany
forename.surname@uni-ulm.de

Abstract

This work addresses the problem of person tracking using additional background information. We augment a particle filter-based tracking algorithm with a first-order probabilistic model expressed through Markov Logic Networks to tackle the data association problem in domains with a high occlusion rate. Using a high-level model description allows us to easily integrate additional information like a floor plan or goal information into a joint model and resolve occlusion situations that would otherwise result in the loss of association. We discuss the engineered model in detail and give an empirical evaluation using an indoor setting.

1 Introduction

This work concerns the problem of providing background-knowledge to the specialized application of person tracking using a high-level probabilistic model. We demonstrate that a hand-crafted model, built using Markov Logic Networks (MLN) [Richardson and Domingos, 2006], can help in solving the data association problem in tracking during situations where high occlusion prevents the correct association between past and new tracks. By leveraging additional information, like a floor plan or knowledge about goals of single persons, we can resolve otherwise opaque situations.

The usage of a first-order probabilistic model like MLNs allows for an easier modeling task, because dependencies are represented by weighted first-order logical formulas instead of, e.g., conditional probability tables for the case of directed models like Bayesian Networks. In addition, the model is formulated in a lifted form and can be instantiated for the desired number of concurrent tracks or persons within the scene, which is not possible when using completely propositional models or specialized template models like dynamic Bayesian networks [Murphy, 2002](which can scale only along the time axis). A model given in a lifted representation also makes it possible to leverage structure information contained within the lifted formulation for more efficient inference [Gogate and Domingos, 2011]; although this approach is not investigated here.

We motivate our work in the context of an indoor situation, where multiple persons move in a two-room office, containing the laser range finder and several areas of interest like a printer or a coffee maker. We measure the quality of our model by the extent to which it is able to correctly associate object tracks emerging from the tracking algorithm with the correct persons inside the scene.

The rest of the paper is laid out as follows. After we discuss related work, we give an overview of the applied tracking algorithm and introduce the concept of Markov Logic Networks. Then, we describe the investigated problem in detail and discuss the used MLN model. We give an empirical evaluation of the described setup and conclude with some possible extensions to the model and an overall discussion.

2 Related Work

In multi-object tracking, state dependent detection probabilities of objects are disregarded in most applications. Thus, a track disappears shortly after entering an occluded area and a new track is created when the object leaves the occluded area again. Consequently, one object is represented by different track IDs. Especially in scenarios where persons interact several times with a system, changed track IDs lead to the loss of the objects history. The multi-object Bayes filter [Mahler, 2007] allows to integrate state dependent detection probabilities even if the scenario is characterized by a high object density [Reuter and

Dietmayer, 2011]. In case of short term occlusions, the usage of state dependent detection probabilities leads to an improved track continuity. A direct integration of goals into the prediction of a persons' state is crucial, since the persons' action may be contradictory to the assigned goal.

Markov Logic Networks have been used by Sadilek and Kautz [2010] for multi-agent activity recognition based on GPS data in a game of capture the flag. While their work can leverage more expert knowledge (the rules of the game), they do not encounter the data association problem present in the tracking scenario, since each person was carrying a personal GPS receiver. Tran and Davis [2008] apply Markov Logic Networks to a parking lot surveillance scene using video data to recognize which person enters which car. They also track pedestrians across a scene and face the problem of data association. Their sensory information emerges from image data and their focus lies in integrating different information sources that are all extracted from the video stream. Markov Logic Networks are also used by Singla and Domingos [2006] for entity resolution in text mining. This is the problem of inferring which references refer to the same entity and it is similar to the data association problem in tracking. The two latter works use an `equals` predicate for identity maintenance, whereas we approach the problem using an association mapping to underlying entities. When grounding the model our approach only creates associations between currently instantiated track IDs and their corresponding entity, wheres using an `equals` predicate will introduce relations between all objects, which does not seem reasonable in a dynamic domain.

3 Multi-Object Tracking

Standard multi-object tracking algorithms often use object individual single-object trackers like the Kalman filter. The drawback of this multi-object tracking approach is the need of a data association step which assigns the received measurements to the trackers using hard decisions or probabilistic methods [Blackman and Popoli, 1999]. Especially in scenarios characterized by a high object density, the data association is error-prone and degrades the performance of the tracking system, since false associations are irreversible.

A rigorous approach to multi-object tracking is the multi-object Bayes filter proposed by Mahler [2007]. The multi-object Bayes filter uses the random finite set statistics to represent the complete environment by a single filter state. In the innovation step of the multi-object Bayes filter, a multi-object likelihood function calculates the affinity between the predicted state set and the received measurement set. Thus, no data association is necessary.

Further, the multi-object Bayes filter allows to integrate state dependent detection probabilities into the filtering algorithm. In Reuter and Dietmayer [2011], an approach to calculate state dependent detection probabilities based on the occupancy grid mapping approach [Thrun et al., 2005] is proposed. Thus, it is possible to keep track of an object which is occluded for the sensor for a short period of time. Using constant detection probabilities would lead to a track loss, if an object is not visible to the sensor for a few measurement cycles. We use the state dependent tracking algorithm as a comparison for our final results. But for input into the high-level model, we use state independent object tracking. This produces more track IDs for association, and in particular is less prone to false association, which we cannot correct in the upper stage.

An implementation of the multi-object Bayes filter is possible using Sequential Monte Carlo (SMC) methods [Reuter and Dietmayer, 2011, Sidenbladh and Wirkander, 2003, Mahler, 2007]. In difference to well known SMC implementations of the standard Bayes filter a particle set, which represents a random finite set using a finite number of state vectors, is used instead of a standard particle. Further, the number of state vectors in the particle set may change at each time step. In case of a SMC implementation, the integration of the mentioned constraints is possible by reducing the weight of a particle set.

Since the multi-object Bayes filter does not perform a measurement to track association, an extraction of the individual objects out of the multi-object posterior density function is necessary, e.g. using the k-means algorithm [Bishop, 2006].

4 Markov Logic Networks

Markov Logic Networks [Richardson and Domingos, 2006] are a member of the family of first-order probabilistic languages [de Salvo Braz et al., 2008] and their semantics are based on undirected graphical models (Markov networks). In contrast to propositional models like Bayesian networks and Markov Networks, where every random variable has to be specified explicitly, in first-order models the random variables are relations over objects and the model can be scaled by providing the appropriate number of object constants. Moreover, MLNs allow the specification of dependencies as weighted first-order logical formulas. Higher weights make those interpretations more likely, in which more groundings of the formula evaluate to true. We will now briefly cover the formal semantics

of MLNs.

A *Markov Logic Network* $L = \{(f_1, w_1), \ldots, (f_n, w_n)\}$ for $n \in \mathbb{N}$ is a set of first-order formulas $f_1, \ldots, f_n$ with given weights $w_1, \ldots, w_n \in \mathbb{R}$. Together with a finite set of constants C, they define a probability distribution over all interpretations (or possible worlds). An interpretation maps each grounding of each predicate to a truth value. The interpretation of functions must be fixed. Probabilistic functions can be emulated using predicates. Let $g_C(f)$ be the set of groundings of formula f obtained by replacing the free variables in f by all combinations of constants from C. Given an interpretation x, then $n_{C,i}(x) \stackrel{\text{def}}{=} |\{g \mid g \in g_C(f_i) \text{ and } x \models g\}|$ is the number of groundings of formula f_i that are true under x. Then, the probability distribution $P_{L,C}$ that is defined by the MLN L with constants C is given as

$$P_{L,C}(X = x) \stackrel{\text{def}}{=} \frac{1}{Z} \prod_i \exp\left(w_i n_{C,i}(x)\right), \qquad (1)$$

where i ranges over all formulas in L, and Z is a normalizing constant.

Given a set of constants, a MLN can be converted to a Markov network, where nodes correspond to atoms and each ground formula induces a clique over all nodes whose atoms appear inside this formula. For practical reasons, a sorted (or typed) logical language is used to describe MLNs. Using sorted terms, we can limit the size of the grounded network. Also, in their basic form, MLNs do only allow restricted usage of logical functions. Usually functions are simulated by specially marked predicates, which enforce a functional dependency of one or more arguments on the remaining arguments. We notate functional arguments of predicates by underlining them. Such a predicate can be translated to a multi-valued random variable.

5 Problem Description

We consider an indoor scene which resembles an office setting. The corresponding floor plan is depicted in Figure 1a. A laser range finder is placed in one corner of the main room and provides distance information in a plane about one meter above ground. The beam almost completely covers the main room, but there exists a second room that has virtually no sensor coverage. There is only one entrance to the room complex and the separate room has only a single exit, which is the door to the main room. The setting contains several features that may serve as goal destinations for persons navigating inside the scene; like a printer or a coffee maker. Knowledge about destinations of persons is taken as given; although it is easy to motivate the existence of such information depending on the application. Issued print jobs could be recognized by a special program installed on the PC, or visits to the coffee maker could be predicted from personal habits.

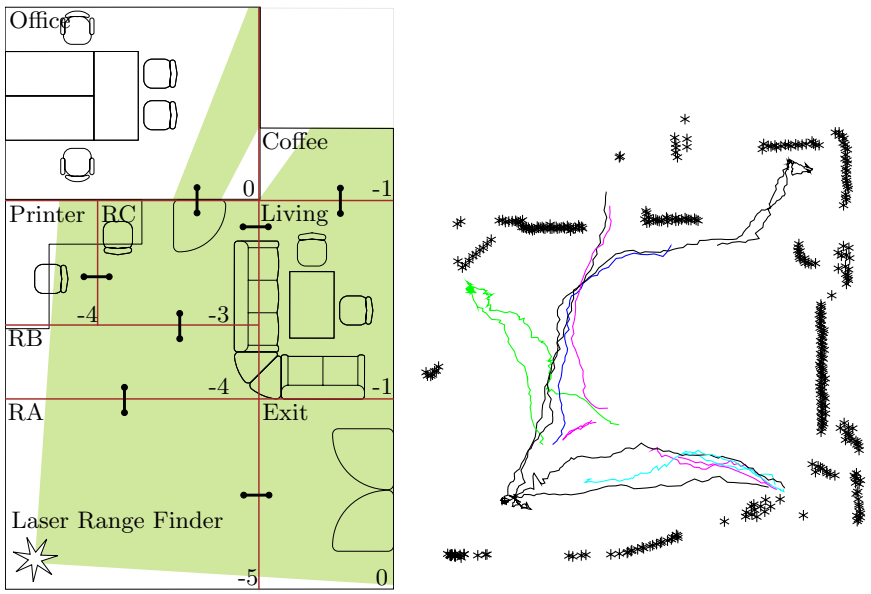
There are one to three persons inside the scene simultaneously. Major occlusion caused by static objects, like walls, occurs when people enter the second room. Minor static occlusion can occur near the coffee maker. During the scenes with more than one person, dynamic occlusion occurs when persons are covered by other persons standing between them and the sensor.

Using only the particle filter-based tracking algorithm to process the output of the laser range finder, problems arise when people produce no measures for an extended period of time because they are inside the separate room or because they are hidden by another person. For shorter occlusion durations it is possible to keep the track of a single occluded person alive for long enough for the person to reappear and re-association is completely handled by the tracking algorithm. If two persons enter the same occlusion area, their estimated positions begin to mix spatially and once they emerge again, re-association becomes more and more arbitrary with increasing occlusion duration. In these scenarios, a direct integration of the Social Force model into the prediction of the persons state of the tracking algorithm may increase the performance of the system [Reuter and Dietmayer, 2010]. The Social Force model aims at describing pedestrian movement by virtual forces exerted by other persons and environmental features [Helbing and Molnar, 1995]. Since the model heavily depends on the destinations of the person, a tight integration with a high-level knowledge base, as described in this work, seems promising for such an approach.

Figure 1b shows an example of the tracking results for one of the sequences with three persons. The trajectories are illustrated by solid lines. Since the results are generated without the usage of the state dependent detection probability, the trajectories are interrupted quite often in the area corresponding to region RA, where a dynamic occlusion occurs.

6 Description of the Model

In this section, we describe the used MLN model and discuss some of the difficulties and design choices we have encountered in its engineering. The complete model is factored into four modules. We begin with a discussion of two concepts that cannot be associated with distinct model parts but influence nearly every aspect – the representation of space and time.



(a) Floor plan (b) Example trajectories

Figure 1: (a) Floor plan of the location used for the experiments. In addition the picture shows the eight discrete regions that are used for the MLN model, how they are connected via the handles and their static occlusion value in their lower right. (b) Example trajectories of a three person sequence: trajectories are illustrated by solid lines in different colors. The black stars are measurements of static objects like walls.

From continuous to discrete space. The basic MLN can only represent discrete random variables. There exists an extension of MLNs to continuous variables [Wang and Domingos, 2008], but no working implementation is available. For this work, we reduce the continuous spatial estimates obtained from the tracking algorithm to a few discrete regions. For ease of modeling and processing of data, we choose a rectangular shape. We do not create a uniform grid, but try to respect functional aspects of the environment concerning the problem. For example, it does not make sense to further split the office into smaller areas if there is no distinction for the sensor (everything is one connected occluded area) and there is only a single goal inside. Even having two separate goals like two work stations might not justify the introduction of separate regions for each, as long as the rest of the model can not discriminate between them. We have defined a total of eight regions, which are depicted in Figure 1a. Sticking with a low number of regions also made an exact evaluation of the final model feasible. Depending on the inference approach, there might be no significant overhead when using a larger state space for the spatial component. Although, the model engineering may become more intricate when opting for more fine grained regions. Taking the characteristics of the sensor into account, a radial layout for regions seems like a promising approach, but this was not investigated.

Representation of time. In order to model dynamic domains, we assign a dedicated time sort, whose constants are elements from the natural numbers. One usually aims to construct a model that fulfills the Markov property, i.e., the state at time $t+1$ only depends on the state at time t. This means that formulas may only contain predicates of at most two different times, which then must be successive. But in the presented model, the predicates that represent the association of tracks to persons are not time-indexed, which makes them static. This makes it difficult to apply standard dynamic inference algorithms, which usually assume the Markov property. But a static variable can be considered as a dynamic variable, for which the same value is enforced in every time step. Fortunately, these static association are only referenced over a limited period of time — the life-time of a track — so they do not pile up over the course of the complete sequence. For time resolution we have settled for the duration of about one second, which seems like a good compromise between inference complexity and accuracy for the given problem.

The tracking model. The basic functionality for interfacing with the tracking algorithm is provided by a MLN module that contains objects of the sorts `Track` and `Person`. To notate variables of some sort, we use the initial letter of the sort name in lower case. For the sort `Track`, the letter 'm' is used be-

cause of the ambiguity with sort Time. For both sorts Track and Time, there exist time-dependent predicates $\texttt{at}_T$: Time × Track × Region and $\texttt{at}_P$: Time × Person × Region that give the current location of a track or person, respectively. The time-independent predicate a : Track → Person associates Track objects to Person objects. The correspondence of tracks to persons inside the MLN is similar to the correspondence of measurements to tracks inside the tracking algorithm. The output of the tracking algorithm is converted to observations of the $\texttt{at}_T$ function and an additional completely observed predicate act : Time × Track, whose purpose is mainly to be able to prune groundings of formulas that depend on inactive track IDs. The usage of this predicate is omitted for clarity. Information about goals of persons can attach to the location of the person objects. The core tracking model then consists of the following two formulas.

$$5 \quad \texttt{at}_T(t, m, \underline{r}) \wedge \texttt{a}(m, \underline{p}) \Rightarrow \texttt{at}_P(t, p, \underline{l}) \qquad (2)$$

$$-3 \quad \texttt{a}(m_1, \underline{p}) \wedge \texttt{a}(m_2, \underline{p}) \wedge m_1 \neq m_2 \qquad (3)$$

Formula 2 probabilistically forces a person to be in the same region as its associated track. By design a track can only be associated to one person at a time because the last parameter of the predicate a is declared functional. Formula 3 probabilistically enforces the association to also be a one-to-one relation. The engineering of the formula weights is explained at the end of this section. This formula is limited to concurrently instantiated tracks using the act predicate (not listed).

The floor plan. We use the static predicate adj : Region × Region to encode the connectedness of the regions. All instances are fully observed and adhere to the floor plan given in Figure 1a. A single formula forces persons to move between regions only according to the given layout:

$$\infty \quad \texttt{at}_P(t, p, \underline{l_1}) \wedge \texttt{at}_P(t+1, p, \underline{l_2}) \Rightarrow \texttt{adj}(l_1, l_2) \quad (4)$$

This formula is deterministic to prevent persons from "teleporting" through the scene. If regions allow for a traversal in less than a second (one time step), this rule becomes invalid. But since the association of tracks to persons allows for some slack, a person in the model can "catch up" to the location of its real counterpart after some time steps, only violating Formula 2.

The occlusion model. To prevent persons without an associated track from wandering across the scene (since no track influences their current location), we need to express that persons usually have a track unless they are indeed occluded. In our setting both static and dynamic occlusions occur, being caused by walls or other persons, respectively. For our experiments, only static occlusion information is modeled. This is done by assigning a certain probability to each region that it may contain untracked persons. The probability is larger for areas of high static occlusion, like the separate room. We also assign a higher occlusion to regions that are more likely to be dynamically covered, like the region around the coffee maker. By assigning low occlusion probabilities to central regions that have a good sensor coverage we penalize persons silently slipping past the sensor. Formula 5 is provided once for each region r. The weight w_r is the occlusion value, which is given in Figure 1a.

$$w_r \quad \texttt{at}_P(t, p, \underline{r}) \wedge \neg\exists m : (\texttt{act}(t, m) \wedge \texttt{a}(m, \underline{p})) \quad (5)$$

Goals and their dynamics. The last model part handles the goals of persons. We associate goals with regions and add another time-dependent function goal : Time × Person × Region, where goals are also allowed to assume the special location NULL to signal that a person currently has no goal. The following formulas describe the dynamics of goals:

$$\infty \quad \texttt{goal}(t, p, \underline{l}) \wedge \neg\texttt{at}_P(t, p, \underline{l}) \Rightarrow \texttt{goal}(t+1, p, \underline{l}) \quad (6)$$

$$\infty \quad \texttt{goal}(t, p, \underline{l}) \wedge \texttt{at}_P(t, p, \underline{l}) \Rightarrow \texttt{goal}(t+1, p, \underline{l}) \vee \texttt{goal}(t+1, p, \underline{\text{NULL}}) \quad (7)$$

$$0.1 \quad \texttt{goal}(t, p, \underline{\text{NULL}}) \quad (8)$$

Formulas 6 and 7 achieve that goals can only be cleared when a person reaches their associated region. Formula 8 encodes the urge of people to clear their goal. Stating this rule in this particular form results in persons trying to reach their goal as soon as possible, since otherwise the penalty accumulates over time. Another way to make people reach their goals is to make it more likely for a person to be inside the region of their goal. Both rules work equally well for our dataset but might make a difference when applied to longer sequences or under a different setting.

Elicitation of weights. There exist two major ways to determine the weights of the probabilistic formulas: Learning from data and elicitation from experts; where for common sense domains, like the one we are dealing with, everyone is usually an expert. Both the learning of weights and the direct specification approaches have been followed in the literature. For the case of our related work, Sadilek and Kautz [2010] and Singla and Domingos [2006] are employing learning and Tran and Davis [2008] specify the weights by hand. Due to the limited size and the common sense nature of our dataset we decided to specify the weights ourself.

The approximation to consider the weight as the logarithmic odds of the formula being true [Richardson and

Domingos, 2006] can serve as a good starting point, but it only holds as long as formulas do not share predicates. After assigning some reasonable initial values, we iteratively looked at predictions of the model for selected sequences and adjusted the weights if the predictions did not conform with our expectations. We began by adjusting the weights for sequences with only one persons and switched to larger test sequences once the model made sensible predictions for the training data at hand. Since MLNs are based on undirected graphical models (which means they are locally unnormalized), there are no absolute correct values, but the weights of different formulas have to be balanced against each other.

7 Preprocessing of Tracking Information

We go on and describe how tracking data is processed for input to the MLN model. After extraction of the individual objects in the multi-object Bayes filter, we obtain a set of single object particles X_m^t for each track ID m and time step t. We then apply two data reduction steps. First, the MLN model works on a coarser time scale of 1.25 steps per second, while the tracking algorithm runs with 12.5 steps per second. We drop the intermediate steps without further processing. A different approach might aggregate them, e.g., by averaging, but this would also distort the meaning of the data, because it cannot be considered a snapshot of the situation anymore.

Depending on the quality of the tracking algorithm and the used object model, there can be many false positive tracks, e.g., when people spread their arms away from their body, crossing the plane of laser beams. To reduce these false tracks, we use the existence probability to eliminate insignificant tracks. It is given by $|X_m^t|/N$; the number of particles for track ID m divided by the total number of particles N. We drop all tracks from a time step whose existence probability is below 0.5. For our test sequences, the output of the tracking algorithm usually contains about thirty tracks per sequence, but only less than ten remain after applying both reduction processes.

For each time step t and each track ID m that survive the described process we add the track as active to our MLN model via observation of the `act` predicate. We then bin the single object particles into the discrete regions. Most of the time all particles are contained in a single region and we create an observation of the $\texttt{at}_T$ function. In cases where the particles of a track m spread over several regions we reflect this as probabilistic evidence by adding a formula $(w_l, \texttt{at}_T(t, m, \underline{l}))$ for each location l and calculating the weight as the logarithmic odds $w_l = \log \frac{p_l}{1-p_l}$, where p_l is the relative frequency of a particle of track m being in region l.

8 The Inference Problem

Markov Logic Networks can be seen as template models for undirected graphical models [Koller and Friedman, 2009]. Their semantics are defined using the ground version of these networks. As such the described MLN represents an undirected version of a dynamic Bayesian network [Murphy, 2002]. The effort for exact inference in such models is usually exponential in the number of variables within one time slice, because most variables within one time step become dependent on each other after some steps in most models.

The model described in this work also suffers from this problem. The cause that all variables of a time slice become dependent lies in the probabilistic data association; which is a hard problem at its core. In our case the problem of exact inference is exponential in $m_T + n_P$, where m_T is the maximum number of simultaneous tracks and n_P is the total number of persons in the model. Here m_T stems from the association predicates and n_P are the instantiations of the $\texttt{at}_P$ predicate for one time step.

For our evaluation we perform exact inference on the model by exploiting context-specific independence [Koller and Friedman, 2009, pp. 171]. Given an assignment to all association variables, the model factorizes into components for each person and thus becomes tractable. Our largest sequence contained only 10 tracks, which results in 3^7 possible associations to three persons after observing the correct association for three initial tracks. After conditioning on the association variables we calculate the partition function for each association using variable elimination along a min-degree variable ordering. This approach is not suited for online filtering. For this purpose a rao-blackwellized particle filter, which collapses all but the association variables, seems like a good solution [Koller and Friedman, 2009, pp. 526]. Evaluating the performance of this inference approach on the presented model is open for future work.

9 Evaluation

We recorded 9 sequences in total; three sequences with one, two and three persons, respectively. The duration of each sequence is about one minute. The course of events is the same among sequences with the same number of persons; the sequences vary during the part where multiple persons wander around the main room.

The setups with one person only feature static occlu-

Sequence	Persons	Unassigned Tracks $\downarrow p_D = c$	Unassigned Tracks $\downarrow p_D(x)$	Model M $\downarrow a_-$	Model M $\uparrow p_{\text{true}}$	Model MO $\downarrow a_-$	Model MO $\uparrow p_{\text{true}}$	Model MOG $\downarrow a_-$	Model MOG $\uparrow p_{\text{true}}$
S1-1532	1	1	1	0	0.34	0	0.99	0	1.00
S1-1640	1	1	1	0	0.34	0	0.99	0	1.00
S1-1737	1	1	1	0	0.34	0	1.00	0	1.00
S2-2056	2	3	2	2	0.10	2	0.27	0	0.54
S2-2207	2	4	4	3	0.00	4	0.01	3	0.03
S2-2329	2	3	3	2	0.09	2	0.26	0	0.51
S3-4628	3	5	1	3	0.25	0	0.45	0	0.89
S3-4734	3	5	3	2	0.15	0	0.52	0	0.55
S3-5306	3	7	4	1	0.13	1	0.12	1	0.25

Table 1: For each of the nine sequences used for evaluation, we give the number of invented tracks minus the number of persons for tracking with constant detection probability $p_D = c$ and with state dependent detection probability $p_D(x)$. For the three MLN models, we give the number of false associations a_- and the probability of the true assignment p_{true}.

sion caused by walls, caused by the single person staying inside the office for several seconds. This results in its track being reinvented upon entering the main room again. With two persons there is dynamic occlusion, where one person covers the other person. Both persons enter the office together and thus cannot be distinguished once they reappear. Goal information for one person can resolve this issue and we can obtain a good association again. In the scenes with three persons, one person enters the office while the two remaining persons stay inside the main room. Dynamic occlusion occurs while all three persons are walking around in front of the sensor. Tracks are lost and recreated often, which can also be observed in Figure 1b inside the area corresponding to region RA. When two persons are simultaneously shadowed by the third one, it is not possible to associate the reappearing tracks to the correct persons just by means of the sensor. In this case goal information can be used to identify the correct association.

We have evaluated three models that incorporate an increasing amount of domain knowledge. The first model `M` uses only the basic tracking model and the floor plan. The second model `MO` comprises all of the first model and the static occlusion model. The third model `MOG` adds information about personal goals. One person visits the coffee maker in each sequence. We assign the `Coffee` region as goal for this person in every sequence. All other persons have no goal assigned. The goal information is able to resolve confusions that happen before the designated person visits its goal region. This does not happen in every sequence.

The MLN is instantiated for three persons in every setup, regardless of the number of persons appearing in the scene. For each model and each sequence, we observe the correct association for the first track of each person and evaluate how well we can associate new tracks. In our dataset, the number of tracks that remain unassigned after labeling the starter tracks varies between one and seven.

The results of our evaluation are given in Table 1. For each sequence, we give the number of false track assignments of the most probably association. In addition, we provide the probability of the correct joint association. This is interesting for cases where no false associations were made even with a simpler model, and can show an improved significance of the correct association when using a more sophisticated model.

To provide a baseline, the number of track confusions and losses of a state-of-the-art multi-object Bayes filter with state dependent detection probability ($p_D(x)$) and the ones using the same filter with constant detection probability ($p_D = c$) are given [Reuter and Dietmayer, 2011]. For both filters, the number of persons inside the scene is subtracted from the total number of significant tracks. If the tracking algorithm works perfectly, this number will be zero. The output of the $p_D = c$ filter is used as input for the MLN stage; so this number equals the number of associations made by the high-level model and thus equals the possible maximum number of false associations.

We observe that the algorithm with the state dependent detection probability reduces the number of unassigned tracks dramatically in the scenarios with three persons, where a lot of short term occlusions occur. In the scenarios with one or two persons, where the long-term occlusions due to static objects dominate, the usage of the state dependent detection probability has nearly no influence on the number of unassigned tracks. The high-level model `MO` using only the floor plan and the static occlusion model delivers results

that are at least on par with the state dependent tracking algorithm. Using goal information for one person can further improve the results. By our judgment, this is not possible by relying solely on the data obtained from the laser range finder in general.

One of the two person sequences (S2-2207) shows very bad performance of the MLN model for all three cases. Our investigation has shown that an outstretched arm has caused a significant track that made it through the data reduction process. The relatively high weight on Formula 3 prevented the association of this track to the owner of the arm. Thus, a third person was forced by the model to appear at this spot and remained present over the course of the scene.

10 Conclusion

We have described an approach to solving the data association problem for person tracking with a high-level probabilistic model described using Markov Logic Networks. We showed how to map the output of a regular tracking algorithm into a discrete spatial representation, which makes it easy to attach additional information, e.g., personal goals, and allows the use of inference techniques for discrete probabilistic models.

Especially in scenarios with long-term occlusions, where even the multi-object Bayes filter is not able to continue to track hidden objects, the association using MLN outperforms the tracking-based approach when using exact evaluation. On the other hand, a sophisticated tracking algorithm is adequate in scenarios with occlusions of no more than one second and might be able to scale more easily to larger domains.

In future, we plan to integrate more information out of the knowledge-base directly into the tracking algorithms like, e.g., probabilistically modeled destinations or goals of a person.

Acknowledgements

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

R. de Salvo Braz, E. Amir, and D. Roth. A survey of first-order probabilistic models. In D. Holmes and L. Jain, editors, *Innovations in Bayesian Networks*, Studies in Computational Intelligence. Springer, 2008.

V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 256–265. AUAI Press, 2011.

D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *PHYSICAL REVIEW E*, 51: 4282–4286, 1995.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

R. P. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House Inc., Norwood, 2007.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, 2002.

S. Reuter and K. Dietmayer. Adapting the state uncertainties of tracks to environmental constraints. In *Proceedings of the 13th International Conference on Information Fusion*, pages 1–7, 2010.

S. Reuter and K. Dietmayer. Pedestrian tracking using random finite sets. In *Proceedings of the 14th International Conference on Information Fusion*, pages 1–8, 2011.

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

A. Sadilek and H. Kautz. Recognizing multi-agent activities from GPS data. In *Proceedings of 24th AAAI Conference on Artificial Intelligence*, pages 1134–1139, 2010.

H. Sidenbladh and S.-L. Wirkander. Tracking random sets of vehicles in terrain. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, page 98, 2003.

P. Singla and P. Domingos. Entity resolution with markov logic. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 572–582, 2006.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

S. Tran and L. Davis. Event modeling and recognition using markov logic networks. *Computer Vision–ECCV 2008*, pages 610–623, 2008.

J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of the 22th AAAI Conference on Artificial Intelligence*, pages 1106–1111, 2008.

Contrasting Temporal Bayesian Network Models for Analyzing HIV Mutations

Pablo Hernandez-Leal, Lindsey Fiedler-Cameras, Alma Rios-Flores,
Jesús. A González and L. Enrique Sucar
{pablohl,lfiedlerc,jagonzalez,esucar}@inaoep.mx
Instituto Nacional de Astrofísica, Óptica y Electrónica
Coordinación de Ciencias Computacionales
Sta. María Tonantzintla, Puebla, México

Abstract

Evolution is an important aspect of viral diseases such as influenza, hepatitis and the human immunodeficiency virus (HIV). This evolution impacts the development of successful vaccines and antiviral drugs, as mutations increase drug resistance. Although mutations providing drug resistance are mostly known, the dynamics of the occurrence of those mutations remains poorly understood. A common graphical model to handle temporal information are Dynamic Bayesian Networks. However, other options to address this problem exist. This is the case of Temporal Nodes Bayesian Networks. In this paper we used both approaches for modeling the relationships between antiretroviral drugs and HIV mutations, in order to analyze temporal occurrence of specific mutations in HIV that may lead to drug resistance. We compare the strengths and limitations of each of these two temporal approaches for this particular problem and show that the obtained models were able to capture some mutational pathways already known (obtained by clinical experimentation).

1 INTRODUCTION

Viral evolution is an important aspect of the epidemiology of viral diseases such as influenza, hepatitis and human immunodeficiency virus (HIV). HIV is the causal agent for the disease known as Acquired Immunodeficiency Syndrome (AIDS), a condition in which progressive failure of the immune system allows opportunistic life-threatening infections to occur.

This viral evolution impacts the development of successful vaccines and antiviral drugs, as mutations caused by viral evolution increase drug resistance. Although the mutations which result in drug resistance are mostly known, the dynamics of the appearance of those mutations and the time of occurrence remains poorly understood.

Bayesian Networks (BNs) have proven to be successful in various domains, including medicine and bioinformatics. However, classical BNs are not well equipped to deal with temporal information. The common approach to handle temporal information is to construct a Dynamic Bayesian Network (DBN) (Dagum, Galper, and Horvitz, 1992), however other options exist such as Temporal Nodes Bayesian Networks (TNBN) (Arroyo-Figueroa and Sucar, 1999).

In this paper we use both approaches, Dynamic Bayesian Networks and Temporal Bayesian Networks, to model the mutational pathways for four specific HIV antiretrovirals. The objective is to compare the pathways that we obtain with our models against the pathways obtained from experimental testing. In this way, we can see if the models reflect the temporal clinical information reported in many reference sources.

2 BAYESIAN NETWORKS

BNs are directed acyclic graphs used to model conditional dependencies between random variables. The data represented by a BN is typically static, however in many contexts a need arises to model processes whose state variables change throughout the course of time. Dynamic Bayesian Networks evolved to tackle this shortcoming.

2.1 DYNAMIC BAYESIAN NETWORKS

A Dynamic Bayesian Network extends the concept of a Bayesian network to incorporate temporal data. Just as with classic BNs, a static causal model is created to represent a process at a single point in time; multiple copies of this model are then generated for each time

point or *slice* belonging to a temporal range of interest and links between copies are inserted to capture temporal relations.

When modeling dynamic information, DBNs obey the assumption that future states are conditionally independent from past states given the present state (Markov property); additionally they assume that the conditional probabilities which describe the temporal relations between random variables of adjacent time slices do not change (stationary process). By allowing these two basic assumptions, DBNs can offer a more compact model of the dynamic process by defining a 2-time-slice Bayesian network (2-TBN). This 2-TBN can be further unrolled to do inference on the entire temporal range of interest.

The learning of a DBN can be seen as a two stage process (Friedman, Murphy, and Russell, 1998). The first stage refers to the learning of the static model and is done in an identical manner as with classic BNs. The second stage learns the transition network, that is, the temporal relations between random variables of different time slices.

An alternative to DBNs are Temporal Nodes Bayesian Networks (Arroyo-Figueroa and Sucar, 1999) which are another extension of Bayesian Networks.

2.2 TEMPORAL NODES BAYESIAN NETWORKS

TNBNs (Arroyo-Figueroa and Sucar, 1999) were proposed to manage uncertainty and temporal reasoning. In a TNBN, each Temporal Node has intervals associated to it. Each node represents an event or a state change of a variable. An arc between two Temporal Nodes corresponds to a causal-temporal relation. One interesting property of this class of models, in contrast to Dynamic Bayesian Networks, is that the temporal intervals can differ in number and size. So, only one (or a few) instance(s) of each variable is required, assuming there is one (or a few) change(s) of a variable state in the temporal range of interest. No copies of the model are needed, thus compacting the representation without losing expressiveness.

A TNBN is composed by a set of TNs connected by arcs. A TN, v_i, is a random variable characterized by a set of states $\mathbf{S}$. Each state is defined by an ordered pair $S = (\lambda, \tau)$, where λ is the particular value taken by v_i during its associated interval $\tau = [a, b]$, corresponding to the time interval in which the state changes, i.e. change in value occurs. In addition, each TN contains an extra default state $s =$ ('no change', $\emptyset$) with no associated interval. Time is discretized in a finite number of intervals, allowing a different number and duration of intervals for each node . Each interval

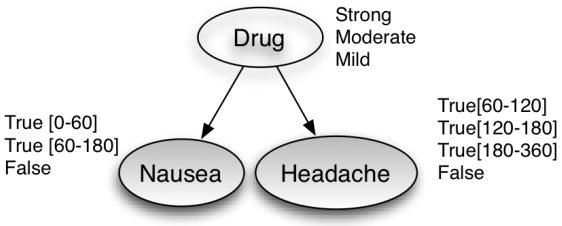


Figure 1: An example of a TNBN. The Drug node is an Instantaneous Node, so it does not have temporal intervals. The Nausea and Headache are temporal nodes with intervals associated to them.

defined for a child node represents the possible delays between the occurrence of one of its parent events and the corresponding child event. If a node lacks defined intervals for all its states then it is referred to as an *instantaneous node*. There is at most one state change for each variable (TN) in the temporal range of interest.

Formally, let $\mathbf{V}$ be a set of temporal and instantaneous nodes and $\mathbf{E}$ a set of arcs between nodes, a TNBN is defined as:

Definition 1. *A TNBN is a pair $B = (G, \Theta)$ where G is a directed acyclic graph, $G = (\mathbf{V}, \mathbf{E})$ and, Θ is a set of parameters quantifying the network. Θ contains the values $\Theta_{v_i} = P(v_i|Pa(v_i))$ for each $v_i \in \mathbf{V}$; where $Pa(v_i)$ represents the set of parents of v_i in G.*

The following is an example of a TNBN of a patient administered with a drug causing two side effects. Its corresponding graphical representation is shown in Figure 1.

Example 1. *Assume that at time $t = 0$, a Drug is administered to a patient. This kind of drug can be classified as* strong, moderate *and* mild. *To simplify the model we will consider only two consequences for the patient, Nausea and Headache. These events are not immediate, we will assume that they depend on the type of drug, therefore, they have temporal intervals associated. For the Nausea node two intervals are defined $[0-60]$, $[60-180]$, for the Headache node three intervals are defined $[60-120]$, $[120-180]$, $[180-360]$. These intervals represent that the state of the node changed during that period of time.*

The learning algorithm for TNBN used in this work has been presented in (Hernandez-Leal, Sucar, and Gonzalez, 2011). We now present a brief description.

1. The algorithm begins by performing an initial discretization of the temporal variables, for example

using an Equal-Width discretization. With this process it obtains an initial approximation of the intervals for all the Temporal Nodes.

2. It then performs a standard BN structural learning using the K2 learning algorithm (Cooper and Herskovits, 1992) to obtain an initial structure. This structure will be used in the third step, the interval learning algorithm.

3. The interval learning algorithm refines the intervals for each TN by means of clustering. For this, it uses the information of the configurations of the parent nodes. To obtain the initial set of intervals a Gaussian mixture model is used as a clustering algorithm for the temporal data. Each cluster corresponds, in principle, to a temporal interval. The intervals are defined in terms of the mean and the standard deviation of the clusters. The algorithm obtains different sets of intervals that are merged and combined, this process generates different interval sets that will be evaluated in terms of the predictive accuracy (Relative Brier Score). The algorithm applies two pruning techniques in order to remove some sets of intervals that may not be useful and also to keep a low complexity of the TNBN. The best set of intervals (that may not be those obtained in the first step) for each TN is selected based on predictive accuracy. When a TN has as parents other Temporal Nodes, the configurations of the parent nodes are not initially known. In order to solve this problem, the intervals are sequentially selected in a top-down fashion according to the TNBN structure.

The algorithm then iterates between the structure learning and the interval learning. However, for the experiments presented in this work, we show the results of only one iteration.

3 HIV AND ANTIRETROVIRAL THERAPY

Viral evolution impacts the development of successful vaccines and antiviral drugs, as mutations (caused by viral evolution) increase drug resistance. In HIV, this is particularly relevant as the virus ranks among the fastest evolving organisms (Freeman, Herron, and Payton, 1998). In viral diseases, such as HIV, it would be important to develop proactive therapies that predict the advent of mutations, thus reducing the possibility of drug resistance, which will then help to predict the duration of a new treatment. Viral therapy failure in patients treated for the HIV-1 infection is commonly associated with the emergence of mutations which are resistant to specific drugs. In addition, a troublesome cross-resistance within the same class of medications complicates the therapeutic options for patients who have treatment regimen failure. Cross-resistance is particularly common among the protease inhibitors (PIs), making the sequential use of these agents frequently problematic. Although the mutations which result in drug resistance are mostly known, the dynamics of the appearance of those mutations on the time of occurrence remains poorly understood.

The relationship between phenotypic susceptibility to some inhibitors and the genotypic pattern was investigated in the same inhibitors. From these studies we now know the resistant patterns associated with the inhibitors most frequently used. This information has led to the ability to select a new salvage therapy. In addition, if we know the pathway and time of occurrence of resistant mutations of common and well known therapies, then this might lead to predicting the duration of new therapies, that use inhibitors that have similar structures or belong to the same class. It is also interesting to compare two or more mutational patterns to see if they share the same mutational pathways, which at the end will help to reduce the possibility of drug resistance.

To combat HIV infection several antiretroviral (ARV) drugs belonging to different drug classes that affect specific steps in the viral replication cycle have been developed. Antiretroviral therapy (ART) generally consists of well-defined combinations of three or four ARV drugs. Due to its remarkable variation capabilities, HIV can rapidly adapt to the selective pressure imposed by ART through the development of drug resistant mutations, that are fixed in the viral population within the host in known mutational pathways. The development of drug resistant viruses compromises HIV control, with the consequence of a further deterioration of the patient's immune system. Many of these ARV drug resistant mutations reduce HIV susceptibility to ARV drugs by themselves, while others need to accumulate in order to cause resistance.

3.1 RELATED WORK

There are several works describing computational models aimed to better understand HIV evolution and immunopathogenesis. A portion of these models is devoted to predict phenotypic HIV resistance to antiretroviral drugs using different approaches such as decision trees (Beerenwinkel et al., 2002) or neural networks (Draghici and Potter, 2003). Other works try to identify relevant associations between clinical variables and HIV disease (Ramirez et al., 2000). In (Chausa et al., 2009), association rules between clinical variables and the failure of the treatment are extracted. The results obtained are temporal rules that have as

Table 1: An example of the HIV patient data. It presents two patients P_1 with 3 temporal studies, and P_2 with two temporal studies.

Patient	Treatment	Mutations	Weeks
P_1	IDV, RTV	L90M, V82A	15
		I54V	45
		M46I	55
P_2	LPV,IDV, RTV	V82A	25
		I54V	45

antecedent the increasing of a subset of clinical variables and as consequent the failure of the treatment, given by side effects of the drugs or by the elevated viral count (unsuccessful therapy). None of the clinical variables considered are HIV mutations. Finally, in (Hernandez-Leal et al., 2011) TNBNs are used to analyze the temporal relationships between all the protease inhibitors and some high frequency mutations. In contrast, the present work uses two different temporal models: DBN and TNBN. Moreover, the experiments presented here are aimed to analyze specific and highly used antiretrovirals (IDV, APV, LPV, RTV), and its corresponding known resistance mutations in order to analyze the mutational pathways.

4 EXPERIMENTS

In this section we present the data used in the experiments, along with the selection method for the drugs and mutations. The first experiment presents the results using a DBN, while the second experiment uses a TNBN. Finally, we contrast the results and models obtained.

4.1 DATA AND PREPROCESSING

Data was gathered from the Stanford HIV Database (HIVDB) (Shafer, 2006) obtained from longitudinal treatment profiles reporting the evolution of mutations in individual sequences.

We retrieved data from patients with HIV subtype B. We choose to work with this subtype because it is the most common in America (Hemelaara et al., 2006), our geographical region of interest. For each patient data retrieved contains a history consisting of a variable number of studies. Information regarding each study consists of an initial treatment (cocktail of drugs) administered to the patient and the list of the most frequent mutations in the viral population within the host at different times (in weeks) after the initial treatment. An example of the data is presented in Table 1.

The number of studies available varies from 1 to 10 studies per patient history. Since we are interested in temporal evolution of the mutational networks, we filtered those patients having less than 2 studies. The filtered dataset consisted of approximately 300 patients.

Antiretrovirals are usually classified according to the enzyme that they target. We focus on protease as this is the smallest of the major enzymes in terms of number of aminoacids. For the experiments we used: Atazanavir (ATV), Lopinavir (LPV), Indinavir (IDV), Ritonavir (RTV). According to the expert's opinion ATV and LPV are the most commonly used antiretrovirals nowadays. IDV was selected because it shares mutational pathways with LPV, and RTV was selected because its frequently used in combination with the other three.

To define the target set of mutations of interest, we used the Major HIV Drug Resistance Mutations according to (Stanford University, 2012). The mutations selected for both experiments are: L90M, V82A, I54V, I84V, V32I, M46I, M46L, I47V, G48V.

Ir order to evaluate the models and to measure the statistical significance of edge strengths we used nonparametric bootstrapping. For this we obtained a reshuffled (re-sampling with replacement) dataset generated from the original dataset and learned the models from this new dataset; this procedure was repeated 10 times. Confidence in a particular directed edge is measured as a percentage of the number of times that edge actually appears in the set of reconstructed graphs. We used two thresholds for considering a relation as important. The first one is a strong relation that appears at least in 90% of the graphs, and the other is a suggestive relation, this occurs with values between 70 and 90%. In Figures 2(a)-2(b) a suggestive relation is shown as an arrow labeled with *, and a strong relationship is presented as an arrow label with **.

4.2 DYNAMIC BAYESIAN NETWORK

In order to obtain the corresponding DBN from the data we began by learning the structure of the static network. For this stage each variable in a patient record was seen to have a binary value, where this value was equal to 1 if that variable was present and 0 if not present. While there are many approaches for learning DBNs such as (Friedman, Murphy, and Russell, 1998; Wang, Yu, and Yao, 2006; Gao et al., 2007) we decided to use a simple approach, therefore the structure of the static network was learned by applying (Chow and Liu, 1968) from which a fully connected tree is obtained. Since the Chow-Liu algorithm does not provide the direction of the arcs, we subsequently applied (Rebane and Pearl, 1987) in conjunction with expert knowledge in order to obtain the final directed acyclic graph. We mention that Rebane and Pearl's

algorithm found the antiretroviral RTV and the mutation L90M to be parent nodes of the antiretroviral IDV; however this relation was found to be invalid and was not established since we know that a mutation cannot be a cause of a medication (expert knowledge). By establishing mutations as effects of medications the directions of the remaining arcs are easily determined.

To obtain the structure of the transition network each record was discretized into equal length time intervals, where the value of a variable was set to 1 if it was present during that time interval and 0 otherwise. Once a variable is observed it remained set to 1 for all subsequent time intervals. If a variable was observed at a time point between state changes, its value was set to 1 for all time intervals greater than the observed time. For our experiments we used different granularities: 5, 8,10 and 20, that corresponds to different number of weeks. The obtained structures were the same except for one new arc in one experiment. When learning the transition network, a node in a time slice can only choose its parents from the previous time slice. In order to choose the best set of parents we applied the Bayesian Information Criterion (BIC) scoring metric to evaluate each selection. This metric returns the probability of the data given the model penalizing the complexity of the model, in other words it favors simpler models. The learning of the transition network was done by using Kevin Murphy's Bayesian Network Toolbox for Matlab (Murphy and others, 2001). Figure 2(a) presents the resulting DBN.

From the model in Figure 2(a) we can observe, that all the nodes, except ATV, have persistent arcs between time slices. In the transition network, arcs from mutations in time slice t appear to be catalysts to the mutations they point to in time slice $t+1$. The static network provides more information on which antiretrovirals are the causing agents of certain mutations. For example, we can observe that the mutation L90M is a reaction to the IDV drug. By following the arcs in the static network and moving through the transition network we can begin to detect mutational pathways.

4.3 TEMPORAL NODES BAYESIAN NETWORKS

To apply the learning algorithm for the TNBN the data is arranged as a table where each column represents a drug or mutation and each row represents a patient case. For the drugs the values are USED or NOT USED, and for the mutations the values are: APPEAR, with the number of weeks in which the mutation appeared for the first time, or NOT, if the mutation did not appear in that case. Thus, the drugs are instantaneous nodes, and the mutations are temporal nodes of the TNBN.

We evaluated different orderings for the K2 algorithm. Specifically, we evaluated all the different combinations for the first two mutations and the order of the rest was chosen randomly. In Figure 2(b) the model with highest predictive accuracy is presented.

The model shows a relation between IDV and RTV, this may suggest that they are mainly used together. ATV is shown isolated from the rest. A reason for this may be that the number of cases that used this drug was low and the algorithm could not find any relations with other drugs or mutations.

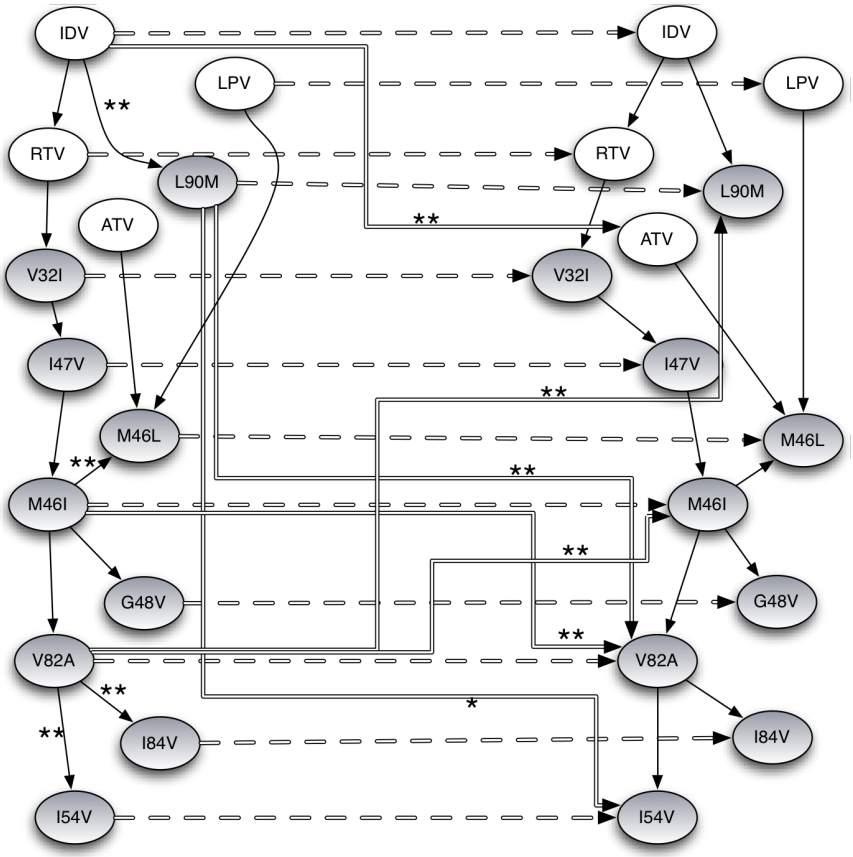
The mutations L90M, I54V and I84V appear to be the first mutations caused by the effects of the drugs. Mutation V82A appears to be important since it has three arcs directed to other mutations. In this model, the mutations M46L, I47V, V32I, V82A and M46I had only a causing mutation as parent. Finally, the mutation G48V appears isolated; this may happen due to the fact that this mutation was infrequent in the data.

4.4 CONTRASTING THE MODELS

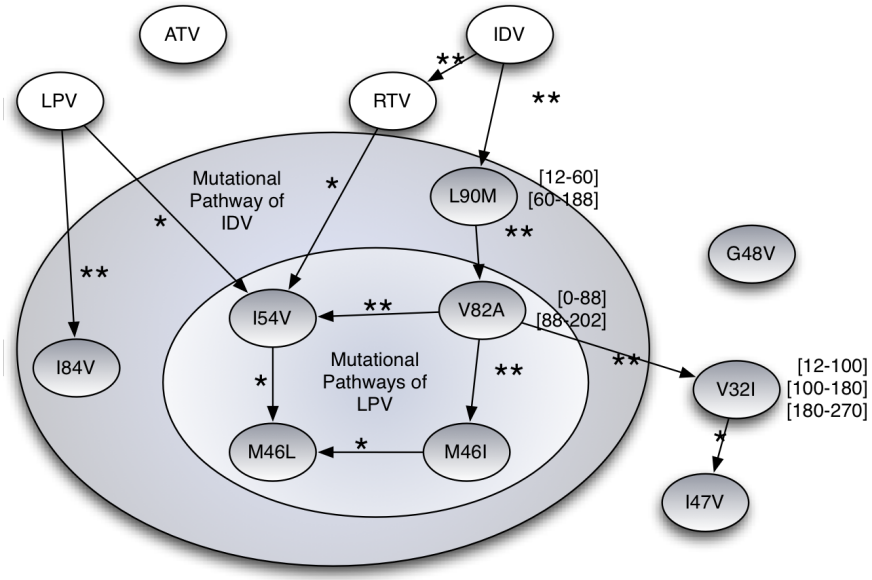
Structure

In order to compare the two models we begin by contrasting the structure displayed by each one. A DBN is typically represented as a 2-TBN in order to give a smaller representation and therefore inference for future times requires the network to be unrolled. Unlike the DBN, a TNBN only has one base network, which can be interpreted as the causal temporal relationships existing between random variables. In a TNBN there is no need to repeat the structure. Therefore, TNBNs offer a more compact representation than DBNs, as DBNs can grow to become increasingly complex, as they are further unrolled to include greater time intervals. Unrolling a DBN can also result in the repetition of nodes whose state has not changed, thus generating unnecessary replications that clutter the model.

The way in which a TNBN is constructed also provides us with different visual information. Given that a TNBN is learned using the K2, the nodes of the resulting model have a temporal ordering, and because the TNBN only consists of one base structure, order of occurrence between different variables is easily visualized. For example, in the context of HIV, pathways formed between mutations can be determined by following the directions of the arcs. In Figure 2(b) we can detect the pathway L90M→V82A→M46I. In contrast, in a DBN temporal orderings are more difficult to visualize solely from the model. However, DBNs offer their own distinct interpretation of the process being modeled. In DBNs, variables that are strongly related can be visualized from the arcs in both the static and



(a) A learned DBN model. Discontinuous lines represent persistent arcs.



(b) A learned TNBN model. Some intervals associated with their respective temporal nodes are shown.

Figure 2: The two temporal models: DBN and TNBN. White nodes represent drugs and grey nodes represent mutations of protease. An arc labeled with a * represents a suggestive relation. An arc with ** represents a strong relation.

transition networks. For example, in Figure 2(a), in time slice t arcs go from mutations L90M and M46I to V82A in time slice $t+1$, indicating a correlation among V82A and both of these other two mutations.

Bootstrapping results

The relations found after performing bootstrapping in the models can be seen in Figures 2(a)-2(b). Both models successfully detected several well known relations among mutations, and while they both coincide in many of these, each one also displays a set of unique relations found. For example, the TNBN detected V82A→V32I as a strong relation, this is consistent with the literature, but was not found by the bootstrapping preformed for the DBN. On the other hand, the DBN was able to detect the relation L90M→I54V; this relation is not present in the TNBN but exists in the literature.

Overall the TNBN was more successful at detecting mutational reactions to specific antiretrovirals. The mutational effects of the medications used are well documented and the TNBN reflects this knowledge. For example, RTV→I54V was found as a suggestive relation; however it is known that when RTV is taken as a booster in combination with IDV, I54V is a common mutational reaction. We note that the TNBN also displays the relation between IDV and RTV.

We also mention that not all relations found by the models have been previously reported. V82A→I84V (found in the DBN) and M46I→M46L (found in both) are as far as the authors know unreported. Further research is needed to determine the correctness of these relations.

Clinical relevance

Both learned models were capable of obtaining known mutational pathways. For example, it is known that the LPV drug causes the pathways:(i) M46I/L, I54V/T/A/S and V82T/F/S (Kempf et al., 2001) , and (ii) V32I, I47V/A, I50V, I54L/M and L76V (Nijhuis et al., 2007; Parkin, Chappey, and Petropoulos, 2003). For IDV the main mutations are V82A/T/F/S/M, M46I/L, I54V/T/A, I84V and L90M (Bélec et al., 2000; Descamps et al., 2005). Moreover M46I/L, I54V/T/A/S and V82T/F/S are reported as major mutations both to IDV and LPV and we can see that in Figures 2(a) and 2(b) the models were able to discover these shared pathways. These results suggest that the models could be applied to new ARVs with structural similarities to determine the duration of the treatment.

5 CONCLUSIONS

Mutational pathways provide important information for decision making in multidrug therapy. In our research, we used HIV data from several patients in order to analyze the temporal occurrence of mutations and create such mutational pathways. We present a comparison of the DBNs and TNBNs models created with this data. Even when Dynamic Bayesian Networks have become a standard for time series modeling, TNBNs offer different advantages. We show why they should be considered as an option when facing problems with dynamic information. Both models were able to capture pathways previously discovered by clinical experiments. These results suggest that temporal BNs are models that can have a significant impact in the battle against the HIV disease. For example, we could use these models to predict mutational pathways and how long new antiretrovirals can be used in specific cases. These models would also help physicians to follow up on patients that are undergoing a therapy that shares similar chemical properties with another treatment whose mutational pathways are already known. As future research, it would be interesting to compare two different cocktail treatments along with the temporal occurrence of drug resistant mutations, in order to predict the most effective treatment. We believe this could aid the experts in the selection of the best treatment for the patient.

Acknowledgements

We would like to thank Dr. Santiago Ávila-Rios and Dr. Gustavo Reyes-Terán from CIENI-INER for their valuable comments and suggestions.

References

Arroyo-Figueroa, G., and Sucar, L. E. 1999. A temporal Bayesian network for diagnosis and prediction. In *Proceedings of the 15th UAI Conference*, 13–22.

Beerenwinkel, N.; Schmidt, B.; Walter, H.; Kaiser, R.; Lengauer, T.; Hoffmann, D.; Korn, K.; and Selbig, J. 2002. Diversity and complexity of HIV-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences of the United States of America* 99(12):8271–8276.

Bélec, L.; Piketty, C.; Si-Mohamed, A.; Goujon, C.; Hallouin, M.; Cotigny, S.; Weiss, L.; and Kazatchkine, M. 2000. High Levels of Drug-Resistant Human Immunodeficiency Virus Variants in Patients Exhibiting Increasing CD4+ T Cell Counts Despite Virologic Failure of Protease Inhibitor Containing Antiretroviral Combi-

nation Therapy. *Journal of Infectious Diseases* 181(5):1808–1812.

Chausa, P.; Cáceres, C.; Sacchi, L.; León, A.; García, F.; Bellazzi, R.; and Gómez, E. 2009. Temporal Data Mining of HIV Registries: Results from a 25 Years Follow-Up. *Artificial Intelligence in Medicine* 56–60.

Chow, C., and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on* 14(3):462–467.

Cooper, G., and Herskovits, E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine learning* 9(4):309–347.

Dagum, P.; Galper, A.; and Horvitz, E. 1992. Dynamic network models for forecasting. In *Proceedings of the 8th Workshop UAI*, 41–48.

Descamps, D.; Joly, V.; Flandre, P.; Peytavin, G.; Meiffrédy, V.; Delarue, S.; Lastère, S.; Aboulker, J.; Yeni, P.; and Brun-Vézinet, F. 2005. Genotypic resistance analyses in nucleoside-pretreated patients failing an indinavir containing regimen: results from a randomized comparative trial. *Journal of clinical virology* 33(2):99–103.

Draghici, S., and Potter, R. B. 2003. Predicting HIV drug resistance with neural networks. *Bioinformatics* 19(1):98–107.

Freeman, S.; Herron, J.; and Payton, M. 1998. *Evolutionary analysis*. Prentice Hall Upper Saddle River, NJ.

Friedman, N.; Murphy, K.; and Russell, S. 1998. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 139–147. Morgan Kaufmann Publishers Inc.

Gao, S.; Xiao, Q.; Pan, Q.; and Li, Q. 2007. Learning dynamic bayesian networks structure based on bayesian optimization algorithm. *Advances in Neural Networks–ISNN 2007* 424–431.

Hemelaara, J.; Gouws, E.; Ghys, P. D.; and Osmanov, S. 2006. Global and regional distribution of HIV-1 genetic subtypes and recombinants in 2004. *AIDS* 20:W13–W23.

Hernandez-Leal, P.; Rios-Flores, A.; Ávila-Rios, S.; Reyes-Terán, G.; González, J.; Orihuela-Espina, F.; Morales, E.; and Sucar, L. 2011. Unveiling HIV mutational networks associated to pharmacological selective pressure: a temporal Bayesian approach. *Probabilistic Problem Solving in BioMedicine* 41.

Hernandez-Leal, P.; Sucar, L. E.; and Gonzalez, J. A. 2011. Learning temporal nodes Bayesian networks. In *The 24th Florida Artificial Intelligence Research Society Conference (FLAIRS-24)*.

Kempf, D.; Isaacson, J.; King, M.; Brun, S.; Xu, Y.; Real, K.; Bernstein, B.; Japour, A.; Sun, E.; and Rode, R. 2001. Identification of genotypic changes in human immunodeficiency virus protease that correlate with reduced susceptibility to the protease inhibitor lopinavir among viral isolates from protease inhibitor-experienced patients. *Journal of Virology* 75(16):7462–7469.

Murphy, K., et al. 2001. The bayes net toolbox for matlab. *Computing science and statistics* 33(2):1024–1034.

Nijhuis, M.; Wensing, A.; Bierman, W.; De Jong, D.; van Rooyen, W.; Kagan, R.; et al. 2007. A novel genetic pathway involving l76v and m46i leading to lopinavir/r resistance. *HIVDRW2007* 12:140.

Parkin, N.; Chappey, C.; and Petropoulos, C. 2003. Improving lopinavir genotype algorithm through phenotype correlations: novel mutation patterns and amprenavir cross-resistance. *Aids* 17(7):955.

Ramirez, J.; Cook, D.; Peterson, L.; and Peterson, D. 2000. Temporal pattern discovery in course-of-disease data. *Engineering in Medicine and Biology Magazine, IEEE* 19(4):63–71.

Rebane, G., and Pearl, J. 1987. The recovery of causal poly-trees from statistical data. In *Third Conference on Uncertainty in Artificial Intelligence (Vol. 87, pp. 222-228)*., volume 86, 222–228.

Shafer, R. 2006. Rationale and uses of a public hiv drug-resistance database. *Journal of Infectious Diseases* 194(Supplement 1):S51–S58.

Stanford University. 2012. Major HIV Drug Resistance Mutations. `http://hivdb.stanford.edu/pages/download/resistanceMutationshandout.pdf`.

Wang, H.; Yu, K.; and Yao, H. 2006. Learning dynamic bayesian networks using evolutionary mcmc. In *Computational Intelligence and Security, 2006 International Conference on*, volume 1, 45–50. IEEE.

Incorporating Metadata into Dynamic Topic Analysis

Tianxi Li
Stanford University
Stanford, CA 94305
tianxili@stanford.edu

Branislav Kveton
Technicolor
Palo Alto, CA 94301
Branislav.Kveton@technicolor.com

Yu Wu
Stanford University
Stanford, CA 94305
yuw2@stanford.edu

Ashwin Kashyap
Technicolor
Palo Alto, CA 94301
Ashwin.Kashyap@technicolor.com

Abstract

Everyday millions of blogs and micro-blogs are posted on the Internet These posts usually come with useful metadata, such as tags, authors, locations, etc. Much of these data are highly specific or personalized. Tracking the evolution of these data helps us to discover trending topics and users' interests, which are key factors in recommendation and advertisement placement systems. In this paper, we use topic models to analyze topic evolution in social media corpora with the help of metadata. Specifically, we propose a flexible dynamic topic model which can easily incorporate various type of metadata. Since our model adds negligible computation cost on the top of Latent Dirichlet Allocation, it can be implemented very efficiently. We test our model on both Twitter data and NIPS paper collection. The results show that our approach provides better performance in terms of held-out likelihood, yet still retains good interpretability.

1 Introduction

Topic evolution analysis has become increasingly important in recent years. Such analysis on social media and webpages could help people understand information spreading better. In addition, it also provides ways to understand latent patterns of corpus, reduce effective dimensionality and classify documents and data. Meanwhile, reseachers manage to fit various types of data into the topic model. For example, image segmentations was modeled as topics in Feifei et al. [6]. User behaviors were also modeled as topics as in Ahmed et al. [1] In such circumstances, topic evolution gains other practical values. For example, knowing the evolution of people's behaviors could improve the performance of item recommendations and advertising strategy. In addition, dynamic feature extraction might also provide richer user profile.

In various applications, one might want to harness metadata for different purposes. When metadata contains useful information for the topic analysis, it can help enhance the precision of the model. For instance, authorship can be used as an indicator of the topics in scientific paper analyzing [14]. Citations can also help reveal the paper's topics [9]. In behavior modeling, metadata such as user_id could be used for personalized analysis.

In this paper, we propose topic evolution model incorporating metadata effects, named metadata-incorporated dynamic topic model (mDTM). This is a flexible model effective for various metadata types and evolution patterns. We demonstrate its applicability by modeling the topic evolution of Twitter data, where we use hashtags as the metadata. This problem is particularly challenging because of the limited length of tweets and their non-standard webish style. Later we use authors as the metadata to run a dynamic author-interest analysis on the NIPS corpus.

The paper is organized as following. Section 2 gives a brief description of backgrounds and prior work. Our model is introduced in Section 3. Finally, the illustrative examples of topic evolution analysis are presented in Section 4.

2 Notations and Related Work

In this paper, the corpus is denoted by D, and each document d in corpus consists of N_d words. Each word

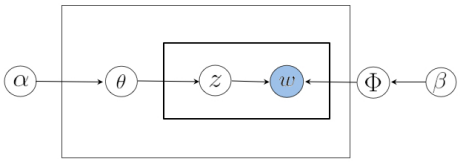
w is an element in the vocabulary of size V. There are K different topics associated with the corpus. Assume the words in the same document are exchangeable. The case of interests is when the documents have other special metadata. We use h to represent the metadata. Assume $h \in H$, where H is the domain of h. For instance, when h is hashtag of a tweet, H can be all the strings of hashtags. Let h_d be the instantiation of $h \in H$ at document d. Now with above notations, we can define the topics to be probability distributions over the vocabulary. Let $p(w|z)$ be the probability of word w appears when the topic is z, then topic z is represented by a V-vector corresponding to a multinomial distribution:

$$(p(1|z), p(2|z) \cdots, p(V|z)).$$

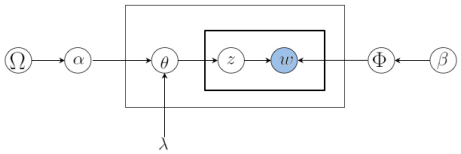
Latent Dirichlet Allocation proposed by Blei et al.[4], is one of the most popular models for topic analysis. LDA assumes the documents are generated by the following process:

(i) for each topic $k = 1, \cdots, K$:
Draw word distribution by $\phi_k \sim \text{Dir}(\beta)$.
(ii) for each document d in the corpus :
(a) Draw a vector of mixture proportion by $\theta_d \sim \text{Dir}(\alpha)$.
(b) for each word position j in d :
(b1): Draw a topic for the position by $z_{d,j} \sim \text{mult}(\theta_d)$.
(b2): Draw a word for the position by $w_{d,j} \sim \text{mult}(\phi_{z_{d,j}})$.

In the process, α is a K-vector and β is a V-vector. θ_d's are K-vectors characterizing a multinomial distribution of the topic mixture for each document d. α and β are called hyperparameters. Throughout this paper, we will use $w_{d,j}$ and $z_{d,j}$ to denote the word and topic in position j of document d respectively. $\text{Dir}(\alpha)$ denotes the Dirichlet distribution with parameter α, and $\text{mult}(\theta)$ denotes the 1-trial multinomial distribution.The model structure of LDA is shown in Figure 1(a), where we use Φ to represent the vectors $\{\phi_1 \cdots \phi_K\}$. In most cases, α and β are chosen to be symmetric vectors. There is work (Wallach et al.[16]) showing that LDA with asymmetric hyperparameters can outperform symmetric settings. For a K-vector $\Omega = (\Omega_1, \cdots, \Omega_K)$, they added the prior of α so as $\alpha \sim \text{Dir}(\Omega)$ can connect LDA to mixture model given by Hierarchical Dirichlet Process (HDP), which



(a) Original LDA graphical structure



(b) Asymmetric LDA with priors.

Figure 1: Graphical structures of LDA models.

is a nonparametric prior allocation process proposed in Teh et al.[15]. Adding the extra prior Ω, the graphical structure of LDA can be represent by Figure 1(b).

As mentioned in Section 1, we would like to take metadata into consideration as in [14]. Labeled-LDA (Ramage et al.[13]) provides another method to use metadata, requiring topics to be chosen from a subset of the label set, where the labels can incorporate certain kinds of metadata. Statistically speaking, this works like adding sparse mixture priors. In Ramage et al.[12], labeled-LDA is used for Twitter data. However, there is no natural way to create labels for different metadata. Such models assume specific generative process for metadata influences, which often limits the model to certain metadata. However, in our model, the impacts of metadata are modeled by empirical estimation rather than a specific probabilistic process, which makes it valid generally.

On the other hand, we need dynamic models to analyze topic evolution. The dynamic topic model (DTM) proposed by Blei works well on the example of *science* papers [3]. However, its logistic Gaussian assumption is no longer conjugate to multinomial distribution, which makes the computation inefficient. Moreover, it is an offline model that needs the entire corpus at one time, thus not suitable for stream data. Iwata et al.[10] uses multi-scale terms to incorporate time relation. This method can be very complicated in some cases and therefore infeasible for large scale datasets. But the

idea of modeling the relation by hyperparameter is really effective in many problems. In [1], a time-varying user model (TVUM) is proposed. It considers users' behaviors over time, and connects different users by the general sampling process. Here we can take a different viewpoint of TVUM. Note that when we take each user's identity as the metadata, TVUM is actually using metadata for interests evolution. In this aspect, it can be seen as a special case and also a starting point of our model.

In the next section, we begin from another view of LDA model, and generalize it to incorporate metadata.

3 Metadata-incorporated Dynamic Topic Model

3.1 Motivation: Define LDA via Markov Chains

The inference of LDA can be done through MCMC sampling. The sampling inference algorithm was proposed in Griffiths et al.[8]. But to understand how LDA works, we need to use the smoother version shown in Figure 1. It is shown in [15] that LDA in this case limits to a HDP mixture model as $K \to \infty$. Thus we will introduce a few more notations and start from HDP aspect of LDA. In the rest of the paper, we will use subscript d to denote relevant variables associated with document d, subscript k to denote the variable associated with topic k, and w to denote variables associated with word w. Following this style, m_k is defined as the number of documents containing words generated from topic k and $\mathbf{m} = (m_1, m_2, \cdots, m_K)$, while $n_{d,k,w}$ is the number of words w in document d that is from topic k. We further use $\cdot$ to denote summation over a specific variable, so $n_{\cdot,k,w}$ is the number of occurrence of words w being drawn from topic k and, $\mathbf{n}_k = (n_{\cdot,k,1}, n_{\cdot,k,2}, \cdots, n_{\cdot,k,V})$. In addition, $n_{d,k,\cdot}$ is the number of words in d which are associated with topic k. When we want to discuss the variables at time t, we use the superscript x^t to represent the variable x in the model of time t. So we have $\mathbf{m}^t = (m_1^t, m_2^t, \cdots, m_K^t)$,$\mathbf{n}_k^t = (n_{\cdot,k,1}^t, n_{\cdot,k,2}^t, \cdots, n_{\cdot,k,V}^t)$. When we focus on discussions at one time slice, which is clear in context, we will ignore the superscript t.

According to the discussion in [15] and the mechanism of Gibbs sampler, we can equivalently define the LDA inference of topic z (for each position of each document) as the stationary distribution of a Markov chain with the transition probability given in Formula (1), in which the superscript $-(d,j)$ refers to the originally defined variables without considering the position j of document d, and $\mathbf{w}^{-(d,j)}, \mathbf{z}^{-(d,j)}$ are the variables of the words and topics of the corpus except the ones at position j in document d, that is, $w_{d,j}$ and $z_{d,j}$ respectively.

The interpretation of this transition probability is that the Markov chain evolves with the following two patterns to arrive new topic states in the document. (i) Choose a topic proportional to the existing topics distribution within the document. This means it tends to keep the topic of each position consistent with the document contents. (ii) With certain probability, it might choose a topic ignoring the existing contents of the document. However, this choice is based on the popularity of topics over the entire corpus. This is a reasonable assumption in many circumstances, and we believe this could explain the power of LDA.

$$P(z_{d,j} = k|\mathbf{w}^{-(d,j)}, \mathbf{z}^{-(d,j)}) \propto (n_{d,k,\cdot}^{-(d,j)} + \lambda \frac{m_k + \Omega_k}{\sum m_k + \Omega_k}) P(w_{d,j}|\phi_k). \quad (1)$$

3.2 Generalization: mDTM

Assume the corpus has metadata h. Our basic assumption is that metadata is a good indicator of topics for each document. For example, a tweet with hashtag "#Microsoft" is much more likely to talk about technology rather than sports. Nearly all the previous works involving a certain type of metadata rely on this assumption. We first define the preferences of metadata over time as a vector function of t and h, $g(h,t) = (g_1(h,t), g_2(h,t), \cdots g_K(h,t))$. The kth element $g_k(h,t)$ is the preference of h to topic k at time t. Since we want to build a dynamic model for topic evolution, we can learn $g(h,t)$, and turn it into another impact on top of the evolutionary effects of β and Ω. Motivated by the definition of LDA given by (1), we define the mDTM inference at a fixed time slice to be the stationary distribution of a Markov chain with transition probability

$$P(z_{d,j} = k|\mathbf{w}^{-(d,j)}, \mathbf{z}^{-(d,j)} \propto (n_{d,k,\cdot}^{-(d,j)} + g_k(h_d,t) + \lambda \frac{m_k + \Omega_k^t}{\sum m_k + \Omega_k^t}) P(w_{d,j}|\phi_k^t). \quad (2)$$

The modification we make has exact effects that we want to incorporate into (1). In addition, this process

provided by mDTM is simple and does not incure too much computation, as shown in the Section 3.4. We only focus on the case where there is only one metadata variable in our discussion. There might be the case that more than one metadata variables are associated with the corpus. For instance, we might have timezone and browser for web log. In this case, we can simply model the effects as additive and estimate the function $g(h,t)$ separately for each metadata variable. Then everything we discuss here could be used for multiple metadata variable case. As we will propose different evolution patterns for the parameters in later sections, here we introduce notation f_Ω and f_β as the evolution functions of Ω and β. Now taking the time effects of evolution into consideration, the entire evolution process of mDTM is as follows:

(1) $t = 0$: initialize the model by LDA.
(2) For $t > 0$:
  (a) Draw Ω_t according to the model of $t-1$ by $\Omega_t = f_\Omega(t-1)$.
  (b) For each topic $k = 1, \cdots, K$: Draw β_k by $\beta_k^t = f_\beta(t-1)$.
  (c) With the current Ω_t and $\{\beta_k^t\}_{k=1}^K$, implement the inference for the process described by equation (2).

We model the evolution of all the effects by separable steps, so the model can be updated when data in new time slice arrives, which makes it possible for stream data processing and online inference. It is very flexible to adjust mDTM for different types of metadata, with different properties as we do not have to assume specific properties of the metadata. Notice that though we generalize the Markov chain definition of LDA to mDTM, we haven't shown the existence of the stationary distribution or limiting behavior of the chain. To address this issue, we can check mixing of the chain, so as to know if the inference is valid. In all of our experiments, such validity is observed. For details and methods about mixing behavior of Markov chains, we refer to Levin et al. (2009) [11].

The evolution patterns $f_\Omega(t)$, $f_\beta(t)$ and $g(h,t)$ are addressed in Section 3.3. Then we give the inference steps of mDTM in Section 3.4.

3.3 Evolution Patterns of mDTM

Now we describe how to model g. Assume metadata is categorical which is the case we normally encounter in applications. Similar methods can be used to choose f_Ω and f_β, so we will only discuss the evolution pattern for $g(h,t)$ in detail. We use $\tilde{n}_{k,h}^t$ to denote the number of the topic k that occurs in all documents having metadata h at time t.

3.3.1 Time-decay Weighted Evolution

We can just take g_k as the weighted average number of topics k appearing in documents with metadata h, using the weights decays over time. This represents our belief that the recent information is more useful to predict the preference. Thus,

$$g_k(h,t) = \sigma \sum_{s<t} \kappa^{t-s} \tilde{n}_{k,h}^s, \tag{3}$$

where σ is a scalar representing the influence of the metadata. This is a straightforward way to encode the evolution pattern, and the computation is very easy.

3.3.2 Bayesian Posterior Evolution

For each $h \in H$, we assume there is a preference vector for h to be $\mu_h^t = (\mu_{1,h}^t, \mu_{2,h}^t, \cdots \mu_{K,h}^t)$ which is a vector in the $K-1$ dimensional simplex, with $\mu_{k,h}^t \geq 0$ for $k = 1 \cdots K$. Then the realization of choosing topic for any $h \in H$ can be seen as $(\tilde{n}_{1,h}^t, \tilde{n}_{2,h}^t, \cdots \tilde{n}_{K,h}^t) \sim \text{Multinomial}(\tilde{n}_h^t, \mu_h^t)$, the $\tilde{n}_h^t$-trial multinomial distribution which is sum of $\tilde{n}_h^t$ independent trials from $\text{mult}(\mu_h^t)$, where $\tilde{n}_h^t$ is the total number of observations of h over the corpus. So we can take the Bayesian estimation by adding a Dirichlet prior by the process:

$$\mu_h^t \sim \text{Dir}(\zeta^{t-1} \cdot \hat{\mu}_h^{t-1})$$

$$(\tilde{n}_{1,h}^t, \tilde{n}_{2,h}^t, \cdots \tilde{n}_{K,h}^t) \sim \text{Multinomial}(\tilde{n}_h^t, \mu_h^t)$$

In such settings, we can choose the posterior expectation as the estimator, which is

$$\hat{\mu}_{k,h}^t = \frac{\tilde{n}_{k,h}^t + \zeta^{t-1} \cdot \hat{\mu}_{k,h}^{t-1}}{\sum \tilde{n}_{k,h}^t + \zeta^{t-1} \cdot \hat{\mu}_{k,h}^{t-1}}. \tag{4}$$

ζ is a scalar representing influence of the prior, which is the Bayesian estimator from previous time. Then let

$$g_k(h,t) = \sigma \hat{\mu}_{k,h}^t$$

in the process, where σ is a scalar representing the influence of the metadata. Such evolution pattern is very simple and smooth and it adds almost no additional computation cost.

This pattern actually also assumes there is a hyperparameter in each time t, which is μ_h^t. Rather than setting it beforehand, we impute the estimate for such hyperparameters by inference from the model. This is the idea of empirical Bayes method. In particular, one could notice that if there is no new data for h after time t, Bayesian posterior evolution would remain the same, while the time-decay evolution gradually shrinks g to zero.

3.3.3 Sparse Preference

In certain cases, we might constrain each document to only choose a small proportion of K topics. Our method to achieve this goal is to force sparsity on the topic choosing process. We can take the occasional appearance of most of the topics as noise, then implement a thresholding to denoise and get the true sparse preference. Define the function $S(a, \epsilon)$ as hard or soft thresholding operator where ϵ is the threshold. Then we can process each variate of the vector resulting from the previous evolution pattern by S, resulting a sparse vector. The soft and hard thresholding functions are defined respectively as

$$S_{\text{soft}}(a, \epsilon) = \text{sign}(a) \cdot \max\{|a| - \epsilon, 0\}$$

$$S_{\text{hard}}(a, \epsilon) = \text{sign}(a) \cdot I\{|a| > \epsilon\}$$

3.3.4 Choice of f_Ω and f_β

Similar evolution patterns for f_Ω and f_β can be chosen. With certain variable changed according to the settings. For f_Ω, one could use m_k^t to replace $\tilde{n}_{k,h}^t$ in (3) and (4). The evolution pattern of β can be derived via replacing $\tilde{n}_{k,h}^t$ in (3), (4) by $\mathbf{n}_k^t$.

3.4 Inference

As mentioned previously, mDTM can be seen as a generalization of TVUM. Suppose now we take user-ID as the only metadata which is categorical, and assume that each document belongs to a certain user-ID, then the parameters associated with each category of the metadata in mDTM become the parameters associated with a particular user. Furthermore, suppose that the documents are the browsing history of a user, then mDTM will be modeling the user's browsing behavior over time. In particular, if we use the time-decay average discussed in Section 3.3.1, the resulting model is equivalent to TVUM after some simple derivation [1]. This connection gives an vivid example about how to transform a specific problem into the settings of mDTM.

The time-varying relationship of mDTM can be represented by a separable term, thus we can incorporate the time-related term and the topic modeling for a fixed time separately. For a fixed time unit, the inference process by Gibbs sampling is easy to derive. Since the special case mentioned before is equivalent to TVUM, we derive the inference process by analogy to that shown in [1]. Suppose now we have the model in previous time $t-1$, the whole process for inference of t is as follows:

(i) Update the new hyperparameters Ω^t and β^t for time t according to the chosen evolution pattern.

(ii) Initially set the starting values. We could set the initial value of α as Ω^t. The initial values for the counts at time t, that is $m_k^t, n_{\cdot,k,w}^t, n_{d,k,\cdot}$, can be computed after randomly choosing topics for each documents and words.

(iii) For each document d, compute the $g(h_d, t)$ according to the chosen evolution pattern in Section 3.3. Then sample the topic for each word position j by the formula

$$P(z_{d,j} = k | w_{d,j}, others)$$
$$\propto (n_{d,k,\cdot}^{-(d,j)} + g_k(h_d, t) + \lambda\alpha_{d,k}) \cdot \frac{n_{\cdot,k,w_{d,j}}^{t,-(d,j)} + \beta_{k,w_{d,j}}^t}{\sum_{w=1}^V n_{\cdot,k,w}^{t,-(d,j)} + \beta_{k,w}^t}.$$

(iv) Sample m_k^t from the Antoniak distribution [2] for Dirichlet process.

(v) Sample α from $\text{Dir}(\mathbf{m}^t + \Omega^t)$. And repeat (iii)-(v).

4 Experiments

To illustrate the model, we conducted two experiments in which metadata is used for different purposes. We first use mDTM on Twitter data for topic analysis, in which we take hashtags as the metadata. In the second experiment, we fit our model on the NIPS paper corpus and try to find information for specific authors,

[1] Actually, TVUM has a slightly different way to define the evolution of Ω, which defines the average in different scales of time, such as daily, weekly and monthly average.

which we use as metadata. For conciseness, we mainly discuss the former in detail, because Twitter data is special and challenging for topic analysis. In the NIPS analysis, on top of the similar results as in previous dynamic models such as DTM, we can extract authors' interests evolution pattern, which would be the main result we present for that experiment.

4.1 Twitter Topic Analysis

4.1.1 Data and Model Settings

The Twitter data in the experiment is from the paper of Yang and Leskovec [18]. We use the English tweets from July 1st, 2009 to August 31st, 2009. For each of the first three days, we randomly sampled 200,000 tweets from the dataset. And around 100,000 tweets were sampled for each of the rest days. We considered the hashtags as the metadata in the experiment. After filtering stop words and ignoring all words appearing less than 10 times, a vocabulary of 12,000 words is selected by TF-IDF ranking. The number of topics was fixed at 50. In mDTM, time-decay weighted average was used for f_Ω and f_β. We simply set $\kappa = 0.3$. Bayesian posterior evolution was used for hashtag and soft-thresholding discussed in Section 3.3.3 was used for the evolution of $g(h_d, t)$. The parameters λ and ϵ are tuned according to the prediction performance in the first week, which is discussed in Section 4.1.4.

Our main interest is how topic popularity and contents change over time.

4.1.2 Topic Popularity Evolution

As can be seen from Equation (2), all the documents with different metadata share the common term $\mathbf{m}$, thus $\mathbf{m}$ can be interpreted as community popularity of topics, separated from the specific preference of metadata. This shows which topics are more popular on Twitter. Figure 2 gives popularity over the two months of some topics, which we labeled manually after checking the word distributions of the topics.

4.1.3 Topic Contents Evolution

Since each topic is represented by a multinomial distribution, one could find out the important words of the topics. Table 1 gives the content evolution of the topic *US politics*. It can be seen that *obama* and *tcot*[2] are very important words. However, words about "Sarah Palin" were mainly popular in July, while the words about "Kennedy" and "Glenn Beck" became popular only at the end of August, all of which roughly match the pattern of search frequencies given by Google Trends[3].

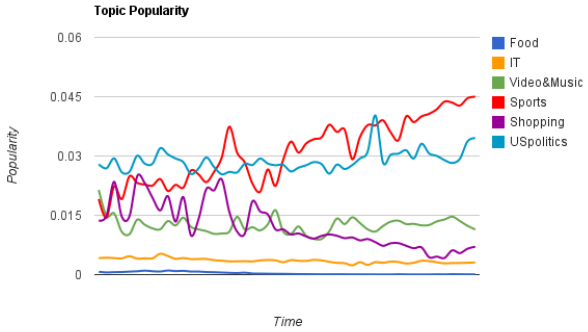


Figure 2: Topic Popularity on Twitter given by mDTM, over the period of July and August 2009.

Table 1: Content evolution of the topic *US politics*

Jul 4	Jul 27	Aug 12	Aug 30
palin	obama	health	kennedy
obama	palin	care	care
sarah	tcot	obama	ted
tcot	sarah	tcot	health
president	president	bill	obama
alaska	healthcare	healthcare	bill
al	health	reform	beck
honduras	obamas	insurance	glenn
governor	speech	president	public
palins	alaska	town	president

4.1.4 Generality Performance

There is no standard method to evaluate dynamic topic models, thus we take a similar approach as in [3] to show the prediction performance on the held-out data. In each day, we treat the next day's data as the held-out data and measure the prediction power of the model.

We compare mDTM with two LDA models without metadata as in [16] to illustrate the improvement provided by metadata modeling[4]. Without metadata, in the first model, we use LDA on the data of each day for inference, and call this model indLDA. The prob-

[2]The word *tcot represents "top conservatives on twitter"*.

[3]We don't provide the results from Google Trends due to the limited space. The search frequencies can be found at www.google.com/trends/

[4]We didn't compare directly with DTM. This is because DTM cannot be used in an online way, thus it cannot serve our purpose.

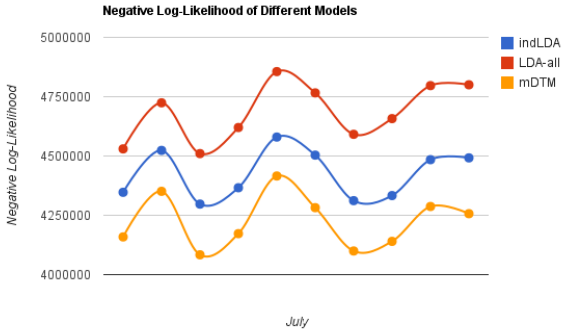


Figure 3: Negative log-likelihood during the early period (July 4th - 10th).

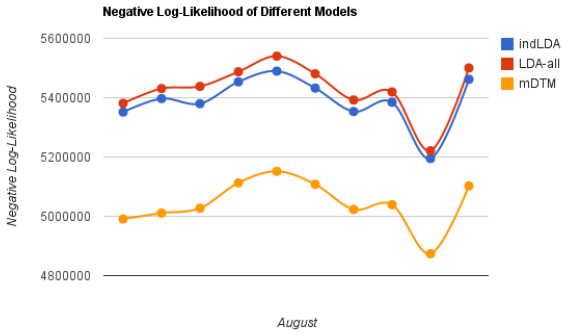


Figure 4: Negative log-likelihood during the end period (Aug 21st - 30th).

lem here is that there is no clear association for topics between days. In the second one, we try to overcame this drawback and take all the data of previous days for inference, which we call LDA-all. It would take nearly two months' data at the end of the period. This would be too much for computation. Thus we further subsampled the data from previous days for LDA-all in the end of the period to make it feasible. LDA-all will not serve for our purpose and so the main interests would be comparing indLDA and mDTM. We report the negative log-likelihood on the held-out data computed as discussed by Wallach et al[17] over the beginning period (July 4th - 10th) and the end period (Aug 21st - 30th). We estimate mDTM as discussed before, but computed the negative log-likelihood ignoring the metadata of the held-out data, thus this gives us an idea of how metadata can improve the modeling for general documents, even those without metadata. There is λ in all of the three models. We tune it and the thresholding parameter ϵ by achieving the best log-likelihood in the first week. Figure 3 and 4 illustrate the results.

As is shown, mDTM always performs better than the other two models. This is not surprising because mDTM has more flexible priors. It is interesting that LDA-all performs even worse than indLDA. This is different from the results of [3]. It might be explained by the differences between Twitter data and scientific paper data. Twitter's topic changes so frequently, but LDA-all takes all the previous days together, which undermines its power.

4.1.5 Effects of Metadata

In Twitter analysis, the topic preference of a specific hashtag is not of interests. However, incorporating hashtags can improve the preformance. On average, there are roughly 10 precent of the tweets having hashtags. But such a small proportion of metadata is able to provide important improvement of the whole corpus, even for the tweets without hashtags. We compute the held-out log-likelihood, for both the model inferred without using hashtags as metadata (called DTM_noTag) and the model mDTM using hashtags. mDTM_noTag can be seen as TVUM with one user. Note that when compute the held-out log-likelihood. We take the improvement of hashtags as the improvement of negative log-likelihood

$$(-\text{loglik})_{\text{DTM\_noTag}} - (-\text{loglik})_{\text{mDTM}}.$$

Figure 5 illustrates the improvement of negative log-ikelihood on the held-out data over the period. It can be seen that on average, incorporating hashtags as metadata does improve the performance. And this improvement tends to grow as time goes. This might results from the better estimation of most of the metadata preference.

4.1.6 Running Times

Here we present a comparison for timing of mDTM and indLDA. Both were implemented in C++, running under Ubuntu 10.04, with Quad core AMD Opteron Processor and 64 GB RAM. We list average running times (rounded) in Table 2. indLDA is the average time on 10 days (July 4th - July 13th) with 600 sampling iterations each day. mDTM-1 is the mDTM running on the same data with 600 sampling iterations. Since mDTM could inherit information from previous time, we found 300 iterations (or less) are enough for valid inference. Thus we use mDTM-2 to denote mDTM with 300 it-

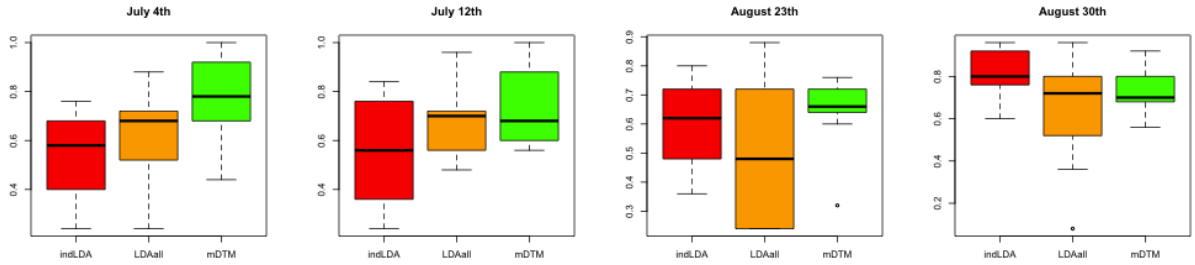


Figure 6: The human evaluation ACR for the three models. Each box is a value distribution of average correct ratios for 10 topics of the corresponding model on certain day.

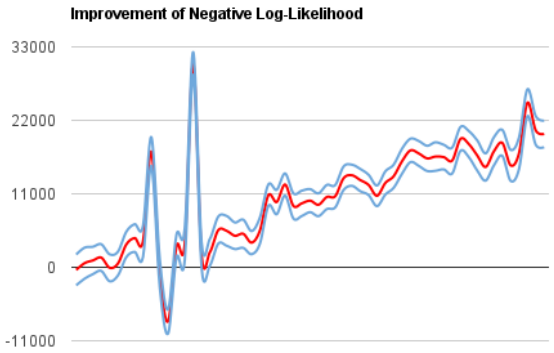


Figure 5: The improvement of negative log-likelihood via hashtags over the period. The red lines are the improvement of $-log(likelihood)$ computed by importance sampling. The blue lines are the intervals at each estimation point given by 2 standard deviations of the sampling.

erations. It can be seen that mDTM is much faster than LDA.

Table 2: Running times of three different models

indLDA	mDTM-1	mDTM-2
58min 41s	67min 13s	39min 24s

4.1.7 Interpretability

The previous sections show that mDTM is better than indLDA and LDA-all at generality. However, the interpretability of the topics is also of interests. Chang et al. [5] revealed that models with better performance on held-out likelihood might have poor interpretability. Here we use the method in [5] to ask humans to evaluate the interpretability. We choose July 4th (the first day after three initial days), July 11th (after one week of July 4th), August 30th (the last day) and August 23th (one week before the end) for experiments. However, news would be difficult for people to recognize after more than one year, so we only chose 10 stable topics from each model[5]. For every topic in each model, we construct the list by permuting top 15 words for that topic together with 5 intruder words which have low probability in that topic but high probability in some other topics. Suppose we have S subjects, then for each topic k, we compute the average correct ratio (ACR)

$$ACR(k) = \sum_{s=1}^{S} C(s,k)/(5S),$$

where $C(s,k)$ is the number of correct intruders chosen by subject s for topic k. We conducted a human evaluation experiment on Mechanical Turk with 150 subjects in total. Figure 6 shows the boxplot of ACR distribution within each model on each day.

It can be seen that mDTM does not lose much interpretability despite its better prediction performance, which is different from the observations in [5]. We hypothesize that this is due to the impacts of metadata.

4.2 NIPS Topic Analysis

In this section, we illustrate a different application of mDTM, that is, to extract specific information of metadata.

[5] We count the number of different words in the top 20 words list on two consecutive days, and sum such numbers during the whole period together. A larger sum number means that the topic word list changes frequently. Then we select 10 topics that are the most stable. The topics in different time are not associated for indLDA and LDA-all. We connect a pair of topics between two consecutive days if they have the most overlap on top 20 words.

4.2.1 Data and Model Settings

The dataset for this experiment contains the text file of NIPS conference from 1987 to 2003 in Globerson et al[7][6]. We only use the text of the paper and take the authors as the metadata. The papers in 1987-1990 were used for the first time unit to initiate the model, and each year after that was taken as a new time unit. The preprocessing details of the data can be found on the website. We further deleted all the numbers and a few stop words. The resulting vocabulary has 10,005 words. The number of topics K was set as 80. Bayesian posterior evolution was used for g and f_β. And f_Ω was set as time-decay weighted average with $\kappa = 0.3$. We don't use sparse preference in this example. The parameter λ is again tuned by log-likelihood as before.

4.2.2 Author-topic interests

As before, we could see the topic contents and popularity trends over time. Here, we only focus on the special information given by metadata in this experiment. When taking authors as metadata, an interesting information result provided by mDTM is the interests of authors, similar to the results of [14]. Figure 7 shows the results given by mDTM for author "Jordan_M". The height of the red bars represents the $\hat{\mu}_{k,h}$ from Equation (4) for h="Jordan_M", which can be interpreted as the topic interests according to the past information.

It can be seen that authors' favorite topics remained nearly the same during the three years, though the interest level for individual topics varied. When we know the topic interests of the author, we can further investigate the contents of the user's favorite topics, which is a way to detect the user's interests that would be useful in many applications. Table 3 shows the top 10 words for four topics of significant interests to "Jordan_M" in 1999, according to the result in Figure 7. We can roughly see they are mainly about "clustering methods", "common descriptive terms", "graphical models" and "mixture models & density estimation", which is a reasonable approximation.

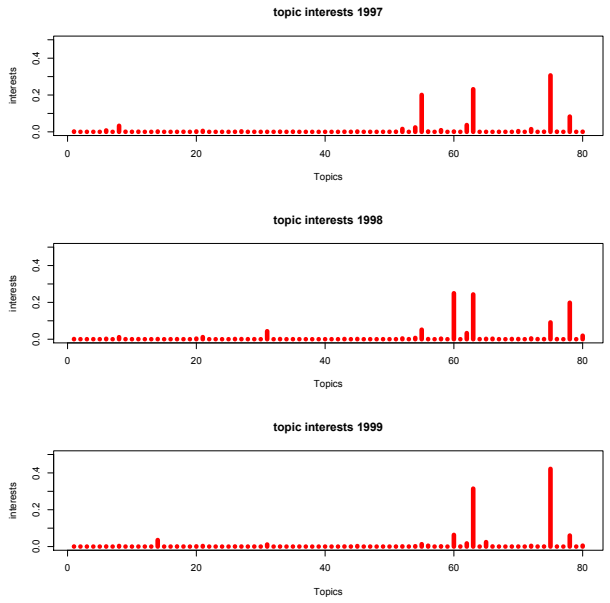


Figure 7: Topic preference from mDTM of 80 topics, for author "Jordan_M" in 1997, 1998 and 1999.

Topic 60	Topic 63	Topic 75	Topic 78
clustering	function	variational	model
clusters	number	nodes	data
information	figure	networks	models
data	results	inference	parameters
algorithm	set	gaussian	likelihood
cluster	data	graphical	mixture
feature	case	field	distribution
selection	based	conditional	log
risk	model	jordan	em
partition	problem	node	gaussian

Table 3: Four significant topics for "Jordan_M" selected from Figure 7 in 1999.

5 Conclusion

In this paper, we have developed a topic evolution model that incorporats metadata impacts. Flexible evolution patterns are proposed, which can be chosen according to properties of data and the applications. We also demonstrate the use of the model on Twitter data and NIPS data, revealing its advantage with respect to generality, computation and interpretability.

The work can be extended in many new ways. For the moment, it cannot model the birth and death of topics. One way to solve this problem is to use general prior allocation mechanism such as HDP. There has been work using this idea for static models. In addition, the generality and flexibility of mDTM make it possible to build other evolution patterns for hyperparameters, which might be more suitable for specific purposes of modeling.

[6]Data can be found at http://ai.stanford.edu/ gal/data.html

References

[1] A. Ahmed, Y. Low, M. Aly, V. Josifovski, and A. J. Smola. Scalable distributed inference of dynamic user interests for behavioral targeting. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 114–122, New York, NY, USA, 2011. ACM.

[2] C. E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[3] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM.

[4] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[5] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, 2009.

[6] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *CVPR*, pages 524–531, 2005.

[7] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, 8:2265–2295, 2007.

[8] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, Apr. 2004.

[9] Q. He, B. Chen, J. Pei, B. Qiu, P. Mitra, and L. Giles. Detecting topic evolution in scientific literature: how can citations help? In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 957–966, New York, NY, USA, 2009. ACM.

[10] T. Iwata, T. Yamada, Y. Sakurai, and N. Ueda. Online multiscale dynamic topic models. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 663–672, New York, NY, USA, 2010. ACM.

[11] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2009.

[12] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *ICWSM*, 2010.

[13] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore, August 2009. Association for Computational Linguistics.

[14] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press.

[15] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[16] H. Wallach, D. Mimno, and A. McCallum. Rethinking lda: Why priors matter. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1973–1981. 2009.

[17] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1105–1112, New York, NY, USA, 2009. ACM.

[18] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 177–186, New York, NY, USA, 2011. ACM.

Reactive Bayesian Network Computation using Feedback Control: An Empirical Study

Ole J. Mengshoel, Abe Ishihara, and Erik Reed
Carnegie Mellon University
Moffett Field, CA 94035
{ole.mengshoel, abe.ishihara, erik.reed}@sv.cmu.edu

Abstract

This paper investigates the challenge of integrating intelligent systems into varying computational platforms and application mixes while providing reactive (or soft real-time) response. We integrate Bayesian network computation with feedback control, thereby achieving our reactive objective. As a case study we investigate fault diagnosis using Bayesian networks. While we consider the likelihood weighting and junction tree propagation Bayesian network inference algorithms in some detail, we hypothesize that the techniques developed can be broadly applied to achieve reactive intelligent systems. In this paper's empirical study we demonstrate reactive fault diagnosis for an electrical power system.

1 INTRODUCTION

Substantial progress has been made over the last few decades in the areas of learning and reasoning using Bayesian networks (BNs) [29]. While most BN inference problems are computationally hard (NP-hard or worse) in the general case [6, 33, 28], efficient algorithms have been developed and demonstrated in a wide range of automated reasoning applications including model-based diagnosis [19, 24, 32, 31].

There has recently been an explosion in the number and types of computers available, capable of running implementations of sophisticated algorithms including BN-based algorithms. This is a result of the advancements in both hardware and software platforms now powering computers, which range from smart phones and tablet PCs to high-end gaming machines and high-performance computing (HPC) clusters.

The proliferation of computers of highly varying capability provides new opportunities for AI systems and also raises several interesting research questions [21]. Can BN-based algorithms be implemented, adapted, or "wrapped" such that they reactively (or in soft real-time) compute meaningful results across a myriad of commercially available computers? Can algorithms be optimized such that users may generally continue to operate their computers normally while also running AI systems–that is, continue to check email, chat, and partake in social networking?

To start answering these questions, this paper develops an architecture that integrates Bayesian network computation and feedback control. The architecture contains two feedback loops, a lower-level inner loop and a higher-level outer loop. What drives the inner loop is the difference between desired completion time (or setpoint, set to achieve reactivity) and actual completion time. The goal of the outer loop is to trade off between higher-level issues such as speed of inference, resource allocation, and accuracy.

While the architecture is general, we have for validation purposes used experimental data from an electrical power network located at the NASA Ames Research Center [30]. We demonstrate our novel approach in the area of fault diagnosis for this electrical power system, and investigate the BN inference algorithms of likelihood weighting [34], variable elimination [20, 9], junction tree propagation [18, 35], and belief propagation [29, 26]. Results for five different computers and three different operating systems are demonstrated. We show system identification results for both junction tree propagation and likelihood weighting, and demonstrate outer loop control by successfully changing the setpoint and BN inference algorithm employed.

By integrating control theoretic techniques and uncertainty processing using BNs, this research aims to: improve the reactivity and adaptability of uncertainty processing in different applications; improve the timeliness of computation under dynamic workloads

on varying computational platforms (smart phones, GPUs, multi-core CPUs, Hadoop clusters, ...); and reduce the effort and cost associated with developing and deploying BN-based software applications. Our goals are similar to those of anytime algorithms [8, 37, 3], which provide a way to trade off between computation time and solution quality. However, our feedback approach is more general in that it handles both anytime and non-anytime algorithms. In fact, we show experimentally how our approach enables us to gracefully switch, on-line, from a non-anytime algorithm (e.g., junction tree propagation) to an anytime algorithm (e.g., likelihood weighting) while achieving real-time response.

We are investigating control theory as it applies to software and our actuators are computational actuators rather than control surfaces on an aircraft or wheels on a robotic vehicle. For example, our actuators may change the number of processes being handled by a computer or the input parameters to C or Java programs that implement algorithms for BN computation. In contrast, the great majority of previous control theory research has, with several notable exceptions in computing and software [12, 2], focused on physical systems governed by Newtonian mechanics. In particular, we know of no similar work in the area of BN computation, including computation for diagnosis or prognosis, where control theory is applied.

The approach developed in this paper applies in situations where the computational load on a computer running AI (here, BN) applications is a key concern and where, simultaneously, we can prioritize and control the computational processes. Specifically, we partition computational processes into *high-criticality*, *medium-criticality*, and *low-criticality*; the high-critical processes are reactive. Our experiments are in the area of electrical power system diagnosis. However, the techniques are more widely applicable and we now outline a few areas where this approach may prove powerful. Examples of high-criticality tasks in which Bayesian networks (and similar probabilistic graphical models) can be used and for which one might want to provide a reactive response include medical monitoring, sensor fusion, speech recognition on a smartphone, gesture recognition, and video recognition. Examples of low-criticality tasks, which would be down-prioritized, suspended or killed by our approach (depending on circumstances) include virus scanning, backups, disk defragmentation, software updates, and volunteer computing (such as Seti@Home and Folding@Home).

What these applications have in common is that reactivity is important for some computational processes. However, reactivity is not important enough to warrant the use of a hard real-time operating system, which is often used in aerospace and other applications where safety is paramount. Our approach is tailored to soft real-time settings where low-criticality tasks can, if needed, be down-prioritized, suspended or terminated in order to provide reactive response for high-criticality tasks.

The rest of this paper is organized as follows. In Section 2 we provide background on Bayesian network computation, feedback control, and anytime algorithms. Section 3 discusses our architecture integrating feedback control and BN inference algorithms. In Section 4 we present the experimental setting and electrical power system data, while Section 5 discusses empirical results. We conclude and outline future research in Section 6.

2 PRELIMINARIES

2.1 BAYESIAN NETWORK INFERENCE

BNs represent multivariate probability distributions and are used for reasoning and learning under uncertainty [29]. Probability theory and graph theory form the basis of BNs: Roughly speaking, random variables are represented as nodes in a directed acyclic graph (DAG), while conditional dependencies are represented as graph edges. Each node is parameterized by a conditional probability density or distribution. A BN is a compact representation of a joint probability distribution if its graph is relatively sparse.

Formally, we let $\boldsymbol{X}$ be the BN nodes, $\boldsymbol{E} \subset \boldsymbol{X}$ the evidence nodes, and $\boldsymbol{e}$ the evidence. While the nodes $\boldsymbol{X}$ can represent either discrete and continuous random variables, we focus in this paper on the discrete case in which a node $X \in \boldsymbol{X}$ has a finite number of states $\Omega(X) = \{x_1, \ldots x_m\}$. A BN factorizes a joint distribution $\Pr(\boldsymbol{X})$, and allows for different probabilistic queries to be formulated and supported by efficient algorithms; they all assume that nodes in $\boldsymbol{E}$ are clamped to values $\boldsymbol{e}$. Considering the remaining nodes $\boldsymbol{R} = \boldsymbol{X} - \boldsymbol{E}$, the probabilistic queries are with respect to the posterior distribution $P(\boldsymbol{R} \mid \boldsymbol{e})$. Specifically, computation of most probable explanations (MPEs) amounts to finding an MPE over $\boldsymbol{R}$, or $\text{MPE}(\boldsymbol{e})$. Computation of marginals (or beliefs) amounts to inferring for one or more query nodes $\boldsymbol{Q} \subseteq \boldsymbol{R}$, where $Q \in \boldsymbol{Q}$, the posterior probabilities $\Pr(Q \mid \boldsymbol{e})$ which we denote $\text{BEL}(\boldsymbol{Q}, \boldsymbol{e})$. In diagnosis using BNs [19, 24, 32, 31], the terminology health nodes $\boldsymbol{H}$, where $\boldsymbol{Q} = \boldsymbol{H}$, is often used. By picking, for each $Q \in \boldsymbol{Q}$, a most likely state $q \in \Omega(Q)$, we obtain the most likely states $\text{MLS}(\boldsymbol{Q}, \boldsymbol{e})$ from $\text{BEL}(\boldsymbol{Q}, \boldsymbol{e})$. Computation of the maximum a posteriori probability (MAP) generalizes MPE computa-

tion and finds a most probable instantiation over nodes $\boldsymbol{Q} \subseteq \boldsymbol{R}$, MAP($\boldsymbol{Q}, \boldsymbol{e}$). MAP can be approximated using MPE and MLS; these two approximations are of interest because of the greater computational complexity of MAP [28] compared to MPE and BEL [6].

Different BN inference algorithms can be used to perform the above BEL, MPE, and MAP computations. We distinguish between exact and inexact algorithms. Exact algorithms for BEL and MPE computation include belief propagation in singly connected BNs [29], junction tree propagation [18, 35], conditioning [29, 14], variable elimination [20, 9], and arithmetic circuit evaluation [7, 4]. Inexact algorithms such as loopy belief propagation[1] [29, 26] and likelihood weighting [34] have been used to compute marginals; they have also been used to compute MPEs [16, 25] and MAPs [28]. In this paper we focus on computation of marginals; however the framework also applies to other queries including MPE and MAP.

2.2 ANYTIME REASONING

An anytime algorithm improves its solutions according to the computational resource allocated to it, and returns an (approximate) answer if interrupted [8]. Fundamentally, this iterative improvement algorithm framework provides a way to trade off between computation time and solution quality. Anytime algorithms are a useful tool for real-time system design, and there are also results on the composition of anytime algorithms [37]. Originally, the anytime approach was developed for single agents, recently it was extended to handle multiple agents [3].

Among BN inference algorithms, stochastic local search [16, 28, 22, 25], likelihood weighting [34], loopy belief propagation [29, 26], and conditioning [29, 14] can be considered anytime algorithms. However, many important and high-performing algorithms—including junction tree propagation [18, 35], variable elimination [20, 9], and arithmetic circuit evaluation [7, 4]—are unfortunately not anytime algorithms.

2.3 FEEDBACK CONTROL

Feedback control involves the manipulation of a system in which data, either measured or estimated, is transformed and utilized to modify the behavior of the system. This behavior is typically defined in terms of metrics such as stability, boundedness, response time, or over-shoot. Improving such metrics by means of feedback control is typically motivated by the desire to achieve some higher level objective such as ensuring a comfortable ride on a passenger transport jet, an average production rate on a robotic assembly line, or a maximal duration of a credit card transaction.

Recently, control theory has been applied to computing systems including control of HTTP servers [11], email servers [27], quality of service assurance [36], internet traffic control [13], and load balancing. Typically, computing systems are different from traditional feedback control applications in robotics and aircraft. First, modeling of the plant does not typically start from Newtonian mechanics; rather it often begins with a black box approach. Second, actuation can in some cases be almost instantaneous, such as flipping a bit, or writing a short integer to memory, i.e. specifying the maximum number of connections to a server. Third, measurements are often non-noisy but delayed. In analog sensing, filtering is used to remove noise, and at the same time can introduce significant (and unwanted) phase lag. However, in computing systems, a more difficult delay appears at the measurement. In applications such as control of an email server, the (discrete) delay is associated with the completion of a Remote Procedure Call (RPC). This delay is usually not known. Finally, disturbances can significantly impact performance. Take the example of the IBM Domino server [12]: Certain combinations of requests, made independently by different users impact CPU utilization in a nonlinear manner; this can be regarded as a stochastic disturbance. These disturbances, which may depend on time of day and day of week, can have a significant impact.

3 CONTROLLING BAYESIAN NETWORK COMPUTATION

Our goal is to support the application of both non-anytime and anytime BN inference algorithms in reactive settings by introducing techniques from feedback control. A key concept in feedback control is the *plant*. Here, the plant to be controlled is a dynamic system operating in the discrete time domain. This definition encompasses computers including mobile phones, tablets, laptops, desktops, and multi-core systems. In this paper, the plant is a digital computer performing a high-criticality (or *reactive*) process. The reactive process runs, in our case, a BN used for diagnosis. We also assume that on this computer, low- and medium-criticality (or *background*) processes are running and are competing for CPU cycles, memory, and other computer resources. The impact that these background processes have on the reactive process and its ultimate outcome (detection of a failure event) depends on a number of parameters such as operating

[1] Pearl's main emphasis was originally on exact propagation in singly connected BNs, however he also mentioned inexact propagation in arbitrary BNs [29, Exercise 4.7],

system, RAM, processor type and clock-speed, CPU cache, etc. Since hardware and software vary immensely, this represents a source of significant uncertainty and presents challenges for both modeling and control.

We introduce the control system framework illustrated in Figure 1. The framework supports both anytime (and inexact) as well as non-anytime (and typically exact) algorithms, and distinguishes between inner and outer control loops because their objectives differ.

Inner Loop: The goal of the inner loop, which is traditional to feedback control, is to make careful adjustments according to the parameters set by the outer loop. These are key parameters in our integrated approach: $r(k)$ is desired completion time (or *Setpoint*) for sample time k. We would like the computational process to finish within this time. $y(k)$ is the actual completion time (or just *Actual*) for sample time k. $e(k) = r(k)$ - $y(k)$ is the error signal at time k; $u(k)$ is the maximal number of low-criticality processes (or *Max Processes*). $u(k)$, also known as the control law, is taken to be a proportional-derivative controller [12].

We partition computational processes into *high-criticality* (and having an $r(k)$ value associated with them), *medium-criticality*, and *low-criticality* (subject to actuation by control system). High-criticality processes are reactive and essential. Medium-criticality processes are non-reactive but essential. Low-criticality processes are non-reactive and non-essential. The actual (or measured) number of low-criticality processes (or *Actual Processes*) at sample time k is denoted $v(k)$.

Our approach uses simple black-box prediction techniques, as detailed below. It is the combination of this black-box approach with the use of a desired computation time (setpoint $r(k)$ in Figure 1) and an actual computation time (output $y(k)$ in Figure 1) that makes it work. The actual computation time is just a measurement of how long a BN computation took. The desired computation time depends on the frequency of our reactive process and other factors. For example, a frequency of 10 Hz means that the desired computation time is upper-bounded by 100 milliseconds.

Outer Loop: The goal of the outer loop is to jointly optimize speed of inference, resource allocation, accuracy, and other factors. Unlike inexact anytime algorithms, which have a similar objective, our approach can use exact algorithms. If, for example, the exact junction tree propagation and variable elimination algorithms are employed, reduced accuracy is not an issue. However, accuracy is in general important, and accuracy versus computation time trade-offs are performed in the outer loop of the framework. For inexact

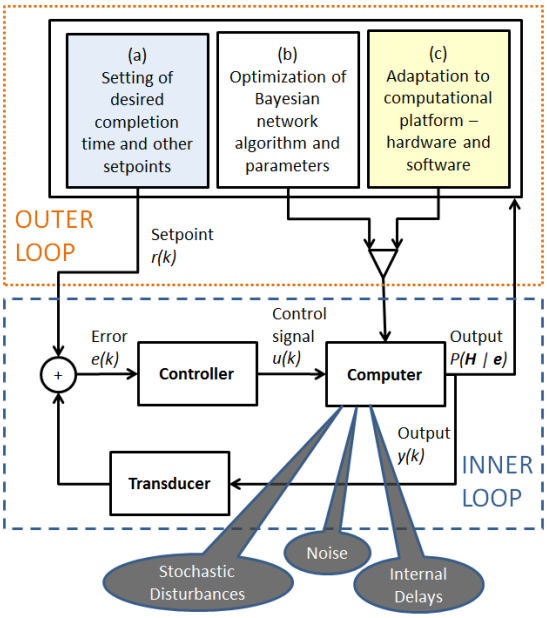


Figure 1: Our integrated architecture, where we wrap Bayesian network computation into two feedback control loops, a traditional inner loop where the Controller controls a Plant (here, a Computer) and a higher-level outer loop. While there are several options, this paper uses desired and actual computation time for setpoint $r(k)$ and output $y(k)$ respectively.

particle-based algorithms such as likelihood weighting and particle filtering, $p(k)$—the number of particles—is an example of such a parameter. One would like to use a very large number of particles for simulation, since this improves the accuracy of the estimate of the posterior $P(\boldsymbol{H}(k) \mid \boldsymbol{e}(k))$, however this may take too much time and one needs to carefully restrict the number of particles. In most experimental results with likelihood weighting reported in this paper, we assume that $p(k) = p$ and $r(k) = r$ are fixed (or stationary).[2]

4 METHODS AND DATA

System health management, which may utilize Bayesian network, can integrate information from heterogenous sensors and perform computation for the purpose of fault diagnosis, prognosis, and mitigation [19, 32, 31, 5]. Electrical power system fault diagnosis using Bayesian networks is the main focus of the experiments in this paper.

Data and Bayesian Network: We used experimental data from a real-world electrical power network, known as ADAPT, located at the NASA Ames Re-

[2] However, in Section 5.3 (see Figure 6) we discuss how and why $r(k)$ can be changed over time.

Computer	OS	Data Set	R^2	MSE
Laptop	Win7	1	0.824	0.606
Laptop	Win7	2	0.906	1.91
Desktop	Win7	1	0.810	0.315
Desktop	Win7	2	0.906	0.220
High End	Suse11	1	0.778	0.024
High End	Suse11	2	0.807	0.0051
Server	Ubu9	1	0.935	1.17
Server	Ubu9	2	0.946	0.450
Old Server	Ubu11	1	0.857	0.067
Old Server	Ubu11	2	0.813	0.0188

Table 1: Estimation of model parameters for different computers, operating systems (OSs), and data sets.

search Center [30]. ADAPT contains capabilities for power generation, power distribution, and loads representative for what can be found in aerospace vehicles. For the purpose of this paper we focus on a small part of ADAPT containing the following components: EY183 (relay), DC482 (DC load), E181 (voltage sensor), IT181 (current sensor), and ESH183 (relay feedback sensor). Scenarios are taken from Tier 2 of the DX 2009 competition data set.[3] These scenarios consist of nominal runs, with no faults in ADAPT, as well as runs involving one or more faults in components or sensors, diagnosed using BNT's implementation[4] of likelihood weighting (LW) [34], variable elimination (VE) [20, 9], junction tree propagation (JTP) [18, 35], loopy belief propagation (LBP) [26], and Pearl's belief propagation (PBP) [29].[5] It has previously been established that BNs perform very well in this domain, with the BN-based ProDiagnose system [32, 31] having the best performance in 3 of 4 of the DX international diagnostic challenges in 2009 and 2010.[6]

Computational Platforms: Five different computers, representing a broad spectrum of computing capability, were used. The *Laptop* computer is a 2.40GHz Intel i5 M520 with 4 cores, 8 GB of RAM, and 3 MB cache memory. The *Desktop* computer is a 3.20GHz AMD Phenom II X6 1090T with 6 cores, 8 GB of RAM, and 3 MB cache memory. The *High End* computer is a 2.00GHz Intel Xeon X7550 with 64 cores, 126 GB of RAM, and 18 MB cache memory. The *Server* computer is a 2.50GHz Intel Core 2 Quad Q8300 with 4 cores, 8GB GB of RAM, and 2 MB cache memory. The *Old Server* computer is a 2.80GHz Intel Xeon with 4 cores, 4 GB of RAM, and 1 MB cache memory. Broadly speaking, the High End computer is most powerful, due to its many cores and large memories, while the Old Server is least powerful due to its comparatively small memories. As reflected in Table 1, the operating systems Windows 7 (Win7) and several Linux variants–Ubuntu (Ubu9 and Ubu11) and Suse (Suse11) were employed in experiments.

Disturbance Generation: In the following, we model for simplicity low-criticality process disturbances to a computer (including user disturbances) as a Poisson process with rate λ. In the experiments, the low-criticality OS processes are CPU-intensive and execute mathematical operations in a tight loop; there is no I/O. This introduces a stochastic delay in the actuation of the plant: even if the control input $u(k)$ increases at the next time step, $u(k+1) > u(k)$, the probability of the number of processes actually increasing is low. The Poisson disturbance term is regarded as unmodeled dynamics and is not explicitly compensated for in the error term of the ARX model (1). It is of interest to model this phenomenon by a stochastic delay in the control input to the plant.

Note that our approach handles situations in which the resource consumption of low-criticality processes varies dramatically. For example, there can be many low-criticality processes (executing in parallel) that are mostly idle. On the other hand, even a few resource-hungry low-criticality processes could be too much to handle. If many "mostly idle" processes are present, our controller automatically sets the "Max number of processes" parameter $u(k)$ higher than if there are a few resource-hungry low-criticality processes. See Figure 6 and Figure 5 for how $u(k)$ is varied by our controller, due to the varying computational load of low-criticality and Bayesian network processes. The controller maps, see Figure 1, from a setpoint $r(k)$ (Computation time) to a control signal $u(k)$ (Max number of processes), and so is able to handle the varying resource use of different processes.

5 EMPIRICAL RESULTS

In this section we present experimental results on ADAPT data for our reactive approach, as enabled by feedback control and summarized in Figure 1.

5.1 COMPUTATION AS A PLANT

Control Input: The input to the plant (i.e., the computer in Figure 1), which ultimately is computed by a control algorithm, is denoted by $u(k)$. This input determines the number of low-criticality processes that are allowed to run at any given sample time. Denote the actual number of low-criticality processes by $v(k)$ where k denotes a sampling index. In the exam-

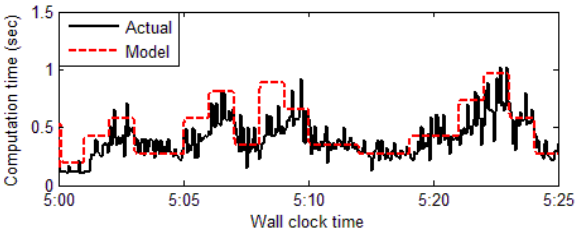
[3] http://www.dx-competition.org/

[4] http://code.google.com/p/bnt/

[5] The specific BNT functions we use are pearl_inf_engine (PBP), belprop_inf_engine (LBP), likelihood_weighting_inf_engine (LW), jtree_inf_engine (JT), and var_elim_inf_engine (VE).

[6] http://www.dx-competition.org/

(a) Junction tree propagation (JTP)



(b) Likelihood weighting (LW)

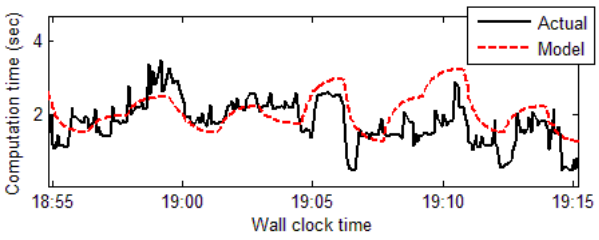


Figure 2: Results of system identification. Two different square waves used as input $u(k)$ to system identification, where we compare Actual completion time $y(k)$ and Model completion time $y_m(k)$ on a Laptop for (a) junction tree propagation versus (b) likelihood weighting.

ples that are used in this paper, if $v(k) \geq u(k)$, then processes are terminated until $v(k) < u(k)$.[7] The termination process is nearly instantaneous with respect to the sampling period.

Plant Modeling: There are several approaches to the modeling of computation for control applications. The approach taken here is termed linear Auto-Regressive modeling with eXogenous input (linear ARX). Nonlinear approaches, such as discrete time neural networks, may also be used. Nonlinear modeling is more complex yet may be able to capture inherent nonlinear behavior otherwise unaccounted for in ARX modeling. On the other hand, linear modeling is generally simpler to understand and implement.[8]

Suppose the plant is described by an ARX model with an additive noise term. Denote $P^{(i)}$, $y^{(i)}(k)$, $u(k)$, and $\eta(k)$, the plant operator, scalar output, input, and noise at sample time k, respectively. The integer i denotes a specific plant or device, such as a particular laptop or desktop. In general, we will assume that we have a finite class of plants $P^{(i)}$ for $i \in [1, M]$.

In general, the relationship between these quantities is given by:

$$y^{(i)}(k) = P^{(i)}u(k) = \phi^T(k-d)\theta^{(i)} + \eta(k) \qquad (1)$$

where $\phi^T(k-d)$ denotes the regression vector and consists of a tapped delay line of input and output measurements, and $\theta^{(i)}$ denotes a vector of real-valued parameters corresponding to the ith plant. A number of methods exist to estimate $\theta^{(i)}$ in both batch and online modes. Similar ARX models have been used in modeling a number of digital processes [12].

Open-loop Modeling and Generalization: Figure 2 shows the result of *open-loop* system identification, specifically the performance of a model where the parameters were estimated using least-squares (batch-mode). This figure compares, for varying sample time k, the *Actual* computation time $y(k)$ (see Figure 1) with our model's predicted computation time $y_m(k)$ (or just *Model*). For both JTP and LW, the rate of Poisson process creation is 45 seconds for LW and the sampling rate is $\frac{1}{5}$ Hz. Figure 2(a) depicts the performance for JTP, while Figure 2(b) shows LW. Overall, the fit between the model and actual behavior is quite good in both cases, perhaps slightly less so for JTP. The following may be the explanation why the JTP graph is more "blocky" and less of a fit than the LW graph: JTP is significantly faster than LW, thus JTP is more sensitive to the other computational processes.

Further details on how open-loop system identification is performed and generalizes are provided in Figure 3.[9] Here, for (1) we simplified $y^{(i)}(k)$ to $y(k)$ and used a first order discrete time model

$$y(k) = c_1 y(k-1) + c_2 u(k-1) + c_3 \qquad (2)$$

to model a specific plant's input output behavior. Input signals used to generate the training and testing data are shown in Figure 3(a), and consist of pseudo-random $u(k)$ square waves. Figure 3(b) shows the result of open-loop system identification for two different computers. Figure 3(b) shows the performance of the model where the parameters were estimated using least-squares (batch-mode), and depicts the performance of the laptop (top) and the server (bottom) using the regression parameters obtained via laptop-generated data. It is clear from these empirical results

[7] Currently, we randomly terminate one of the low-criticality OS processes. Additional information, such as process priority, can easily be used to guide termination.

[8] The ARX model was chosen for several reasons; most importantly it gave solid system identification performance. Allowing the plant model itself to be a probabilistic graphical model would be interesting, and stochastic approaches (based on stochastic differential equations) have been developed in control theory [10, 15]. In light of its performance, we believe that the standard ARX approach is well-justified and leave stochastic control to future work.

[9] While not illustrated in Figure 3, we note that closed loop system identification is also required and presents additional data challenges, for example, data collinearities.

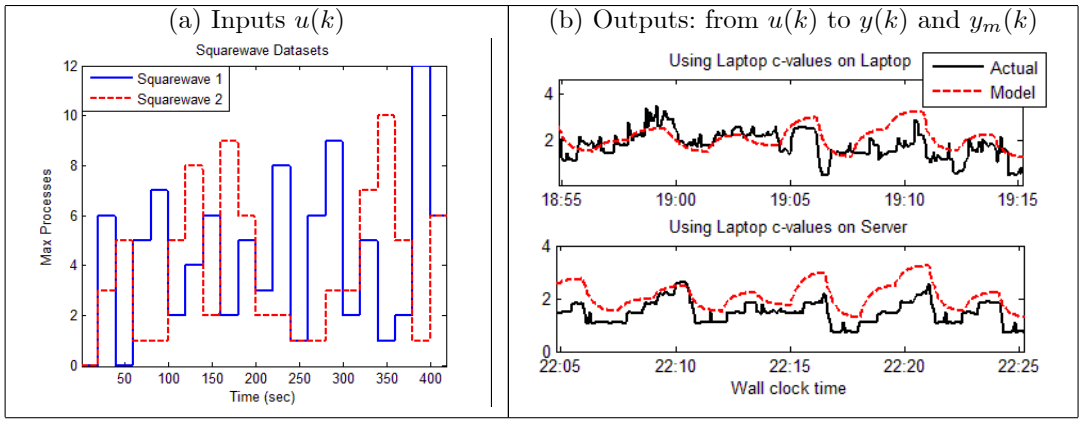


Figure 3: System identification and generalization. (a) Two different squarewaves, Squarewave 1 and Squarewave 2, used as input $u(k)$. Squarewave 1 was used for training of $y_m(k)$ and Squarewave 2 was used for testing. (b) Application to two different computers, comparing Actual completion time $y(k)$ and Model completion time $y_m(k)$: (top) Laptop–good fit and (bottom) Server–decent fit.

that models obtained on one platform (here, laptop) performs best on that platform. The models are likely to perform less well on another platform (here, server), but are still of some value.

5.2 INNER LOOP CONTROL

The objective of the control system is to minimize the error signal given by $e(k) = r(k) - y(k)$ where $r(k)$ denotes the reference or desired computational time required for BN computation. The output, $y(k)$, denotes the actual time the computer takes to complete the BN computation and generate the posterior probabilities BEL($\boldsymbol{Q}, \boldsymbol{e}$) as discussed above.

From the control engineering perspective, there are several key issues: (1) *Static uncertainty in plant parameters*: A tablet PC running Windows Vista will behave differently from a High Performance Computing (HPC) cluster running openSUSE Linux. A control system designed for one computer may not perform well on another. In other words, the control system parameters are uncertain and need to be estimated. (2) *Stochastic disturbances*: The process that determines when a low-criticality process is generated by a user is, in general, stochastic. This is, in itself, an active area of research [1]. Furthermore, the impact a particular low-criticality process has on the high-criticality process may vary substantially. (3) *Stochastic delay in the input process*: When the control input $u(k)$ increases, the actual number of processes may or may not increase. This represents a stochastic delay in the actuation of the digital system being controlled.

To illustrate how our approach handles the above, we optimized a linear controller given by the following Z-transfer function: $H(z) = \frac{K}{1-\alpha z^{-1}}$, where K and α are real-valued controller gain parameters. $H(z)$ is the transfer function from the error signal $e(k)$ to the input to the plant $u(k)$. Performance for a fixed setpoint is shown in Figure 6(a). Computation time is maintained at approximately $r(k) = 2$ sec, however since this is a soft real-time approach there are excursions above this setpoint.

5.3 OUTER LOOP CONTROL

Our proposed research is based on controlling BN computation in an inner loop as well as in an outer loop (see Figure 1); so far we discussed the inner loop. We now briefly discuss the outer loop, where the output BEL($\boldsymbol{Q}, \boldsymbol{e}$) of BN computation, as well as other factors external to the inner loop, may change desired completion time and also plant behavior (sampling frequency, BN computation algorithm, number of particles assuming a simulation algorithm, etc.).

Adaptation to Computational Platform (see Figure 1(c)): Table 1 summarizes results of running likelihood weighting on the ADAPT BN, using five different computers and three different operating systems. System identification was performed on all computers using the two input data sets shown in Figure 3(a). Batch least squares was used to determine the parameters of a first order linear ARX model. Table 1 shows the resulting R^2 and mean squared error (MSE) of the parameter fit for the various computers and operating systems. In most cases, we observed a low MSE.[10]

[10]It is important to note the underlying variations in the ARX model parameters, which are used by the control design process. The diverse set of parameters, omitted here to save space, illustrate the benefit of using techniques from adaptive feedback control, specifically learning from

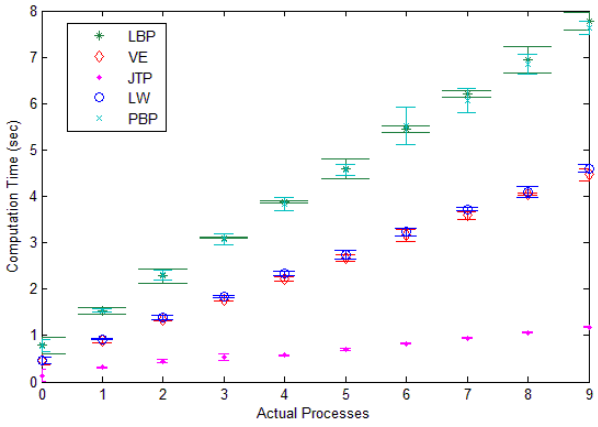


Figure 4: Actual computation time $y(k)$ as a function of actual number of processes $v(k)$ for five different BN inference algorithms LBP, VE, JTP, LW, and PBP running on High End computer.

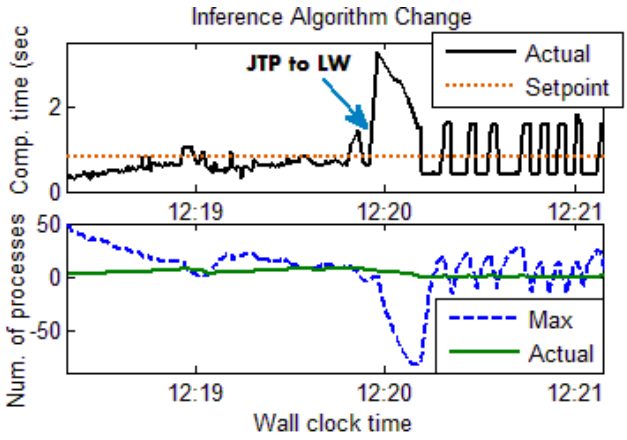


Figure 5: Experiment on Laptop using 1 Hz sampling rate, showing a successful switch of BN inference algorithm from junction tree propagation (JTP) to likelihood weighting (LW) using feedback control.

Optimizing BN Algorithm and Parameters (see Figure 1(b)): In Figure 4, we show the steady state computation time results for the five BN inference algorithms discussed above. Here, steady state is defined loosely as the time k^* such that $y(k+1) = y(k) = y_{ss}$ for all $k \geq k^*$. Plugging this into (2) yields $y_{ss} = \frac{c_2 u_{ss} + c_3}{1 - c_1}$, where $u(k) = u_{ss} \ \forall k \geq k^*$. Thus, the linear gain coefficient (slopes in Figure 4) for each algorithm is given by $\frac{c_2}{1-c_1}$.

The results fall into three groups: fast (JTP), medium (VE and LW), and slow (LBP and PBP) computations. Also, the standard error in computation time varies between the algorithms, with JTP again being the best with a very small standard error.

While JTP is exact and fast, its Achilles' heel is memory consumption [23]. Consequently, it can be necessary, when running other memory-intensive processes, to use a less memory-intensive but inexact algorithm like LW. How should one switch between two algorithms, say the non-anytime algorithm JTP and the anytime algorithm LW, that have very different computational resource requirements but operate on the same BN? Feedback control can help in this regard, see Figure 5. From a control perspective, this is considered *dynamic uncertainty in plant parameters*: Given a computer $P^{(i)}$, a change in the BN algorithm will impact the way the plant, $P^{(i)}$, responds. Here, we switch from JTP to LW around 12:20, while maintaining the setpoint (on average) after a transient period lasting around 10 seconds.[11] This is, to the best of our knowledge, the first demonstration of a successful on-line switch between two very different inference algorithms (from JTP to LW) while a desired completion time is maintained.

Changing the Setpoint (see Figure 1(a)): There is both a supply side and a demand side in computing. On the supply side, one can control the supply of computational resources, in the form of computers, CPUs, CPU threads, or GPU threads. On the demand side, the outer loop can vary the Setpoint $r(k)$, perhaps in combination with varying other outer loop parameters such as sampling frequency $f_C(k)$ and number of particles $p(k)$ used in likelihood weighting [34] or particle filtering [17].

One reason for the outer loop to vary the computational resources allocated to the high-criticality process is illustrated in the following. Suppose, for $k < k^*$, that $P(\boldsymbol{H}(k) \mid \boldsymbol{e}(k))$ suggested that there was one or more faults in the ADAPT electrical power system, while $P(\boldsymbol{H}(k^*) \mid \boldsymbol{e}(k^*))$ indicated that this was a false alarm. In this case, we may want to be less stringent about ensuring that computation of $P(\boldsymbol{H}(k^*+1) \mid \boldsymbol{e}(k^*+1))$, $P(\boldsymbol{H}(k^*+2) \mid \boldsymbol{e}(k^*+2))$, etc. finishes in a timely fashion, in other words it makes sense to put $r(k^*+1) > r(k^*)$. Figure 6(b) shows how this type of step change, at $k^* \approx 23{:}28$, is supported by our control-theoretic framework. The baseline of not varying $r(k)$ is shown in Figure 6(a). We here assume that sample frequency $f_C(k) = f_C$ is constant, and consider (i) $r(k-1) = r(k) < 1/f_C$ (as when $r(k) = 2$ sec in both Figure 6(a) and Figure 6(b)) versus (ii) $r(k) \approx 1/f_C$ (as when $r(k) = 4$ sec Figure 6(b)). The advantage of (i) is that there is a much greater chance that BN computation finishes before a new computational cy-

the computational environment and adapt to changes.

[11] Note, our approach does not make any *hard* real-time guarantees, only *soft* ones, and consequently the actual computation time is sometimes greater than the setpoint.

(a) Fixed Setpoint $r(k)$ (b) Varying Setpoint $r(k)$

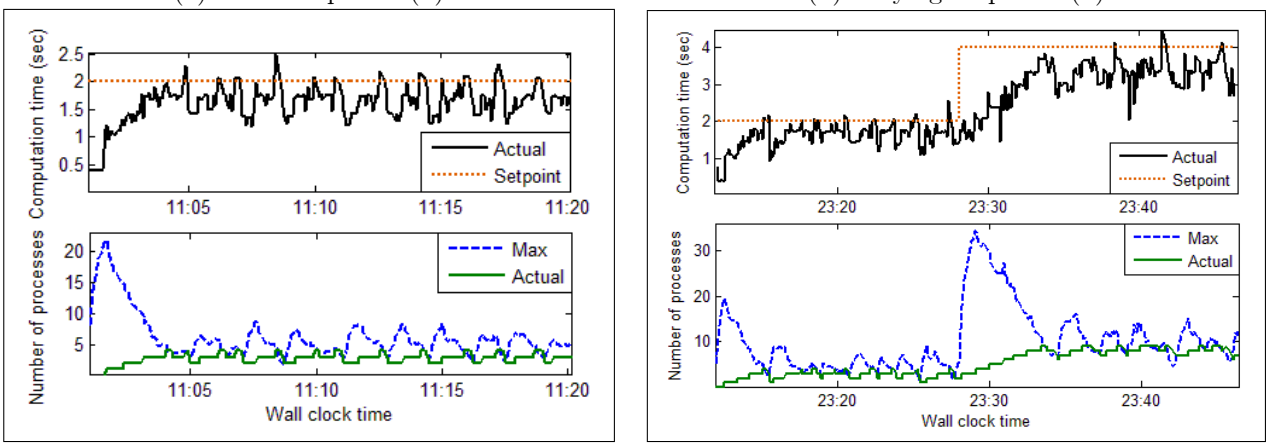


Figure 6: Outerloop optimization for LW, using: (a) fixed Setpoint $r(k) = 2$ and (b) change of Setpoint, approximately at time 23:28, from $r(k) = 2$ to $r(k) = 4$.

cle starts, which is essential when EPS faults are more likely. The advantage of (ii), on the other hand, is that more low-criticality processes are allowed to run.

The trade-off between fast inference for the high-criticality process versus running many low-criticality processes is illustrated in Figure 6. Figure 6(b)'s bottom panel shows an increase in the number of processes, on average, as a result of the increase in $r(k)$ at $k^* \approx$ 23:28; no similar increase can be found in Figure 6(a)'s bottom panel.

6 DISCUSSION AND OUTLOOK

Deploying BN algorithms and other resource-intensive AI algorithms can be a challenge when there are non-trivial constraints on computational resources in applications. In this paper, we have focused on supporting requirements for reactive response without requiring dedicated hard real-time computational resources according to worst-case computational requirements. We have focused on reactive diagnosis using BNs, motivated by domains with some but uncertain domain knowledge (hence probabilistic graphical models, specifically BNs) as well as uncertainty with respect to the computational platform and environment (hence feedback control).

An alternative approach to achieving reactive response is the use of anytime algorithms. The motivation behind anytime inference—namely the goal of intelligent and reactive systems—and our work is quite similar. However, the approaches are very different. Anytime algorithms are inherently inexact and produce solutions whose quality gradually improve with computation time [37]. We focus on what can done, on the computing system level, for a broad range of existing Bayesian network inference algorithms including both exact and inexact algorithms. As a consequence, our approach enables the use of exact but non-anytime algorithms (like variable elimination [20, 9], junction tree propagation [18, 35], and arithmetic circuit evaluation [7, 4]) in reactive settings. The benefit of this is that such exact algorithms often perform very well, however they do have limitations and consequently there are situations where they are unsuitable. With our approach, one can use these exact algorithms and then switch to an inexact (often anytime) algorithm only if needed, rather than having to always use an anytime algorithm.

We are in this paper using rather basic control theory ideas. This enables new results and many opportunities for future work, both theoretical and experimental, and we invite other researchers to participate in the exploration of this exciting area of reseach. We are, for example, developing adaptive control methods that leverage online system identification of the process. There are also many interesting research opportunities related to the use of BN posteriors as well as their accuracy, and perhaps multiple BNs at different levels of detail, in the outer control loop. In this area, there is a strong connection to anytime algorithms and metareasoning that can be further investigated, enabling more reactive and more capable intelligent systems.

Acknowledgements

This material is based upon work supported, in part, by NSF awards CCF0937044 and ECCS0931978.

References

[1] T. Beauvisage. Computer usage in daily life. In *Proc. of the 27th International Conference on Human factors in Computing Systems (CHI-09)*, pages 575–584, Boston, MA, 2009.

[2] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Engineering self-adaptive systems through feedback loops. In B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, editors, *Software Engineering for Self-Adaptive Systems*, pages 48–70. Springer-Verlag, 2009.

[3] A. Carlin and S. Zilberstein. Decentralized monitoring of distributed anytime algorithms. In *Proc. of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pages 157–164, Taipei, Taiwan, 2011.

[4] M. Chavira and A. Darwiche. Compiling Bayesian networks using variable elimination. In *Proc. of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2443–2449, Hyderabad, India, 2007.

[5] A. Choi, A. Darwiche, L. Zheng, and O. J. Mengshoel. A tutorial on Bayesian networks for system health management. In A. Srivastava and J. Han, editors, *Data Mining in Systems Health Management: Detection, Diagnostics, and Prognostics*. Chapman and Hall/CRC Press, 2011.

[6] F. G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[7] A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[8] T. Dean and M. S. Boddy. An analysis of time-dependent planning. In *Proc. of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54, St. Paul, MN, 1988.

[9] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[10] H. Deng, M. Krstic, and R. J. Williams. Stabilization of stochastic nonlinear systems driven by noise of unknown covariance. *IEEE Transactions on Automatic Control*, 46(8):1237 – 53, 2001.

[11] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. In *Proc. of Network Operations and Management Symposium (NOMS-02)*, pages 219–234, 2002.

[12] J. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. Wiley, 2004.

[13] C. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proc. of IEEE INFOCOM*, pages 1726–1734, 2000.

[14] E. J. Horvitz, H. J. Suermondt, and G. F. Cooper. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proc. of the Fifth Conference on Uncertainty in Artificial Intelligence (UAI-89)*, pages 182–193, Windsor, Ontario, 1989.

[15] A. K. Ishihara, J. van Doornik, and S. Ben-Menahem. Stochastic stability of a neural-net robot controller subject to signal-dependent noise in the learning rule. *International Journal of Adaptive Control and Signal Processing*, 24(6):445–466, 2010.

[16] K. Kask and R. Dechter. Stochastic local search for Bayesian networks. In *Proc. of the Seventh International Workshop on Artificial Intelligence and Statistics (AISTATS-99)*, Fort Lauderdale, FL, Jan 1999.

[17] D. Koller and U. Lerner. Sampling in Factored Dynamic Systems. In A. Doucet, J. F. G. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods In Practice*, pages 445–464. Springer-Verlag, 2001.

[18] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B*, 50(2):157–224, 1988.

[19] U. Lerner, R. Parr, D. Koller, and G. Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proc. of the Seventeenth national Conference on Artificial Intelligence (AAAI-00)*, pages 531–537, 2000.

[20] Z. Li and B. D'Ambrosio. Efficient inference in Bayes nets as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81, 1994.

[21] O. J. Mengshoel. Designing resource-bounded reasoners using Bayesian networks: System

health monitoring and diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 330–337, Nashville, TN, 2007.

[22] O. J. Mengshoel. Understanding the role of noise in stochastic local search: Analysis and experiments. *Artificial Intelligence*, 172(8-9):955–990, 2008.

[23] O. J. Mengshoel. Understanding the scalability of bayesian network inference using clique tree growth curves. *Artificial Intelligence*, 174:984–1006, 2010.

[24] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun. Probabilistic model-based diagnosis: An electrical power system case study. *IEEE Trans. on Systems, Man, and Cybernetics*, 40(5):874–885, 2010.

[25] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Portfolios in stochastic local search: Efficiently computing most probable explanations in Bayesian networks. *Journal of Automated Reasoning*, 46(2):103–160, 2011.

[26] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. of the Fifteenth Conference on Uncertainty in AI (UAI-99)*, pages 467–475, 1999.

[27] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. *Real-Time Systems*, 23(1):841–854, 2002.

[28] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research (JAIR)*, 21:101–133, 2004.

[29] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[30] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Advanced diagnostics and prognostics testbed. In *Proc. of the 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 178–185, Nashville, TN, 2007.

[31] B. W. Ricks, C. Harrison, and O. J. Mengshoel. Integrating probabilistic reasoning and statistical quality control techniques for fault diagnosis in hybrid domains. In *Proc. of the Annual Conference of the PHM Society 2011 (PHM-11)*, Montreal, Canada, 2011.

[32] 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.

[33] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[34] R. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence 5*, pages 221–231, Amsterdam, 1990. Elsevier.

[35] P. P. Shenoy. A valuation-based language for expert systems. *International Journal of Approximate Reasoning*, 5(3):383–411, 1989.

[36] C.-Z. Xu, B. Liu, and J. Wei. Model predictive feedback control for QoS assurance in webservers. *Computer*, 41:66–72, March 2008.

[37] S. Zilberstein and S. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82:181–213, 1996.

A State-Transition DBN for Management of Willows in an American Heritage River Catchment

Ann E. Nicholson
Clayton School of IT,
Monash Univ., Australia
ann.nicholson@monash.edu

Yung En Chee
Australian Centre of Excellence
for Risk Analysis,
Univ. of Melbourne, Australia
yechee@unimelb.edu.au

Pedro Quintana-Ascencio
Department of Biology,
Univ. of Central Florida, USA
Pedro.Quintana-Ascencio@ucf.edu

Abstract

Expansion of willows in the naturally mixed landscape of vegetation types in the Upper St. Johns River Basin in Florida, USA, impacts upon biodiversity, aesthetic and recreational values. Managers need an integrated knowledge base to support decisions on where, when and how to control willows. Modelling the spread of willows over space and time requires spatially explicit data on willow occupancy, an understanding of dispersal mechanisms and how the various life-history stages of willows respond to environmental factors and management actions. We describe an architecture for a management tool that integrates environmental spatial data from GIS, dispersal dynamics from a process model and Bayesian Networks (BNs) for modelling the influence of environmental and management actions on the key life-history stages of willows. In this paper we focus on modelling temporal changes in willow stages using a form of Dynamic Bayesian Network (DBN). Starting from a state-transition (ST) model of the willow's lifecyle, from germination to seed-producing adult, we describe the expert elicitation process used to develop a ST-DBN structure, that follows the template described by Nicholson and Flores (2011). We present a scenario-based evaluation of the prototype ST-DBN model.

1 INTRODUCTION

The Upper St. Johns River (USJR) basin in east-central Florida (Figure 1) covers an area of 4890km^2 of which 1620km^2 was originally floodplain marsh dominated by forested wetlands, shrub swamps and herbaceous wetlands. By the 1970s, about two-thirds of the

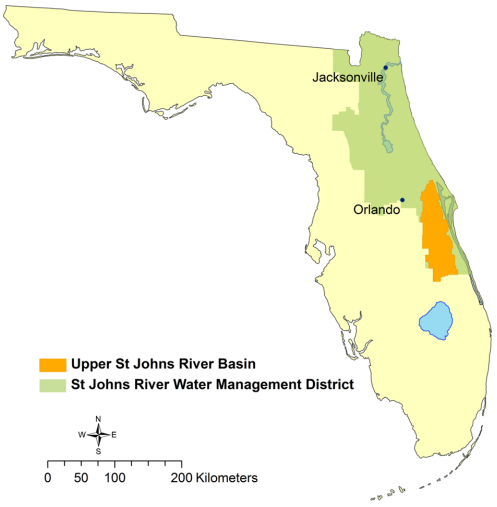


Figure 1: Location of the St. Johns River Water Management District (SJRWMD) and Upper St. Johns River basin in east-central Florida, USA.

historical marshlands had been drained for agriculture and other purposes. The natural hydrological regime was severely altered by the loss of marshlands, and networks of canals, ditches and levees. This led to loss of floodplain storage capacity, increased flood susceptibility and severity, degraded water quality, extensive habitat loss and declines in fish, wading birds, waterfowl and other wildlife. In 1988, the St. Johns River Water Management District (SJRWMD) and the US Army Corps of Engineers began restoration of 607 km^2 of the USJR basin by acquiring land, building storages and plugging drainage canals. The St. Johns River was designated an American Heritage River in 1998.

In the last 50 years, woody shrubs, primarily, Carolina willow (*Salix caroliniana* Michx.), have invaded areas that were historically herbaceous marsh (Kinser et al., 1997). In some management compartments, the area of willows has more than doubled between 1989 and 2001 (Quintana-Ascencio and Fauth, 2010). This change to the historical composition of mixed vegeta-

tion 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.

Managing the spread of willows over space and time requires spatially explicit data on willow occupancy, an understanding of dispersal mechanisms and how the various life-history stages of willows respond to environmental factors and management actions. We describe an architecture for a management tool that integrates environmental spatial data from a Geographical Information System (GIS), dispersal dynamics from a process model and state-transition Dynamic Bayesian Networks (ST-DBNs) (Nicholson and Flores, 2011) for modelling the influence of environmental and management actions on the key life-history stages of willows.

State-transition (ST) models are a convenient means of organising information and synthesising understanding 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.

2 BACKGROUND

2.1 WILLOWS IN UPPER ST. JOHNS RIVER CATCHMENT

S.caroliana is one of four willow species native to the SJRWMD. It occurs over a wide range of saturated soil types along lakeshores and stream banks, and in swamps and marshes. *S.caroliana* produces a very large number of small seeds that disperse by wind and water. Fecundity increases with size, but an average adult can produce 165,000 seeds annually (Quintana-Ascencio et al., unpublished data).

Seeds do not exhibit dormancy and have only a short period of viability. For good germination and establishment to occur, the seedbed must be unshaded and free of competition (i.e. bare) and consistently moist but not inundated (Kinser et al., 1997; Pezeshi et al., 1998; Lee, Ponzio et al., 2005). Such conditions can result from natural and human disturbances such as extended spring drawdown of slough areas, natural and controlled burns, grazing and mechanical clearing. Early seedling establishment and survival is governed by the soil moisture regime and degree of competition from other plants. Soil moisture in turn, depends on water-table elevation and soil characteristics such as texture and organic matter content (Pezeshki et al., 1998). However, even under favourable conditions establishment and survival rates are very low. Experimental data for seedling establishment in mucky (high organic matter) soil resulted in survival rates of 7%, whilst seedlings in mixed and sandy soil had negligible survival rates (Quintana-Ascencio and Fauth, 2010).

Once germinants become a yearling or sapling, survival rates are much higher (in the region of 50-100%) and varies depending on the hydrological regime, with prolonged inundation having an adverse impact on survival rates (Quintana-Ascencio and Fauth, 2010).

Like other willow species, *S.caroliana* is thin-barked and fire-sensitive. However, its response to fire can be complex and is mediated by factors such as burn intensity and conditions during and after burning. For instance, if water levels during a burn are sufficient to protect a portion of the willow stem, resprouting may follow after the burn. On the other hand, intense fires in unflooded marshlands can result in willow mortality (Kinser et al., 1997).

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. They recognise that different areas differ in terms of their "invasibility" as well as biodiversity, aesthetic and recreational value. Furthermore, different management interventions are subject to different spatial, environmental and operational constraints, and induce different effects on willows, depending on willow life-history stage and level of cover at the time of treatment. The application of prescribed fire depends on water levels and the quantity of burnable understorey vegetation; mechanical treatment requires dry/drought conditions and suitable substrate that can support the weight of heavy equipment. Fire can produce a range of subtle and complex responses, whereas mechanical clearing obliterates extant vegetation, returning an area to an unoccupied state, regardless of the willow stage at time of treatment. The architecture of our management tool aims to explicitly accommodate these spatial characteristics and management considerations in modelling the temporal dynamics of willow population structure and cover.

2.2 BAYESIAN NETWORKS FOR ENVIRONMENTAL MODELLING

Bayesian networks (Pearl, 1988) are becoming increasingly popular for environmental and ecological mod-

elling and risk assessment. There have been several recent surveys: Uusitalo (2007); Hart and Pollino (2009); Korb and Nicholson (2010); Aguilera et al. (2011), and guidelines for building BNs for environmental applications (e.g. Varis and Kuikka, 1999; Marcot et al., 2006; Kuhnert et al., 2010). A typical early application involved building a model of the response of a particular species or landscape, to environmental conditions and/or management actions, in a limited area; e.g. modeling the effects of eutrophication (excessive nutrients) in the Neuse River watershed (Borsuk et al., 2004), or predicting future abundance and diversity of native fish in the Goulburn River in south-eastern Australia (Pollino et al., 2007). Such models often had no explicit representation of time, other than that implicit in the causal process; or a single time-scale node was used to "flip" the BN's prediction from one time-scale to another (e.g. in Pollino et al. (2007), from 1-year to 5-years). However, some environmental applications concerned with system behaviour over time and/or space have used DBNs and Object-oriented Bayesian Networks (OOBNs) to support this explicitly.

BNs are increasingly being coupled with Geographic Information Systems (GIS) (e.g., Stassopoulou et al., 1998; Smith et al., 2007; Johnson et al., 2012). In such applications, there is typically one copy of the BN associated with each cell in the GIS. Data layers in the GIS may be used as inputs to the BN, and outputs from one or more BN nodes may be fed back to the GIS. Our tool architecture, presented in Section 3, follows this basic structure.

Dynamic Bayesian Networks (DBNs) are a variant of ordinary BNs (Dean and Kanazawa, 1989; Kjærulff, 1992; 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 and Chalabi, 2007; Dawsey et al., 2007; Shihab, 2008), their uptake has been limited. This is perhaps because they are perceived to be "very tedious" (Uusitalo, 2007), or because DBN algorithms are available only in software resulting from research projects,[1] with DBN functionality less well supported in the more widely used commercial products.[2]

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, grasslands and woodlands, see Bestelmeyer et al., 2003; Sadler et al., 2010; Rumpff et al., 2011). In this paper, we apply the template proposed in Nicholson and Flores (2011), shown in Figure 2, which formalised and extended Bashari et al.'s model, combining BNs with the qualitative STMs. S^T represents the state of the system, has n possible values $s_1 \ldots s_n$, and may directly influence any of the environmental and management factors, which are divided into m main factors, $F_1, \ldots, F_m$ (which directly influence transitions) and other sub-factors, $X_1, \ldots, X_r$ (which influence the main factors).

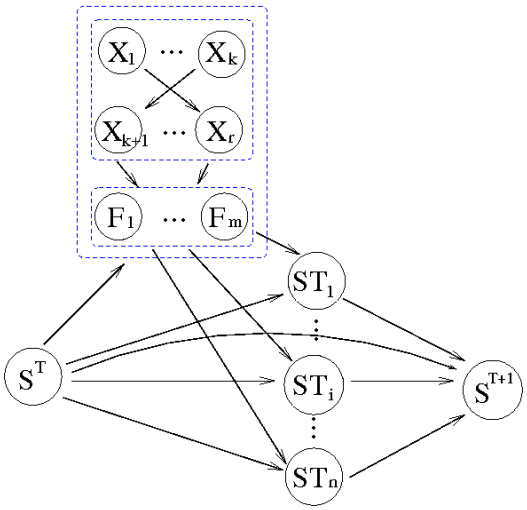


Figure 2: The generic ST-DBN combining STMs with DBNs (Nicholson & Flores, 2011, Fig.10).

The transition nodes, $ST_1, \ldots ST_i, \ldots, ST_n$ represent the transitions from each state s_i, each with at most $n+1$ values (though usually with fewer), one for each "next" state plus "impossible", giving explicit modelling of impossible transitions. As with ordinary DBNs, there is an implied δT, which can be included explicitly as a parent of all the ST nodes, if the time step varies. Each transition node ST has only some of the causal factors as parents. The CPT for the ST node is just a partition of the corresponding CPT if the problem was represented as an ordinary DBN, without the transition nodes. The next state node, S^{T+1}, has to combine the results of all the different transition nodes, given the starting state S, and thus has $n+1$ parents. However, the relationship between the transition nodes and S^{T+1} is deterministic, so the CPT can be generated from a straightforward equation.

Nicholson and Flores (2011) presented a complexity analysis of the ST-DBN, compared to an ordinary DBN (without transition nodes). This showed that any models that explicitly represent all the transitions (i.e. that have ST nodes), only remain tractable when

[1] e.g. BNT, `code.google.com/p/bnt`

[2] For example, the Netica Application (`www.norsys.com`) GUI interface has some DBN functionality, but this is not included in its API.

there are natural constraints in the domain; that is, if the underlying state transition matrix for S is sparse, and if different factors influence different transitions. Such constraints were identified for the willow management problem in the USJR basin.

3 ARCHITECTURE

Figure 3 shows the system architecture for the integrated management tool. It includes a GIS database, a dispersal process model, a ST-DBN model of willow response to environment and management and a management framework. For each cell (modelling unit), the GIS database supplies data on environmental attributes such as soil and vegetation type and information about landscape position and context (e.g. proximity to canal structures or type of surrounding land cover). This data provides inputs to parameterise the dispersal process model, which then makes predictions on seed production that can be mapped and linked to the ST-DBN. The data on spatial context also informs the construction of management strategies (defined here as a set of spatially explicit management actions) and assists in decisions about feasible locations for applying particular management actions. We chose a cell size of 100x100 m (1 ha) to represent a modelling unit. This reflects the resolution of available data for environmental attributes, makes the computational demand associated with dispersal modelling feasible, and is a reasonable scale with respect to candidate management actions.

The ST-DBN synthesises current understanding about how environmental conditions and management actions, acting separately and in various combinations, influence transitions between the key stages of management interest. For each cell, the underlying ST-DBN takes input from the GIS database and management decisions, and predicts willow response for the next timestep. These predictions can then be mapped and aggregated across the target management area to produce evaluation metrics for managers. In this way, managers can "implement", visually compare and quantitatively evaluate different candidate management strategies (or scenarios). The remainder of this paper focuses on the development of the ST-DBN structure.

4 A ST-DBN FOR WILLOWS

The development of the ST-DBN (Figure 4), drew upon a range of sources and used a combination of knowledge derived from ecological and physiological theory, field observations, field and glasshouse experiments and experts (e.g. Kinser et al., 1997; Pezeshi et al., 1998; Lee, Ponzio et al., 2005; Lee, Synder et al., 2005; Ponzio et al., 2006; Quintana-Ascencio & Fauth, 2010). The knowledge engineering process was iterative and incremental, following Boneh (2010), using a series of workshops (2 full-day, 4 half-day) between the knowledge engineers with BN modelling expertise (the first two authors) and the domain expert (the third author), over a six week period. Between each workshop, the models were updated in the BN software, reviewed and revised.

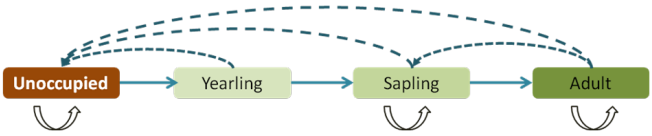


Figure 5: The four willow stages of management interest and the possible transitions of each stage. Arrows indicate the direction of possible transitions.

4.1 NODES

The key points of interest are whether willow is present in a cell or not, and if present, its lifecycle stage and its level of cover.

The stages of management interest modelled in the Stage node are: unoccupied, yearling, sapling (non-reproductive juvenile) and adult.

The possible transitions amongst these four stages are shown in Figure 5. Some stage transitions are not possible (e.g. adults and saplings cannot become yearlings and yearlings cannot remain as yearlings at the next time step). The time step across the ST-DBN was chosen to be one year. An annual time step was considered appropriate given the willow's growth and seed production cycle. Our domain expert did not see any benefit in modelling at a finer temporal scale. In particular, seedlings were only of interest from a management point of view if they survived to the yearling stage.

For these four stages or states, the BN has four corresponding transition nodes (shown in Fig. 4): UnOcc_Transition represents the possible transitions from Stage(T)=Unoccupied, Yearling_Transition represents the possible transitions from Stage(T)=Yearling, etc. Note that each S_Transition node has an additional state, NA (Not Applicable), for when Stage(T) was other than S.

Level_of_Cover refers to the proportion of area within a cell that is occupied by willows of any lifecycle stage. When the willows reach the Adult (seed-producing) stage, Size and Level_of_Cover are factors that influence Seed_Production.

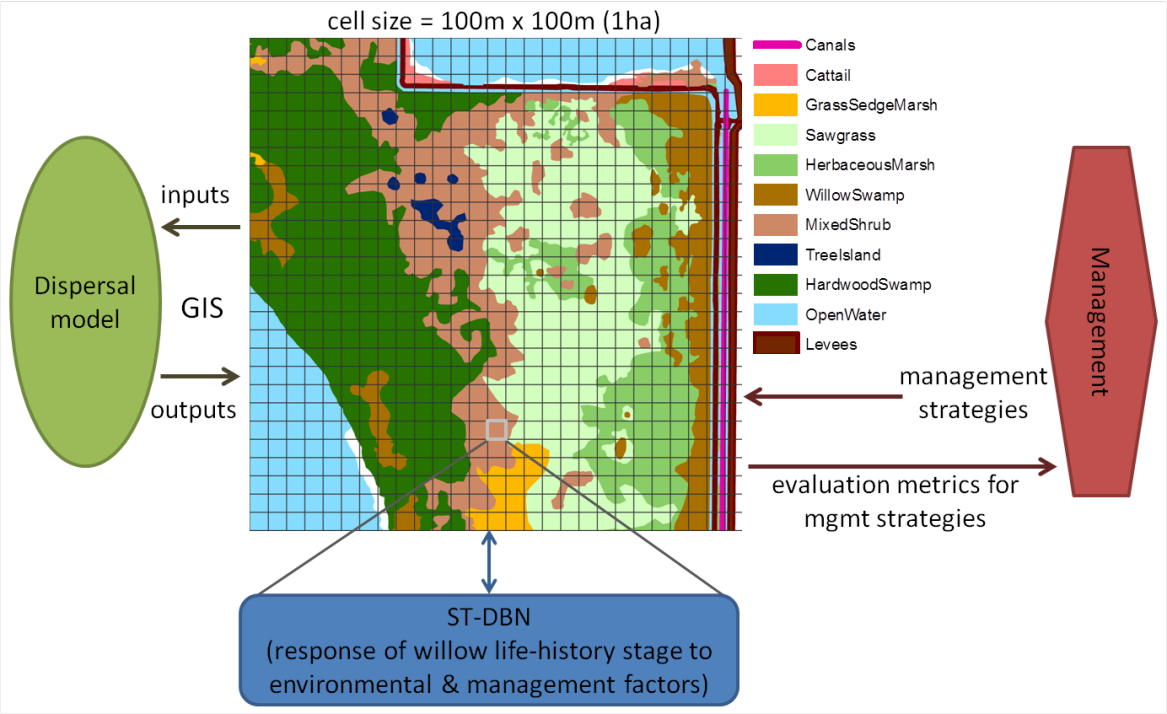


Figure 3: System architecture of the integrated management tool comprising a GIS database, a dispersal process model, a ST-DBN model of willow response to environment and management and a management framework. GIS excerpt shows a portion of the Blue Cypress Marsh Conservatioin Area within the USJR basin.

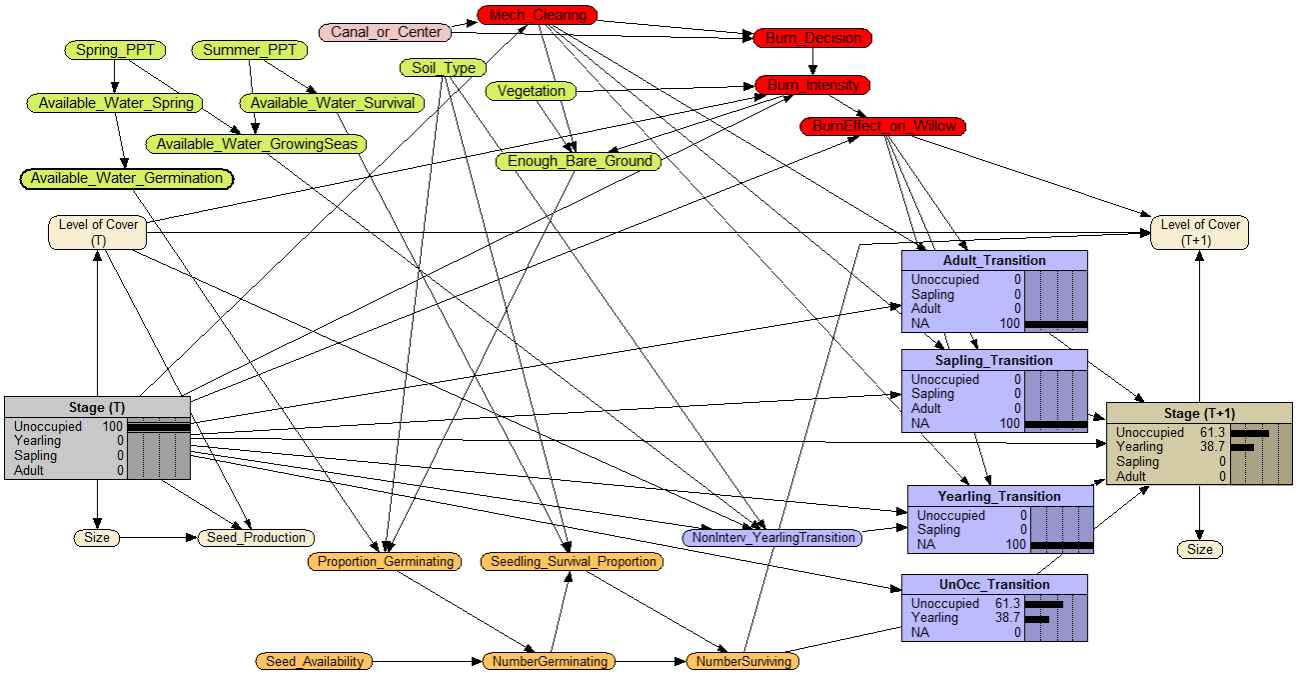


Figure 4: Willows ST-DBN, showing posteriors for Stage and transition nodes for the scenario starting with the cell Unoccupied by willows, with favourable conditions for the transition to Yearling: high seed availability, just right spring (germination) and summer (survival) precipitation, "mucky" (organic, water holding) soil, enough bare ground, no mechanical clearing or prescribed burn.

Stage transitions are governed by environmental and management factors, acting alone or in some combination. Environmental factors include soil type, amount of bare ground, spring and summer precipitation and local vegetation type. Candidate management actions include mechanical clearing (roller-chopping), burning, grazing, herbicide application and hydrological manipulation. Each are subject to different spatial, environmental and operational constraints, and induce different effects on willows, depending on willow life-history stage and level of cover at the time of treatment. For this prototype model, we concentrate on mechanical clearing and burning. Table 1 gives a full listing of the Willow ST-DBN nodes, grouped into (colour-coded) categories. Continuous variables were discretised for implementation in Netica, with discretisation breakpoints determined by a combination of empirical data and expert judgement.

4.2 ARCS

Next, we describe the nature and influences on the possible transitions, represented by the arcs in the Willow ST-DBN (shown in Fig. 4).

Unoccupied areas can become occupied by yearlings if they are successfully colonised within a time step. Successful colonisation depends upon seed availability (which is determined by seed production in and influx from neighbouring cells) and environmentally favourable conditions for seed germination and subsequent seedling survival. Otherwise, unoccupied areas remain unoccupied. Figure 4 shows the Willow ST-DBN starting as Unoccupied, under favourable conditions. Note that the UnOcc_Transition is split between staying Unoccupied (61.3%) and transitioning to Yearling (38.7%), while all the other Transition nodes show are 100% NA (Not Applicable).

Early survival is low, but yearlings can become saplings when environmental conditions are favourable for growth and they are not impacted by mechanical clearing or burning. Otherwise, mortality will cause areas occupied by yearlings to revert to the unoccupied stage.

As saplings grow, they can become reproductive adults, provided they are not impacted by mechanical clearing or burning. Otherwise, they may remain in the non-reproductive sapling stage, if burn impact is minor, or revert to the unoccupied stage if burn impact is major or if mechanical clearing occurs.

In the absence of mechanical clearing or burning, adults stay in the adult stage. Clearing results in almost complete mortality and reversion to an unoccupied stage. The effect of fire depends upon its burn intensity. If sufficiently severe, it can cause mortality and convert areas occupied by adults back to an unoccupied stage, or it might kill off large stems and reduce canopy cover (Lee, Ponzio et al., 2005; Lee, Synder et al., 2005). When adults are damaged in this way, they become non-reproductive for a period as they attempt to recover by resprouting post-fire. For this period, they functionally resemble saplings and we represent this in our ST-DBN by a transition from adult back to the sapling stage.

The initial Level_of_Cover is determined by the number of seedlings that survive when the Stage transitions from unoccupied to yearling. Stages from yearling onwards are robust to environmental variability (e.g. fluctuations in precipitation and inundation), but they are affected by mechanical clearing (which always returns the cell to Unoccupied) or burning (depending on the burn effectiveness).

Again, following the Nicholson and Flores ST-DBN template, the four transition nodes are all parents of the subsequent Stage(T+1) node.

4.3 PARAMETERISATION

We have two stages to our model parameterisation. In the parameterisation for this first prototype, our aim was to represent high-level behaviour, thus the CPTs were constructed using a combination of expert elicitation of process knowledge, expert interpretation of empirical data from field and glasshouse experiments, deterministic and probabilistic functions, statistical models and expert judgement. We do not report details of these here, for reasons of space; they will be reported elsewhere.

The second phase will involve more detailed parameterisation using judgements elicited from a larger pool of domain experts. We will also use specific results from experiments already completed (see Quintana-Ascencio and Fauth, 2010) to calibrate CPTs for some nodes. The field and greenhouse experiments do not provide enough cases to learn the CPTs, nor do they cover an exhaustive range of scenarios. However, they will provide guidance for the parameterisation.

5 SCENARIO-BASED EVALUATION

For this first prototype of the Willow ST-DBN, we conducted scenario-based evaluation with our domain expert throughout the knowledge engineering process. We examined multiple scenarios designed to probe the encoded relationships for key environmentally-driven processes, such as seedling survival and expected responses to management actions, such as the effect of burning. By inputting different combinations of values

Table 1: The nodes of the Willow ST-DBN, grouped into categories with colour-coding (see Figure 4).

Category (node colour)	Nodes
Aspects of willow state (tan)	Stage, Level_of_Cover, Size and Seed_Production
Germination & seedling survival processes (orange)	Seed_Availability, Proportion_Germinating, NumberGerminating Seedling_Survival Proportion and NumberSurviving
Environmental conditions (green)	Soil_Type, Vegetation, Enough_Bare_Ground, spring and summer precipitation (Spring_PPT, Summer_PPT) seasonal water availability for germination, survival and growth (Available_Water_Spring, Available_Water_Germination, Available_Water_Survival, Available_Water_GrowingSeas Canal_or_Centre (i.e. accessibility)
Management options (red)	Mech_Clearing, Burn_Decision (and associated with this option, Burn_Intensity and BurnEffect_on_Willow)
State-transitions (purple)	UnOcc_Transition, NonInterv_YearlingTransition† , Yearling_Transition, Sapling_Transition and Adult_Transition

† Representing expected yearling transition without overlay of management actions.

Table 2: Subset of scenario evaluation results, used to evaluate high-level behaviour of the Willow ST-DBN. For each scenario, columns on the left show the evidence entered; the 4 columns on the right show the distribution for Stage(T+1). For the Yearling, Sapling and Adult scenarios, the Level_of_Cover is High; the probabilities of transitions to UnOccupied are greater for lower levels of cover.

No.	Stage(T) (Seed Avail= High)	Soil	Avail Water Spring.	Avail Water Survival	Enough Bare Ground	Stage(T+1)			
						UnOcc	Yearling	Sapling	Adult
1.	UnOcc	Sandy	JustRight	JustRight	Yes	88.4	11.6	0	0
2.	UnOcc	Mucky	JustRight	JustRight	Yes	61.3	38.7	0	0
3.	UnOcc	Mucky	JustRight	TooMuch	Yes	94.6	5.4	0	0
4.	UnOcc	Mucky	TooLittle	JustRight	Yes	100	0	0	0
	Stage(T)	Soil	Avail Water Growing Season	Mech. Clearing	Burn Decision (Vegetation= Grassland)	Stage(T+1)			
						UnOcc	Yearling	Sapling	Adult
5.	Yearling	Mucky	JustRight	No	No	1	0	99.0	0
6.	Yearling	Sandy	JustRight	No	No	20.0	0	80.0	0
7.	Yearling	Sandy	TooLittle	No	No	40.0	0	60.0	0
8.	Yearling	Sandy	TooMuch	No	No	98.5	0	1.5	0
9.	Yearling	Mucky	JustRight	Yes	No	99.0	0	1.0	0
10.	Yearling	Mucky	TooLittle	No	Yes	81.9	0	18.1	0
	Stage(T)	Vegetation		Mech. Clearing	Burn Decision	Stage(T+1)			
						UnOcc	Yearling	Sapling	Adult
11.	Sapling	[Any]		No	No	10.0	0	67.0	23.0
12.	Sapling	[Any]		Yes	No	99.5	0	0.5	0
13.	Sapling	HerbWet		No	Yes	20.0	0	71.1	8.9
14.	Sapling	Woodland		No	Yes	15.0	0	69.1	15.9
15.	Sapling	Grassland		No	Yes	22.7	0	70.9	6.4
16.	Adult	[Any]		No	No	1.0	0	0	99.0
17.	Adult	[Any]		Yes	No	99.0	0	0	1.0
18.	Adult	HerbWet		No	Yes	0.92	0	1.6	97.5
19.	Adult	Woodland		No	Yes	0.96	0	0.8	98.2
20.	Adult	Grassland		No	Yes	0.8	0	4.0	95.2

for the relevant environment and management variables, and examining the results in key intermediate and final output nodes, we were able to identify errors in CPTs, logical inconsistencies, and nodes that needed splitting, combining or redefining.

Table 2 presents a small subset of these scenarios together with the distributions obtained for Stage(T+1), while Figure 6 shows fragments of the BN with posterior distributions for some of the variables of interest.[3] The evaluation results in Table 2 and Figure 6 are consistent with our understanding of the influence of environment and management actions on key life-history stages of willows, as described in Sections 2.1 and 4. This suggests the basic structure of the prototype ST-DBN (the nodes and their values, together with the arcs) is appropriate.

6 CONCLUSIONS

We have described an architecture for a willow management tool for the Upper St. Johns River basin, Florida, USA, that integrates environmental spatial data from GIS, dispersal dynamics from a process model and BNs for modelling the influence of environmental drivers and management actions on the key life-history stages of willows. The focus of this paper has been on modelling temporal changes in willow stages using a form of DBN. Starting from a state-transition (ST) model of the willow's lifecyle, from germination to seed-producing adult, we described the process used to develop a ST-DBN structure that follows the template described by Nicholson and Flores (2011). The high-level behaviour of this prototype Willow ST-DBN has been demonstrated through scenario-based evaluation.

Our next task is to evaluate the model and revise the parameterisation of the model using judgements from a larger pool of domain experts, together with specific experimental results, where appropriate and available. Once the ST-DBN for an individual cell passes acceptance testing by our domain experts, we will integrate it with the GIS and the seed dispersal model. This will require introducing a relationship between seed production (an output node in the ST-DBN) and seed availability (some combination of the seed production at nearby cells, as informed by the dispersal process model). Finally, the overall system will be evaluated against management options across the whole river basin.

[3]These are screenshots from the BN software, Netica, with layout of nodes compressed due to reasons of space.

Acknowledgements

AEN and YEC acknowledge the support of ARC Linkage LP110100304. This project benefited from contributions by many thoughtful and hard-working individuals. D.Hall, K.Ponzio and K.Snyder (SJRWMD) provided access to sites, knowledge about willow invasion and advice on experiments. S.Green, J.Navarra, H.Smith and E.Stephens assisted with field and greenhouse work. We thank J.Fauth (UCF) and the graduate students of PQ-A's Restoration Ecology classes for their efforts.

References

Aguilera, P., A. Fernndez, R. Fernndez, R. Rum, and A. Salmern (2011). Bayesian networks in environmental modelling. *Environmental Modelling and Software 26*(12), 1376–1388.

Bashari, H., C. Smith, and O. Bosch (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., J. R. Brown, K. M. Havstad, R. Alexander, G. Chavez, and J. E. Herrick (2003). Development and use of state-and-transition models for rangelands. *Journal of Range Management* (2), 114–126.

Boneh, T. (2010). *Ontology and Bayesian Decision Networks for Supporting the Meteorological Forecasting Process*. Ph. D. thesis, Clayton School of Information Technology, Monash University.

Borsuk, M., C. Stow, and K. Reckhow (2004). A Bayesian network of eutrophication models for synthesis, prediction, and uncertainty analysis. *Ecological Modelling 173*(2-3), 219–239.

Dawsey, W. J., B. S. Minsker, and E. Amir (2007). Real time assessment of drinking water systems using a Dynamic Bayesian network. In *World Environmental and Water Resources Congress*.

Dean, T. and K. Kanazawa (1989). A model for reasoning about persistence and causation. *Computational Intelligence 5*, 142–150.

Hart, B. and C. Pollino (2009). Bayesian modelling for risk-based environmental water allocation. Waterline Report Series No 14, National Water Commission, Canberra, Australia.

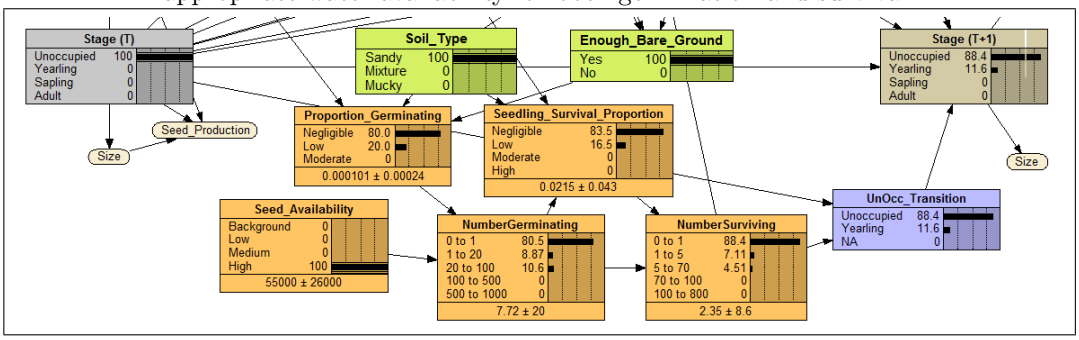
Johnson, S., S. Low-Choy, and K. Mengersen (2012). Integrating Bayesian networks and Geographic Information Systems: goos practice examples. *Integrated Environmental Assessment and Management 8*(3), 473–479.

Kinser, P., M. A. Lee, G. Dambek, M. Williams, K. Ponzio, and C. Adamus (1997). Expansion of willow in the blue cypress marsh conservation area, upper st. johns river basin. Professional Paper SJ97-PP1, St. Johns River Water Management District, Palatka, Florida.

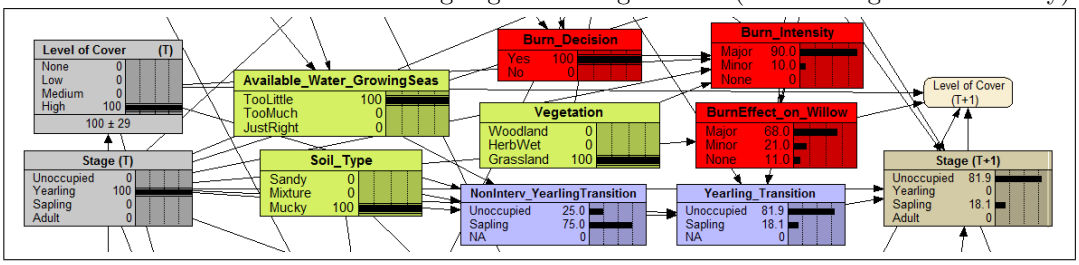
Kjærulff, U. (1992). A computational scheme for reasoning in dynamic probabilistic networks. In *UAI92 – Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, San Mateo, CA, pp. 121–129.

Korb, K. B. and A. E. Nicholson (2010). *Bayesian Artificial Intelligence* (2nd ed.). Chapman & Hall/CRC.

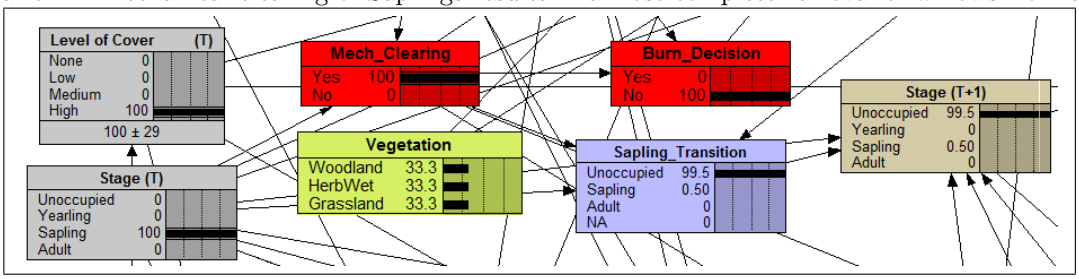
Scenario 1: High seed availability, sandy soil type, sufficient bare ground and appropriate water availability for both germination and survival.



Scenario 10: High cover of Yearlings, mucky soil type, too little water during the growing season, and burn treatment when surrounding vegetation is grassland (which has good burnability)



Scenario 12: Mechanical clearing of Saplings results in almost complete removal of willows from a cell



Scenario 19: Burn treatment for a cell containing high cover of willow Adults when the surrounding vegetation is woodlands, is ineffectual (as Adult willows inhibit burning)

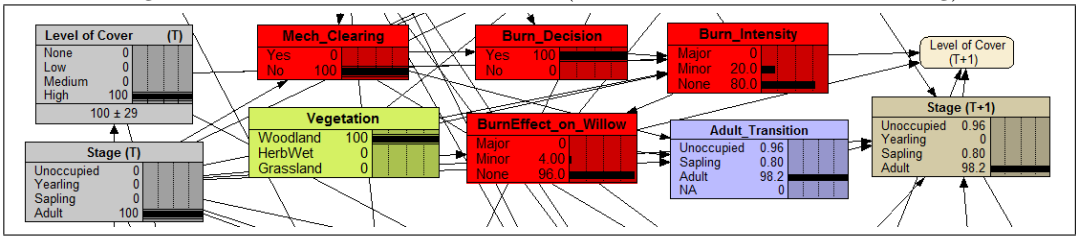


Figure 6: Fragments of the Willow ST-DBN (Netica screenshots) for scenarios 1, 10, 12 and 19 (from Table 2)

Kuhnert, P. M., T. G. Martin, and S. P. Griffiths (2010). A guide to eliciting and using expert knowledge in Bayesian ecological models. *Ecology Letters 13*(7), 900–914.

Lee, M. A. B., K. J. Ponzio, and S. J. Miller (2005). Response of willow (*salix caroliana michx.*) in a floodplain marsh to a growing season prescribed fire. *Natural Areas Journal 25*, 239–245.

Lee, M. A. B., K. L. Synder, P. Valentine-Darby, S. J. Miller, and K. J. Ponzio (2005). Dormant season prescribed fire as a management tool for the control of *salix caroliana* michx. in a floodplain marsh. *Wetlands Ecology and Management 13*, 479–487.

Marcot, B., J. Steventon, G. Sutherland, and R. McCann (2006). Guidelines for developing and updating Bayesian belief networks applied to ecological modeling and conservation. *Canadian Journal of Forest Research 36*(12), 3063–3074.

Nicholson, A. and M. Flores (2011). Combining state and transition models with dynamic Bayesian networks. *Ecological Modelling 222*, 555–566.

Nicholson, A. E. (1992). *Monitoring Discrete Environments using Dynamic Belief Networks*. Ph. D. thesis, Department of Engineering Sciences, Oxford.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann.

Pezeshki, S. R., P. H. Anderson, and F. D. Shields (1998). Effects of soil moisture regimes on growth and survival of black willow (*salix nigra*) posts (cuttings). *Wetlands 18*, 460–470.

Pollino, C., O. Woodberry, A. Nicholson, K. Korb, and B. T. Hart (2007). Parameterisation of a Bayesian network for use in an ecological risk management case study. *Environmental Modelling and Software 22*(8), 1140–1152.

Ponzio, K. J., S. J. Miller, E. C. Underwood, S. P. Rowe, D. J. Voltolina, and T. D. Miller (2006). Response of a willow (*salix caroliana* michx.) community to roller-chopping. *Natural Areas Journal 26*, 53–60.

Quintana-Ascencio, P. and J. E. Fauth (2010). Ecological studies of willow (*salix caroliana*). Final report - year 2., Department of Biology, University of Central Florida, Orlando, Florida.

Rumpff, L., D. Duncan, P. Vesk, D. Keith, and B. Wintle (2011). State-and-transition modelling for adaptive management of native woodlands. *Biological Conservation 144*(4), 1224–1236.

Sadler, R. J., M. Hazelton, M. M. Boer, and P. F. Grierson (2010). Deriving state-and-transition models from an image series of grassland pattern dynamics. *Ecological Modelling 221*(3), 433 – 444.

Shihab, K. (2008). Dynamic modeling of groundwater pollutants with Bayesian networks. *Applied Artificial Intelligence 22*(4), 352–376.

Shihab, K. and N. Chalabi (2007). Dynamic modeling of ground-water quality using Bayesian techniques. *Journal of the American Water Resources Association (JAWRA) 43*(3), 664–674.

Smith, C., A. Howes, B. Price, and C. McAlpine (2007). Using a Bayesian belief network to predict suitable habitat of an endangered mammal–the Julia Creek dunnart (*Sminthopsis douglasi*). *Biological Conservation 139*, 333–347.

Stassopoulou, A., M. Petrou, and J. Kittler (1998). Application of a Bayesian network in a GIS based decision making system. *International Journal of Geographical Information Science 12*(1), 23–45.

Uusitalo, L. (2007). Advantages and challenges of Bayesian networks in environmental modelling. *Ecological Modelling 203*(3-4), 312–318.

Varis, O. and S. Kuikka (1999). Learning Bayesian decision analysis by doing: lessons from environmental and natural resources management. *Ecological Modelling 119*, 177–195.

Conjoint Modeling of Temporal Dependencies in Event Streams

Ankur P. Parikh[*†]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
apparikh@cs.cmu.edu

Asela Gunawardana[*]
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
aselag@microsoft.com

Christopher Meek
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
meek@microsoft.com

Abstract

Many real world applications depend on modeling the temporal dynamics of streams of diverse events, many of which are rare. We introduce a novel model class, Conjoint Piecewise-Constant Conditional Intensity Models, and a learning algorithm that together yield a data-driven approach to parameter sharing with the aim of better modeling such event streams. We empirically demonstrate that our approach yields more accurate models of two real world data sets: search query logs and data center system logs.

1 Introduction

Event streams—temporal sequences of discrete events, are ubiquitous in many domains such as the firing patterns of neurons [2], gene expression data [7], system error logs [13], and web search engine query logs. Learning a model for the temporal dependencies between events can be useful for understanding and exploiting the dynamics in such domains. For example, learning that particular system events on a machine predict failures at a later time may allow a system administrator to prioritize preventive maintenance. Understanding how web search queries for commercially valuable terms are dependent on prior queries, possibly for other topics, can help in targeted advertising. In many cases the data exhibits complex temporal dependencies. For example, what a user will query for at a particular time can depend on queries issued in the last few minutes, the previous day, as well as in the extended past. In a data center, the likelihood of a machine failing may depend on cascades of various prior warnings, errors, and failures.

In many domains, it is valuable to model fine distinctions between event types. For example, in targeted advertising, it is valuable to distinguish whether a user will issue queries related to mental health or to back pain rather than simply predicting that a user will issue a healthcare query. In a data center, it is more useful to learn that particular disk errors predict failed reboots than to know that generic error messages predict generic failures. While useful to model, such fine grained event types are rarer than the coarser grained ones that include them, and are therefore more difficult to model. Many models of temporal dependencies in event streams, such as the *Piecewise-Constant Conditional Intensity Model* (PCIM) [8], learn the dependencies of each event type separately, using independent sub-models for each event type. Thus, they are not able to model rare events well.

In this paper, we address this problem using parameter sharing, by introducing *Conjoint Piecewise-Constant Conditional Intensity Models* (C-PCIMs) and a learning algorithm for C-PCIMs, which yield a novel data-driven approach to modeling fine grained event streams. C-PCIMs generalize PCIMs by allowing parameters to be shared across event types, and our learning algorithm uses the data to determine which parameters should be shared. In particular, we give a conjugate prior that allows parameter learning for the C-PCIM to be performed as efficiently as for the PCIM, and that leads to a closed-form marginal likelihood, allowing efficient structure learning. During structure learning, the C-PCIM learns what event types in what historical contexts can be modeled by shared parameters, thereby allowing more efficient use of data during parameter estimation. In cases where events are structured, with the different event types having known attributes, we show how structure learning can take advantage of these attributes to distinguish between different event types when their depen-

[*] These authors contributed equally to the work.

[†] This research was conducted while APP was a research intern at Microsoft Research, Redmond.

dencies differ, while sharing parameters when they do not. Finally, we give empirical evidence that demonstrates the value of C-PCIMs in two real applications which are not well addressed by existing approaches– modeling the temporal query dynamics of web search users and modeling the temporal dynamics of system events in a data center. In the second application, we demonstrate that the expressive power of C-PCIMs combined with the data driven learning approach allows us to relax the strong assumption of identical machines, yielding further accuracy improvements.

2 Related Work

Event streams can be modeled in either discrete or continuous time. Using discrete time approaches such as Hidden Markov Models (HMMs) [1, 14] and Dynamic Bayesian Networks (DBNs) [5] require event times to be discretized, which requires a choice of sampling rate, and with it, trade-offs involving fidelity of representation, time-span of dependencies, and computational cost. We avoid this choice, and model event streams in continuous time. There have been a number of recent approaches for modeling continuous-time processes. C-PCIMs, like PCIMs [8], model event streams as marked point processes, where events have both an arrival time and a label specifying the type of event, via conditional intensity functions. A number of other closely related approaches [15, 23, 24] use regression techniques such as generalized linear models, Cox regression and Aalen regression to model conditional intensity functions. Continuous Time Noisy-Or [21] and Poisson cascades [22] are also approaches for modeling event streams. These approaches do not address model selection, and require a parametric form for temporal dependencies to be specified. Predictive performance is strongly impacted by this modeling choice, which is domain dependent [21, 22]. There has also been some recent work on nonparametric Bayesian approaches for modeling unlabeled event streams [17]. Continuous Time Bayesian Networks (CTBNs) [12] and Markov Jump Processes [16] are Markov process models of the trajectories of discrete variables over continuous time. In contrast to PCIMs and C-PCIMs, they are Markov process models. A CTBN can be used to model an event stream by modeling each kind of event as a transition of a "toggle" variable [20], and using latent state variables to model their dynamics, to give a continuous time analog of an HMM.

Conjoint PCIMs differ from PCIMs in the way that parameter are shared. Such approaches have been used in other problems such as for building hidden Markov models with large state spaces [9, 10], and for building n-gram language models [11]. Hierarchical Gamma-Exponential processes [18] are a hierarchical nonparametric Bayesian approach to conjoint modeling in Markov processes such as CTBNs. Regularization approaches may also be used for conjoint modeling [25], although we are unaware of applications to continuous time event modeling.

3 The Model

We represent an event sequence as $y = \{(t_i, l_i)\}_{i=1}^n$ with $0 < t_1 < \cdots < t_n$, where $t_i \in [0, \infty)$ is the time of the ith event and l_i is its label, drawn from a finite label set $\mathcal{L}$. The *history at time* t of event sequence y is the sub-sequence $h(t, y) = \{(t_i, l_i) \mid (t_i, l_i) \in y, t_i \leq t\}$. We write h_i for $h(t_{i-1}, y)$ when it is clear from context which y is meant. By convention $t_0 = 0$. We define the *ending time* $t(y)$ of an event sequence y as the time of the last event in y: $t(y) = \max(\{t : (t, l) \in y\})$ so that $t(h_i) = t_{i-1}$.

The data x, which is a particular event sequence, is modeled as a realization of a *regular marked point process* [4, 6] with likelihood

$$p(x|\theta) = \prod_{l \in \mathcal{L}} \prod_{i=1}^{n} \lambda_l(t_i|h_i, \theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)} \quad (1)$$

where $\lambda_l(t|h;\theta)$ is the *conditional intensity function* [4] for label l, and $\Lambda_l(t|h;\theta) = \int_{t(h)}^{t} \lambda_l(\tau|h;\theta)d\tau$. We write $\mathbf{1}_{\mathcal{Z}}(\cdot)$ for the indicator function of a set $\mathcal{Z}$ and $\mathbf{1}_z(\cdot)$ for the indicator of the singleton $\{z\}$. Intuitively, $\lambda_l(t|h;\theta)$ is the expected rate of events with label l at time t given the history h. Note that despite the similarity to the likelihood of a non-homogeneous Poisson process, this likelihood does not in general define a Poisson process as the conditioning on history can cause the independent increments property of Poisson processes to not hold. The conditioning on the entire history also means that such processes are non-Markovian. *Piecewise Constant Conditional Intensity Models* (PCIMs) [8] are a particular class of marked point process where the conditional intensity functions are restricted to be piecewise constant. In this paper, we introduce *Conjoint PCIMs* which are PCIMs that use a conjoint representation for the conditional intensity functions. These models are described below.

3.1 PCIMs

In this section, we review PCIMs [8]. PCIMs are a class of marked point process where the conditional intensity function for each label is a piecewise constant function of time, taking one of a finite number of values. That is $\lambda_l(t|y)$ is piecewise constant in t for all $t > t(y)$, and takes on values $\{\lambda_{ls}\}$ for $s \in \Sigma_l$, where Σ_l is a finite label-dependent *state set*. The value λ_{ls} taken on by $\lambda_l(t|y)$ at t for each y is specified

by a piecewise constant *state function* $\sigma_l(t, y)$, so that $\lambda_l(t|y) = \lambda_{l\sigma_l(t,y)}$.

Note that the state s summarizes all the information about t and y necessary for computing $\lambda_l(t|y)$ in that given s, $\lambda_l(t|y)$ can be computed without further information about t and y. However, unlike in Markov models such as CTBNs [12], the state does not contain all the information about t and y for predicting future states.

As described above, the conditional intensity function $\lambda_l(t|y)$ can be specified by a structure $S_l = (\Sigma_l, \sigma_l(\cdot, \cdot))$ consisting of the state set Σ_l and the state function $\sigma_l(\cdot, \cdot)$ and a parameter parameter vector θ_l composed of a non-negative intensity λ_{ls} for each $s \in \Sigma_l$. In turn, a PCIM is specified by a structure S and a parameter Θ that consist of the per-label structures S_l and per-label parameter vectors θ_l.

Gunawardana et al. show [8] that given the structure S, a product of Gamma distributions is a conjugate prior for Θ, and that under this prior, the marginal likelihood of the data can be given in closed form. Thus, parameter estimation can be done in closed form given a structure, and imposing a structural prior allows a closed form Bayesian score to be computed for a structure.

The structure of a PCIM can be represented by a set of decision trees [8]. In particular, each state function σ_l can be represented by a decision tree whose leaves represent the states $s \in \Sigma_l$, as shown in the example of Figure 1. The decision nodes in each tree contain functions that map a time t and a history y to one of its child nodes. These functions are piecewise constant in time, so that the state function $\sigma_l(t, y)$ represented by a decision tree is also piecewise constant. Structure learning for each label l is performed by starting with the trivial tree, and iteratively refining it greedily based on the closed form Bayesian score mentioned above. A detailed presentation of this learning procedure as generalized to C-PCIMs is given in section 4.1.

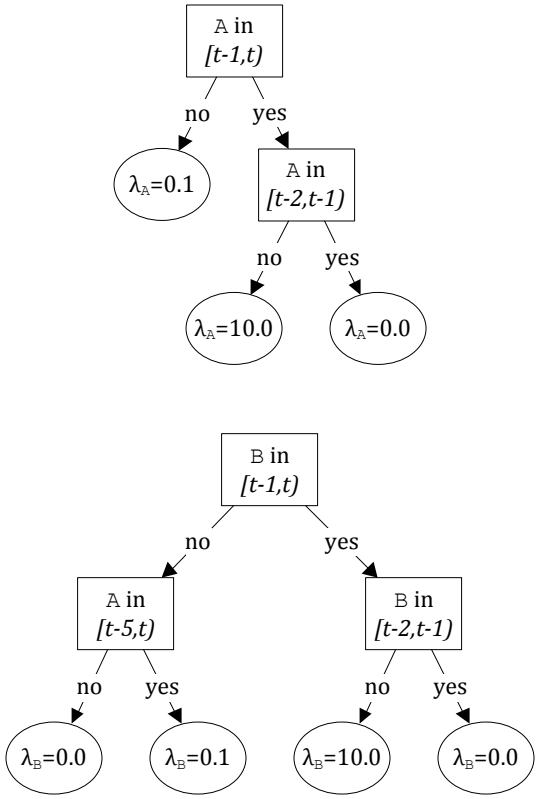


Figure 1: Example Decision trees representing a PCIM for a problem with $\mathcal{L} = \{\mathtt{A}, \mathtt{B}\}$.

3.2 Conjoint PCIMs

Conjoint PCIMs (C-PCIMs), like PCIMs, are marked point processes where the conditional intensity function $\lambda_l(t|y)$ are piecewise constant and take on a finite number of values, but unlike in PCIMs, the conditional intensity functions in C-PCIMs take on values from a single set of values shared across all labels $l \in \mathcal{L}$. Thus $\lambda_l(t|y)$ takes on values $\{\lambda_s\}$ for $s \in \Sigma$ which are shared across all l. Which of these values is taken on by $\lambda_l(t|y)$ is specified by a C-PCIM state function $\sigma(l, t, y)$ which unlike PCIM state functions, is also a function of the label l whose conditional intensity function is being evaluated. Thus, $\lambda_l(t|y) = \lambda_{\sigma(l,t,y)}$. C-PCIMs therefore allow an intensity value λ_s to be shared across conditional intensity functions for different labels, possibly at different times and for different histories. Thus, a C-PCIM is defined by a structure $S = (\Sigma, \sigma(\cdot, \cdot, \cdot))$ consisting of a state set Σ and a state function $\sigma(\cdot, \cdot, \cdot)$ as well as a parameter vector $\Theta = \{\lambda_s\}_{s\in\Sigma}$, all of which are shared across labels $l \in \mathcal{L}$.

We use a decision tree representation of the structure S of a C-PCIM. However, instead of using a different decision tree for each label l as in Gunawardana et al. [8], we use a single tree that is used across all labels, as shown in the example of Figure 2. The leaves of the tree represent states $s \in \Sigma$. However, the decision nodes of a C-PCIM tree contain functions that can depend on l as well as on t and y. In particular, each decision node contains a *basis state function* f chosen from a given set $\mathcal{B}$. Each basis state function $f(l, t, y)$ is piecewise constant in t for each l and y, and takes values from a finite basis state set Σ_f. Thus, a decision node with basis state function f has a child node corresponding to each $s' \in \Sigma_f$. Since the basis state functions are defined to be valid piecewise constant state functions, the mapping from (l, t, y) to

leaves given by the tree is also a valid piecewise constant state function $\sigma(l, t, y)$.

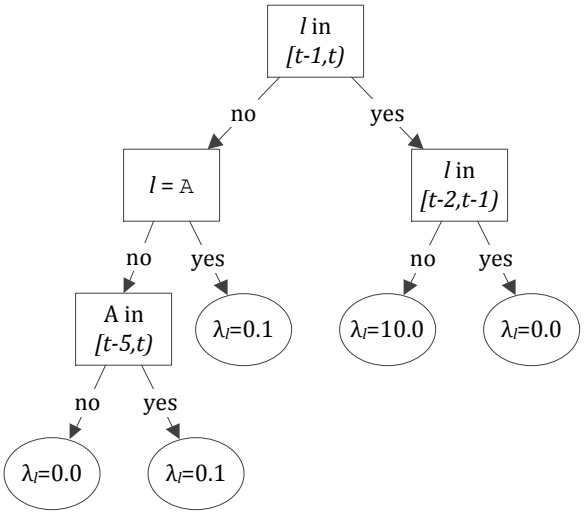


Figure 2: Decision tree representing a C-PCIM equivalent to the example PCIM of Figure 1.

4 Learning Conjoint PCIMs

In this section, we directly generalize the parameter and structure learning approaches for PCIMs [8] to apply to C-PCIMs. For C-PCIMs, the likelihood of equation (1) can be written as

$$p(x|S,\Theta) = \prod_{s\in\Sigma} \lambda_s^{c_s(x)} e^{-\lambda_s d_s(x)} \tag{2}$$

where $d_s(x)$ and $c_s(x)$ are sufficient statistics of the data. $d_s(x)$ is the total duration spent in state s, i.e., that $\sigma(l, t, h(t, x)) = s$ for some l. $c_s(x)$ is the number of times a label l occurs in x when the state function maps to state s for label l. Formally,

$$d_s(x) = \sum_{l\in\mathcal{L}} \int_0^{t(x)} \mathbf{1}_s\left(\sigma\left(l, \tau, h\left(\tau, x\right)\right)\right) d\tau$$
$$c_s(x) = \sum_i \mathbf{1}_l\left(l_i\right) \mathbf{1}_s\left(\sigma(l, t_i, h_i)\right).$$

Note that since $\sigma(l, \tau, h)$ in the integral above is piecewise constant in τ, the integral reduces to a sum over the constant pieces of $\sigma(l, \tau, h)$.

A product of Gamma priors on λ_s is conjugate for Θ. The corresponding prior and the posterior densities for λ_s are given by

$$p(\lambda_s|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_s^{\alpha-1} e^{-\beta\lambda_s}$$
$$p(\lambda_s|\alpha,\beta,x) = p\left(\lambda_s|\alpha + c_s(x), \beta + d_s(x)\right).$$

In our experiments, we obtain the point estimate $\hat{\Theta} = \mathbf{E}[\Theta|S, x]$ from the training data, given by

$$\hat{\lambda}_s = \frac{\alpha + c_s(x)}{\beta + d_s(x)}.$$

4.1 Structure Learning

For structure learning, we can write the marginal likelihood of the data x given the structure S in closed form as

$$p(x|S) = \prod_{s\in\Sigma} \underbrace{\frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\Gamma(\alpha + c_s(x))}{(\beta + d_s(x))^{\alpha+c_s(x)}}}_{\gamma_s(x)}.$$

As with PCIMs, we use a Bayesian decision tree building procedure [3] in order to learn the structure S. We begin with the trivial structure $\Sigma = s_0$, $\sigma(l, t, y) = s_0$ with only the root node s_0, and refine the structure S by iteratively splitting leaves $s \in \Sigma$ based on basis state functions $f \in \mathcal{B}$. In particular Given a current structure $S = (\Sigma, \sigma)$, a new structure $S' = (\Sigma', \sigma')$ is produced by selecting a state $s \in \Sigma_l$ and a basis state function $f \in \mathcal{B}$ and refining s based on f as follows:

$$\Sigma'_l = \left(\bigcup_{s'\in\Sigma_f} \{s \odot s'\}\right) \cup \Sigma_l \backslash s$$
$$\sigma'(l, t, y) = \begin{cases} s \odot f(l, t, y) & \text{if } \sigma(l, t, y) = s \\ \sigma(l, t, y) & \text{otherwise} \end{cases}$$

where $\odot$ is the concatenation operator. Thus, S' can be represented as a tree where leaf s of the sub-tree S has been split according to the result of f.

In order to select the state s and basis state function f to use in producing a refined structure S' from S, we define a factored prior

$$p(S) \propto \kappa^{|\Sigma|}$$

on the structure S. The posterior probability of a structure S given the data x is then proportional to $p(S)p(x|S) = \prod_{s\in\Sigma} \kappa\gamma_s(x)$, which can be computed in closed form. Thus, the gain in $p(S|x)$ due to splitting state s using basis state function f is

$$\begin{aligned} \text{Gain}(S \to S') &= \frac{p(S'|x)}{p(S|x)} \\ &= \frac{\prod_{s'\in\Sigma'} \kappa\gamma_{s'}(x)}{\prod_{s\in\Sigma} \kappa\gamma_s(x)} \\ &= \frac{\prod_{s'\in\Sigma_f} \kappa\gamma_{s\odot s'}(x)}{\kappa\gamma_s(x)}. \end{aligned}$$

We refine the structure greedily, choosing the refinement with the highest gain, until no further gain results.

5 Basis State Functions for C-PCIMs

The modeling power of a family of C-PCIM is determined by the basis $\mathcal{B}$ of state functions selected. The basis needs to capture the aspects of the history that determine the intensities of events, and need to distinguish labels with different intensities. In addition, the basis needs to allow the sharing of parameters between labels to allow for generalization of event behavior between labels. This is particularly important in problems where some labels occur rarely. We will give basis state functions that take advantage of known structure in the label sets in order to do this. We first describe how we capture label space structure through label attributes, and then give an ontology of basis state functions that make use of this structure.

5.1 Structured Labels

When the labels have a known structure, we will take advantage of it in order to define models that can learn dependencies between events with labels that may be rare, or even not occur in the training data. For example, in data center system event logs, events may be labeled with the machine on which the event took place and the type of the event. An event of type `disk-sector-error` may occur on machine `9,045`. While we may have never observed a disk sector error on a different machine `6,732`, we may wish to allow the structure learning procedure to determine whether the behavior of `disk-sector-error` events generalizes across machines. Thus, we would like the basis state functions to be able to query for the message represented by a label independently of the machine.

In general, we assume that the label set $\mathcal{L}$ has a set of attributes $\mathcal{A}$, where each attribute $a \in \mathcal{A}$ can take values in a set $\mathcal{V}_a$. Label l takes on value $v_a(l)$ of attribute a. In the example above, $\mathcal{A} = \{\texttt{machine-id}, \texttt{event-type}\}$, and $\mathcal{V}_{\texttt{machine-id}}$ ranges over all the machines in the data center, and $\mathcal{V}_{\texttt{event-type}}$ ranges over all possible events. If prior information about the labels is available, it may be encoded through label attributes. For example, if we know a priori that machines in the data center are grouped into database servers and web servers, we could introduce an attribute `server-type` that takes on values $\mathcal{V}_{\texttt{server-type}}\{\texttt{database}, \texttt{web}\}$. On the other hand, we can access label identity as an attribute by using the attribute `identity` with $\mathcal{V}_{\texttt{identity}} = \mathcal{L}$, $v_{\texttt{identity}}(l) = l$. In the descriptions below, we will therefore always assume that there are label attributes defined. In cases where no structural information is available we will simply use $\mathcal{A} = \{\texttt{identity}\}$. In particular, the basis state functions for PCIMs [8] do not explicitly use label attributes, but can be described as using this trivial identity attribute.

5.2 Types of Basis State Functions

Allowing large classes of basis state functions that depend arbitrarily on both the history and time as well as the label is difficult computationally. We therefore restrict attention to three specific classes of basis state functions in this paper.

History Basis State Functions: A history basis state function $f(l, t, y)$ depends only on the history y and the time t and not the label l. In this paper, we concentrate on a particular class of history basis state functions $f_{a,v,d_1,d_2}(l, t, y)$ indexed by an attribute $a \in \mathcal{A}$, a value $v \in \mathcal{V}_a$ and time offsets $d_2 > d_1 \geq 0$, and given by

$$f_{a,v,d_1,d_2}(l,t,y) = \begin{cases} 1 & \text{if } \exists (t', l') \in y : \\ & \qquad t' \in [t - d_2, t - d_1) \\ & \qquad \wedge\, v_a(l') = v \\ 0 & \text{otherwise.} \end{cases}$$

That is $f_{a,v,d_1,d_2}(l, t, y)$ tests if the history y contains an event in the time window between d_1 and d_2 before t, whose label has the value v of attribute a.

Example. In modeling web search query logs, the history basis state function $f_{\texttt{query-category},\texttt{Health},1\text{ hr},1\text{ day}}(l, t, y)$ tests whether y contains a query whose `query-category` attribute is `Health` between 1 hour and 1 day before t.

Label Basis State Functions: A label basis state function $f(l, t, y)$ depends only on the label l and not the time t nor the history y. A label basis state function is $f_{a,v}(l, t, y)$ is indexed by an attribute $a \in \mathcal{A}$, a value $v \in \mathcal{V}_a$ and is given by

$$f_{a,v}(l,t,y) = \begin{cases} 1 & \text{if } v_a(l) = v \\ 0 & \text{otherwise.} \end{cases}$$

That is $f_{a,v}(l, t, y)$ simply tests whether the attribute a of label l has value v.

Example. The label basis state function $f_{\texttt{query-category},\texttt{Health}}(l, t, y)$ tests whether l has `query-category` attribute `Health`.

Match Basis State Functions: A match basis function $f_{a,d_1,d_2}(l, t, y)$ is given by

$$f_{a,d_1,d_2}(l,t,y) = \begin{cases} 1 & \text{if } \exists (t', l') \in y : \\ & \qquad t' \in [t - d_2, t - d_1) \\ & \qquad \wedge\, v_a(l') = v_a(l) \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the match basis state function tests whether the history y contains an event in the time window between d_1 and d_2 before t, whose label matches l in attribute a. This kind of basis state function is useful in modeling repetitions of a given attribute in an event stream.

Example. The basis state function $f_{\texttt{query-category},1\text{ hr},1\text{ day}}(l,t,y)$ tests whether y contains a query with the same `query-category` attribute as l between 1 hour and 1 day before t.

We note that history basis state functions are equivalent to the history basis functions of PCIMs [8] when the attribute a is restricted to be the `identity` attribute described above. Thus, a PCIM can be represented as a C-PCIM that uses only the `identity` attribute, and whose state function uses a tree that first splits based on every possible label basis state function, and then uses only history basis state functions. We use the term *Attribute PCIM* (A-PCIM) to refer to the slight generalization of PCIMs that allows the history basis state functions to use arbitrary attributes.

6 Experimental Results

In this section we evaluate the value of C-PCIMs on two real world tasks with large structured label spaces. The first is to model the query behavior of web search users, and the second is to model the behavior of a cluster of machines in a commercial data center.

In order to evaluate the value of C-PCIMs in modeling event streams with large label spaces, we compare C-PCIMs to PCIMs. To explore the gains due to conjoint modeling as opposed to the use of richer label attributes in modeling, we also compare to A-PCIMs. It has been shown that PCIMs have better computational and predictive performance in comparison to Poisson networks [15], which model conditional intensity functions using generalize linear models. We therefore do not compare to Poisson networks and other closely related approaches that use regression techniques to model the conditional intensity functions [23, 24]. CTBNs with "toggle" variables modeling events and latent variables modeling dynamics are a natural baseline for continuous time event modeling. We explored building such models using the CTBN-RLE toolkit [20], but the current version does not scale to these data sets [19].

6.1 Web Search Query Behavior

Queries issued by the users of a major commercial search engine were used to generate a training set consisting of approximately 100,000 queries collected from approximately 6,000 users over a two month period and a test set consisting of approximately 170,000 queries from approximately 6,000 users over the next month. There was no user or time overlap between the training and test sets. The test set was restricted to contain only users whose query history was available for at least two weeks. The queries were automatically mapped to a two level hierarchy of categories such as `Health & Wellness/Mental Health`. Thus, our label set consisted of the 476 categories in the hierarchy, while the 37 coarse level categories were used as a second (i.e. non-`identity`) attribute, which we name `coarse`. All labels occurred in the training set.

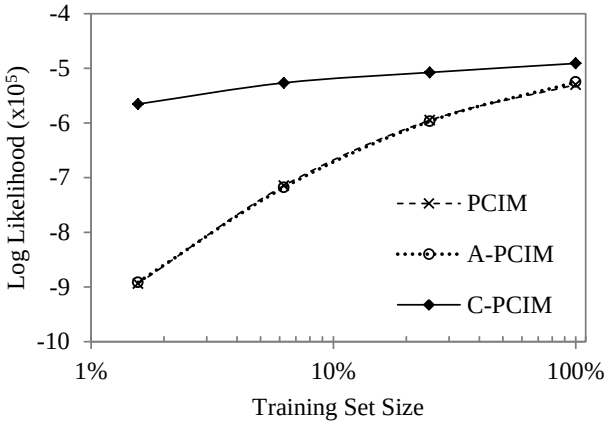


Figure 3: Test set log likelihood of PCIM, A-PCIM, and C-PCIM for the search query data, as a function of training set size.

We use a product of terms of the form of equation (1), one per user, to model the data, choosing not to model inter-user dependencies. We investigated three model classes for modeling users' query behavior. First, we built PCIMs which used only history basis state functions using the `identity` attribute (i.e. the history basis functions only tested for the occurrence of specific labels in the history). Second, we built A-PCIMs that were also allowed to use the `coarse` attributes in the history basis functions. Third, we built C-PCIMs that also used label basis state functions that used the `identity` and `coarse` attributes and match state functions that used the `identity` attribute. Match state functions using the `coarse` attribute were not allowed due to reduce the computation time. All models used a prior with $\alpha = 1/365$ and $\beta = 1$ day, and $\kappa = 0.001$. The history and match basis state functions used the time windows $[t-1\text{ hr}, t)$, $[t-1\text{ day}, t-1\text{ hr})$, $[t-7\text{ days}, t-1\text{ day})$, and $(\infty, t-7\text{ days})$. All models took less than 12 hours to train on a single 3 GHz workstation.

Figure 3 shows the test set log likelihood of all three

models as the amount of training data is varied. The log likelihoods of the PCIM and A-PCIM are nearly indistinguishable, while the C-PCIM performs much better, especially with smaller amounts of training data. This is the expected behavior, since C-PCIMs are better able to share parameters. Note that since C-PCIMs, A-PCIMs, and PCIMs define the same family of marked point processes, we expect them to approach the same predictive performance as the training set grows. While the gap between C-PCIMs and A-PCIMs/PCIMs does shrink as the amount of training data grows, C-PCIMs have higher training set likelihood even when all the training data is used. We examined the likelihoods assigned to each test event by the C-PCIM and the A-PCIM trained with the full training set, in order to determine the statistical significance of this gap. We grouped the per-event likelihoods by label, and found that the C-PCIM significantly outperforms the A-PCIM ($p = 0.01$) on 391 out of the 436 labels observed in the test set, while under-performing the A-PCIM on none, according to a paired sign test.

To understand the practical impact of the likelihood gains, we used importance sampling [8] to forecast whether each test user would issue `Health & Wellness/Mental Health` queries on the eighth day in the test set given their behavior in the first week. Precision-recall curves for the three models are given in Figure 4. Although there are only eight test users who issued these queries on that day, C-PCIMs make much better predictions than A-PCIMs and PCIMs. Such predictions are useful in applications such as targeted display (banner) advertising, where an advertiser may only wish their advertisements to be shown to web users who are interested in a topic related to their advertisement. The assumption is that users who will issue a query in a particular category during the day may be more receptive to advertisements related to that category during that day.

6.2 Data Center Machine Behavior

System logs from a cluster of machines in a commercial data center were used to generate a data set of about 300,000 logged system events from 71 machines, over the period of a month. There were 221 possible messages. This gave 15,691 possible labels, each of which was a machine-message combination. Each machine belonged to one of four machine types that was known a priori. Thus, each label had four attributes $\{\texttt{machine}, \texttt{message}, \texttt{machine-type}, \texttt{identity}\}$. The first two weeks of data was used for training, and the rest for testing.

We use a product of terms of the form of equation (1), one per machine, to model the data, choosing to not allow inter-machine dependencies. We experimented with using the power of C-PCIMs to allow machine specific dependencies. In particular, we compare a PCIM and a C-PCIM that treat machines identically with a PCIM and a C-PCIM that allow machine specific dependencies. Non-identical modeling of machines may be useful if, for example, a certain machine is old is more prone to failures. Models that treat machines identically are allowed to use only the `message` attributes of labels. The PCIM that treats machines identically is forced to use the same conditional intensity function for all labels that agree on their `message` attributes, pooling data from these labels during training. Note that in this setting, PCIMs and A-PCIMs are equivalent in both the identical machine and non-identical machine cases.

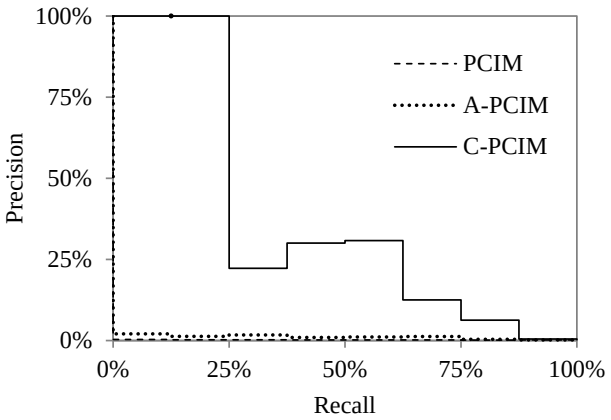


Figure 4: Precision-Recall curves of PCIM, A-PCIM, and C-PCIM for predicting `Health & Wellness/Mental Health` queries.

All models used the prior $\alpha = 0.01$, $\beta = 1.0$ day, and $\kappa = 0.01$. The C-PCIM trees were truncated at a depth of 30 to save computation. The history and match basis state functions used the time windows $[t - 20 \text{ min}, t)$ and $[t - 1 \text{ hr}, t - 20 \text{ min})$. The models took less than 2 hours to train, except the non-identical machine PCIM, which learned a separate tree for each of the 5,307 labels present in the training data. This took less than 12 hours.

Figure 5 shows the log likelihood of events after the first two weeks given the events of the first two weeks for all four models. Note the log likelihoods can be positive since the likelihoods are density values and not probabilities. It can be seen that building separate PCIMs for each label, without pooling data across machines for each message results in a large loss in test set likelihood, due to data sparsity. Both C-PCIMs outperform the PCIMs. Note that the identical machine C-PCIM only has access to message information, and therefore does not have access to any label struc-

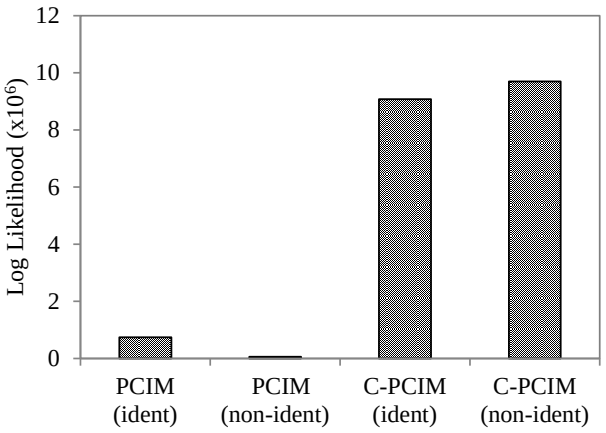


Figure 5: Test set log likelihood of PCIMs and C-PCIMs that treat machines as identical, and PCIMs and C-PCIMs that do not, for the data center event log data.

ture. However, it gives a large gain over PCIMs due to the use of match basis state functions, which model repeated events. The ability of the non-identical machine C-PCIM to leverage label structure to learn dependencies specific to machines or machine types leads to a further gain in likelihood. Unfortunately, events of interest such as machine failures are very rare in this data set, so that there are not enough test cases to obtain statistically significant comparisons between C-PCIMs and PCIMs.

7 Conclusions

We have introduced Conjoint PCIMs, and shown how they improve upon PCIMs in modeling the dynamics of two real world event streams with large structured label sets. We have shown how conjoint modeling across labels allows better models to be built from sparse data, and how C-PCIMs can leverage known structure in the label space. We have also shown that the predictive gains of C-PCIMs are not achieved by Attribute PCIMs that leverage label structure but do not use conjoint modeling.

While it would be of interest to compare the performance of C-PCIMs with other approaches such as CTBNs, limitations in the currently available CTBN implementation prevented us from doing so. It would also be interesting to investigate approaches that extend CTBNs to allow them to take advantage of label structure in the manner of C-PCIMs. Another future direction of interest is to investigate other extensions of PCIMs that allow parameter sharing across labels, such as hierarchical Bayesian approaches or regularization approaches.

References

[1] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37(6):1554–1563, December 1966.

[2] Emery N. Brown, Robert E. Kass, and Parth P. Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuro.*, 7(5), 2004.

[3] David Maxwell Chickering, David Heckerman, and Christopher Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI*, 1997.

[4] D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Elementary Theory and Methods*, volume I. Springer, 2nd edition, 2003.

[5] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *AAAI*, 1988.

[6] Vanessa Didelez. Graphical models for marked point processes based on local independence. *J. Roy. Stat. Soc., Ser. B*, 70(1):245–264, 2008.

[7] N. Friedman, I. Nachman, and D. Peér. Using Bayesian networks to analyze expression data. *J. Comp. Bio.*, 7:601–620, 2000.

[8] Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *NIPS*, 2011.

[9] Mei-Yuh Hwang and Xuedong Huang. Shared-distribution hidden Markov models for speech recognition. *IEEE Trans. Spch. & Aud. Proc.*, 1(4):414–420, October 1993.

[10] Mei-Yuh Hwang, Xuedong Huang, and Fileno A. Alleva. Predicting unseen triphones with senones. *IEEE Trans. Spch. & Aud. Proc.*, 4(6):412–419, November 1996.

[11] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoust., Spch., & Sig.. Proc.*, ASSP-35(3):400–401, March 1987.

[12] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Learning continuous time Bayesian networks. In *UAI*, 2003.

[13] Adam Oliner and Jon Stearley. What supercomputers say - an analysis of five system logs. In *IEEE/IFIP Conf. Dep. Sys. Net.*, 2007.

[14] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.

[15] Shyamsundar Rajaram, Thore Graepel, and Ralf Herbrich. Poisson-networks: A model for structured point processes. In *AIStats*, 2005.

[16] Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *UAI*, 2011.

[17] Vinayak Rao and Yee Whye Teh. Gaussian process modulated renewal processes. In *NIPS*, 2011.

[18] Ardavan Saeedi and Alexandre Bouchard-Côté. Priors over recurrent continuous time processes. In *NIPS*, 2011.

[19] Christian R. Shelton. Personal communication, 2012.

[20] Christian R. Shelton, Yu Fan, William Lam, Joon Lee, and Jing Xu. Continuous time Bayesian network reasoning and learning engine. *JMLR*, 11, 2010.

[21] Aleksandr Simma, Moises Goldszmidt, John MacCormick, Paul Barham, Richard Brock, Rebecca Isaacs, and Reichard Mortier. CT-NOR: Representing and reasoning about events in continuous time. In *UAI*, 2008.

[22] Aleksandr Simma and Michael I. Jordan. Modeling events with cascades of Poisson processes. In *UAI*, 2010.

[23] Wilson Truccolo, Uri T. Eden, Matthew R. Gellows, John P. Donoghue, and Emery N. Brown. A point process framework relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J. Neurophysiol.*, 93:1074–1089, 2005.

[24] Duy Q. Vu, Arthur U. Asuncion, David R. Hunter, and Padhraic Smyth. Continuous-time regression models for longitudinal networks. In *NIPS*, 2011.

[25] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc., Ser. B*, 68(1):49–67, 2006.

Topics Over Nonparametric Time: A Supervised Topic Model Using Bayesian Nonparametric Density Estimation

Daniel D. Walker, Kevin Seppi, and Eric K. Ringger
Computer Science Department
Brigham Young University
Provo, UT, 84604
danw@lkers.org, {kseppi, ringger}@cs.byu.edu

Abstract

We propose a new supervised topic model that uses a nonparametric density estimator to model the distribution of real-valued metadata given a topic. The model is similar to Topics Over Time, but replaces the beta distributions used in that model with a Dirichlet process mixture of normals. The use of a nonparametric density estimator allows for the fitting of a greater class of metadata densities. We compare our model with existing supervised topic models in terms of prediction and show that it is capable of discovering complex metadata distributions in both synthetic and real data.

1 Introduction

Supervised topic models are a class of topic models that, in addition to modeling documents as mixtures of topics, each with a distribution over words, also model metadata associated with each document. Document collections often include such metadata. For example, timestamps are commonly associated with documents that represent the time of the document's creation. In the case of online product reviews, "star" ratings frequently accompany written reviews to quantify the sentiment of the review's author.

There are three basic reasons that make supervised topic models attractive tools for use with document collections that include metadata. *Better Topics*: one assumption that is often true for document collections is that the topics being discussed are correlated with information that is not necessarily directly encoded in the text. Using the metadata in the inference of topics provides an extra source of information, which could lead to an improvement in modeling the topics that are found. *Prediction*: given a trained supervised topic model and a new document with missing metadata, one can predict the value of the metadata variable for that document. Even though timestamps are typically included in modern, natively digital, documents they may be unavailable or wrong for historical documents that have dbeen igitized using OCR. Also, even relatively modern documents can have missing or incorrect timestamps due to user error or system mis-configuration. For example, in the full Enron e-mail corpus[1], there are 793 email messages with a timestamp before 1985, the year Enron was founded. Of these messages 271 have a timestamp before the year 100. *Analysis*: in order to understand a document collection better, it is often helpful to understand how the metadata and topics are related. For example, one might want to analyze the development of a topic over time, or investigate what the presence of a particular topic means in terms of the sentiment being expressed by the author. One may, for example, plot the distribution of the metadata given a topic from a trained model. Several supervised topic models can be found in the literature and will be discussed in more detail in Section 3. These models make assumptions about the way in which the metadata are distributed given the topic or require the user to specify their own assumptions. Usually, this approach involves using a unimodal distribution, and the same distribution family is used to model the metadata across all topics.

[1] http://www.cs.cmu.edu/~enron

These modeling assumptions are problematic. First, it is easy to imagine metadata and topics that have complex, multi-modal relationships. For example, the U.S. has been involved in two large conflicts with Iraq over the last 20 years. A good topic model trained on news text for that period should ideally discover an Iraq topic and successfully capture the bimodal distribution of that topic in time. Existing supervised topic models, however, will either group both modes into a single mode, or split the two modes into two separate topics. Second, it seems incorrect to assume that the metadata will be distributed similarly across all topics. Some topics may remain fairly uniform over a long period of time, others appear quickly and then fade out over long periods of time (e.g., terrorism after 9/11), others enter the discourse gradually over time (e.g., healthcare reform), still others appear and disappear in a relatively short period of time (e.g., many political scandals).

To address these issues, we introduce a new supervised topic model, Topics Over Nonparametric Time (TONPT), based on the Topics Over Time (TOT) model [12]. Where TOT uses a per-topic beta distribution to model topic-conditional metadata distributions, TONPT uses a nonparametric density estimator, a Dirichlet process mixture (DPM) of normals.

The remainder of the paper is organized as follows: in Section 2 we provide a brief discussion of the Dirichlet process and show how a DPM of normals can be used to approximate a wide variety of densities. Section 3 outlines related work. In Section 4 we introduce the TONPT model and describe the collapsed Gibbs sampler we used to efficiently conduct inference in the model on a given dataset. Section 5 describes experiments that were run in order to compare TONPT with two other supervised topic models and a baseline. Finally, in Section 6 we summarize our results and contributions.

2 Estimating Densities with Dirichlet Process Mixtures

Significant work has been done in the document modeling community to make use of Dirichlet process mixtures with the goal of eliminating the need to specify the number of components in a mixture model. For example, it is possible to cluster documents without specifying a-priori the number of clusters by replacing the Dirichlet-multinomial mixing distribution in the Mixture of Multinomials document model with a Chinese Restaurant Process. The CRP is the distribution over partitions created by the clustering effect of the Dirichlet process [1]. So, one way of using the Dirichlet process is in model-based clustering applications where it is desirable to let the number of clusters be determined dynamically by the data, instead of being specified by the user.

The DP is a distribution over probability measures G with two parameters: a base measure G_0 and a total mass parameter m. Random probability measures drawn from a DP are generally not suitable as likelihoods for continuous random variates because they are discrete. This complication can be overcome by convolving the G with a continuous kernel density f [9, 5, 6]:

$$\begin{aligned} G &\sim \boldsymbol{DP}(m, G_0) \\ x_i|G &\sim \int f(x_i|\theta) dG(\theta) \end{aligned}$$

This model is equivalent to an infinite mixture of f distributions with hierarchical formulation:

$$\begin{aligned} G &\sim \boldsymbol{DP}(m, G_0) \\ \theta_i|G &\sim G \\ x_i|\theta_i &\sim f(x_i|\theta) \end{aligned}$$

In our work we use the normal distribution for f. The normal distribution has many advantages that make it a useful choice here. First, the parameters map intuitively to the idea that the θ parameters in the DPM are the "locations" of the point masses of G and so are a natural fit for the mean parameter of the normal distribution. Second, because the normal is conjugate to the mean of a normal with known variance, we can also choose a conjugate G_0 that has intuitive parameters and simple posterior and marginal forms. Third, the normal is almost trivially extensible to multivariate cases. Fourth, the normal can be centered anywhere on the positive or negative side of the origin which is not true, for example, of the gamma and beta distributions. Finally, just as any 1D signal can be approximated with a sum of sine waves, almost any probability distribution can be approximated with a weighted sum of normal densities.

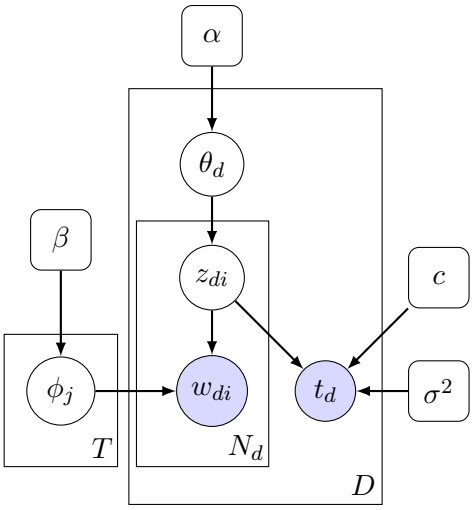


Figure 1: The Supervised LDA model.

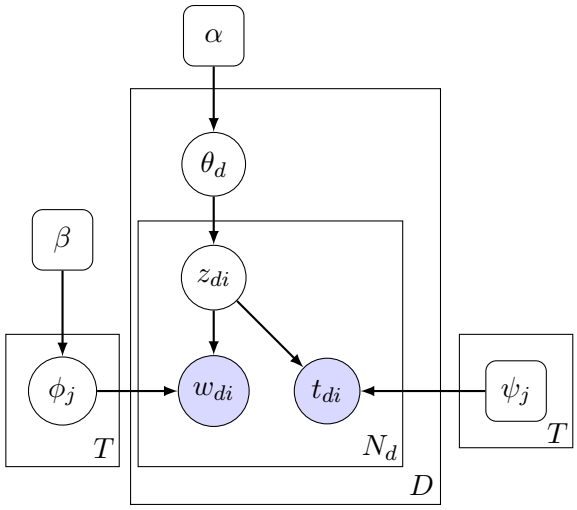


Figure 2: The Topics Over Time model.

3 Related Work

In this section we will describe the three models which are most closely related to our work. In particular, we focus on the issues of prediction and the posterior analysis of metadata distributions in order to highlight the strengths and weaknesses of each model.

The most closely related models to TONPT are Supervised LDA (sLDA) [3] and Topics Over Time [12]. sLDA uses a generalized linear model (GLM) to regress the metadata given the topic proportions of each document. GLMs are flexible in that they allow for the specification of a link and a dispersion function that can change the behavior of the regression model. In practice, however, making such a change to the model requires non-trivial modifications to the inference procedure used to learn the topics and regression co-efficients. In the original sLDA paper, an identity link function and normal dispersion distribution were used. The model, shown in Figure 1, has per-document timestamp variables $t_d \sim \textit{Normal}(c \cdot \overline{z_d}, \sigma^2)$, where c is the vector of linear model coefficients and $\overline{z_d}$ is a topic proportion vector for document d (See Table 1 for a discription of the other variables in the models shown here). This configuration leads to a stochastic EM inference procedure in which one alternately samples from the complete conditional for each topic assignment, given the current values of all the other variables, and then finds the regression coefficients that minimize the sum squared residual of the linear prediction model. Variations of sLDA have been used successfully in several applications including modeling the voting patterns of U.S. legislators [7] and links between documents [4].

Prediction in sLDA is very straightforward, as the latent metadata variable for a document can be marginalized out to produce a vanilla LDA complete conditional distribution for the topic assignments. The procedure for prediction can thus be as simple as first sampling the topic assignments for each word in an unseen document given the assignments in the training set, and then taking the dot product between the estimated topic proportions for the document and the GLM coefficients. In terms of the representation of the distribution of metadata given topics, however, the model is somewhat lacking. The coefficients learned during inference convey only one-dimensional information about the correlation between topics and the metadata. A large positive coefficient for a given topic indicates that documents with a higher proportion of that topic tend to have higher metadata values, and a large negative coefficient means that documents with a higher proportion of that topic tend to have lower metadata values. Coefficients close to zero indicate low correlation between the topic and the metadata.

In TOT, metadata are treated as per-word observations, instead of as a single per-document observation. The model, shown in Figure 2, assumes that each per-word metadatum t_{di} is drawn from a per-topic beta distribution: $t_{di} \sim \textit{Beta}(\psi_{z_{di}1}, \psi_{z_{di}2})$. The inference procedure for TOT is a stochastic EM algorithm, where the topic assignments for each word

are first sampled with a collapsed Gibbs sampler and then the shape parameters for the per-topic beta distributions are point estimated using the Method of Moments based on the mean and variance of the metadata values for the words assigned to each topic.

Prediction in TOT is not as straightforward as for sLDA. Like sLDA, it is possible to integrate out the random variables directly related to the metadata and estimate a topic distribution for a held-out document using vanilla LDA inference. However, because the model does not include a document-level metadata variable, there is no obvious way to predict a single metadata value for held-out documents. We describe a prediction procedure in Section 5, based on work one by Wang and McCallum, that yields acceptable results in practice.

Despite having a more complicated prediction procedure, TOT yields a much richer picture of the trends present in the data. It is possible with TOT, for example, to get an idea of not only whether the metadata are correlated with a topic, but also to see the mean and variance of the per-topic metadata distributions and even to show whether the distribution is skewed or symmetric.

Another related model is the Dirichlet Multinomial Regression (DMR) model [11]. Whereas the sLDA and TOT models both model the metadata generatively, i.e., as random variables conditioned on the topic assignments for a document, the DMR forgoes modeling the metadata explicitly, putting the metadata variables at the "root" of the graphical model and conditioning the document distributions over topics on the metadata values. By forgoing a direct modeling of the metadata, the DMR is able to take advantage of a wide range of metadata types and even to include multiple metadata measurements (or "features") per document. The authors show how, conditioning on the metadata, the DMR is able to outperform other supervised topic models in terms of its ability to fit the observed words of held-out documents, yielding lower perplexity values. The DMR is thus able to accomplish one of the goals of supervised topic modeling very well (the increase in topic quality). However, because it does not propose any distribution over metadata values, it is difficult to conduct the types of analyses or missing metadata predictions possible in TOT and sLDA without resorting to ad-hoc procedures. Because of these deficiencies, we leave the DMR out of the remaining discussion of supervised topic models.

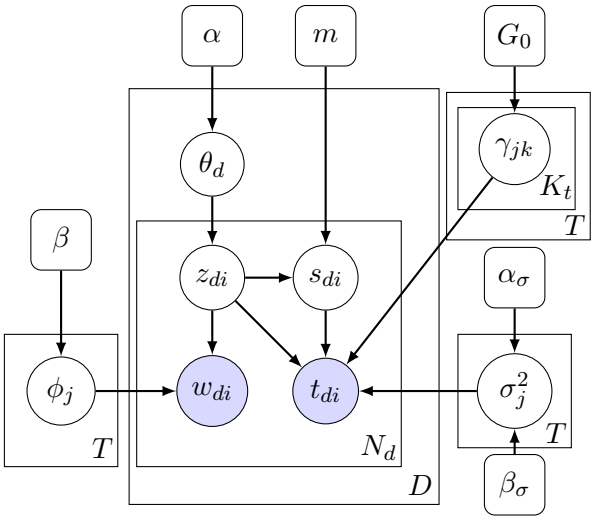


Figure 3: TONPT as used in sampling.

4 Topics Over Nonparametric Time

TONPT models metadata variables associated with each word in the corpus as being drawn from a topic-specific Dirichlet process mixture of normals. In addition, TONPT employs a common base measure G_0 for all of the per-topic DPMs, for which we use a normal with mean μ_0 and variance σ_0^2.

The random variables are distributed as follows:

$$\begin{aligned}
\theta_d|\alpha &\sim \textit{Dirichlet}(\alpha) \\
\phi_t|\beta &\sim \textit{Dirichlet}(\beta) \\
z_{di}|\theta &\sim \textit{Categorical}(\theta_d) \\
w_{di}|z_{di},\phi &\sim \textit{Categorical}(\phi_{z_{di}}) \\
\sigma_j^2|\alpha_\sigma,\beta_\sigma &\sim \textit{InverseGamma}(\alpha_\sigma,\beta_\sigma) \\
G_j|G_0,m &\sim \textit{DP}(G_0,m) \\
t_{di}|G_{z_{di}},\sigma_{z_{di}}^2 &\sim \int f(t_{di};\gamma,\sigma_{z_{di}}^2)dG_{z_{di}}(\gamma)
\end{aligned}$$

where $f(\cdot;\gamma,\sigma^2)$ is the normal p.d.f. with mean γ and variance σ^2. Also, $j \in \{1,\ldots,T\}$, $d \in \{1,\ldots,D\}$, and, given a value for d, $i \in \{1,\ldots,N_d\}$. We note that, as in TOT, the fact that the metadata variable is repeated per-word leads to a deficient generative model, because the metadata are typically observed at a document level and the assumed constraint that all of the metadata values for the words in a document be equivalent is not modeled. The advantage of

Symbol	Meaning
Common Supervised Topic Modeling Variables	
α	Prior parameter for document-topic distributions
θ_d	Parameter for topic mixing distribution for document d
β	Prior parameter for the topic-word distributions
ϕ_j	Parameter for the jth topic-word distribution
z_{di}	Topic label for word i in document d
$\mathbf{z}_{-di}$	All topic assignments except that for z_{di}
$\mathbf{w}$	Vector of all word token types
w_{di}	Type of word token i in document d
t_{di}	Timestamp for word i in document d
t_d	Timestamp for document d
$\mathbf{t}$	Vector of all metadata variable values
$\hat{t}$	A predicted value for the metadata variable
D	The number of documents
T	The number of topics
V	The number of word types
N_d	The number of tokens in document d
TONPT Specific Variables	
m	Total mass parameter for DP mixtures
s_{di}	DP component membership for word i in document d
$\mathbf{s}_{-di}$	All DP component assignments except that for s_{di}
G_0	The base measure of the DP mixtures
μ_0	The mean of the base measure
σ_0^2	The variance of the base measure
γ_{jk}	The mean of the kth mixture component for topic j
$\boldsymbol{\gamma}$	A vector of all the γ values
$\boldsymbol{\gamma}_{-jk}$	$\boldsymbol{\gamma}$ without γ_{jk}
σ_j^2	The variance of the components of the jth DP mixture
$\boldsymbol{\sigma}^2$	A vector of all the DPM σ^2s
$\alpha_\sigma, \beta_\sigma$	Shape and scale parameters for prior on topic σs
K_j	The number of unique observed γs for topic j
n_j	The number of tokens assigned to topic j
n_{jk}	The number of tokens assigned to the kth component of topic j
n_{dj}	The number of tokens in document d assigned to topic j
n_{jv}	The number of times a token of type v was assigned to topic j
$K_{z_{di}}^{<di}$	The number of unique γs observed for topic z_{di} before the ith token of document d
$\tau^{(jk)}$	The set of all t_{di} s.t. $z_{di} = j$ and $s_{di} = k$
$f(y; \mu, \sigma^2)$	The normal p.d.f. at y with mean μ and variance σ^2

Table 1: Mathematical symbols used in the models and derivations of this paper. The common symbols are shared by TONPT, sLDA, and TOT.

this approach is that this configuration simplifies inference, and also naturally balances the plurality of the word variables with the singularity of the metadata variable, allowing the metadata to exert a similarly scaled influence on the topic assignments during inference. In addition, this modeling choice allows for a more fine-grained labeling of documents (e.g., at the word, phrase, or paragraph level) and for finer grained prediction. For example, while timestamps should probably be the same for all words in a document, sentiment does not need to meet this constraint–there are often positive comments even in very negative reviews.

This model does not lend itself well to inference and sampling because of the integral in the distribution over t_{di}. A typical modification that is made to facilitate sampling in mixture models is to use an equivalent hierarchical model. Another modification that is typically made when sampling in mixture models is to separate the "clustering," or mixing, portion of the distribution from the prior over mixture component parameters. The mixing distribution in a DPM is the distribution known as the Chinese Restaurant Process. The Chinese Restaurant Process is used to select an assignment to one of the points that makes up the DP point process for each data observation drawn from G. The locations of these points are independently drawn from G_0.

Figure 3 shows the model that results from decomposing the Dirichlet process into these two component pieces. The K_j unique γ values that have been sampled so far for each topic j are drawn from G_0. The s_{di} variables are indicator variables that take on values in $1, \ldots, K_j$ and represent which of the DPM components each t_{di} was drawn from. This model has the following changes to the variable distributions:

$$s_{di}|z_{di}, \mathbf{s}^{<di}, m \sim \begin{cases} = k \text{ with prob } \propto n_{z_{di},k}^{<di} \\ \quad \text{for } k = 1, \ldots, K_{z_{di}}^{<di} \\ = K_{z_{di}}^{<di} + 1 \text{ with prob } \propto m \end{cases}$$

$$\gamma_{jk}|G_0 \sim G_0$$

$$t_{di}|z_{di}, s_{z_{di}}, \boldsymbol{\gamma}, \boldsymbol{\sigma}^2 \sim f\left(t_{di}; \gamma_{z_{di}s_{di}}, \sigma_{z_{di}}^2\right)$$

Where $\mathbf{s}^{<di}$ refers to all the $s_{d'i'}$ that came "before" s_{di} and before is defined to mean all (d', i') such that $(d' < d)$ or $(d' = d \text{ and } i' < i)$. Likewise, $n_{z_{di},k}^{<di}$ is the count of the number of times that $s_{d'i'} = k$ for all d', i' before s_{di} and $K_{z_{di}}^{<di}$ is the highest value of any

$s_{d'i'}$ (number of unique observed γs) before s_{di}. So, conditioned on z_{di} the s_{di} are distributed according to a Chinese Restaurant Process with mass parameter m.

The θ and ϕ variables in the model are nuisance variables: they are not necessary for the assignment of tokens to topics or for the estimation of the distributions of the response variables so, as is typical when conducting Gibbs sampling on these models, we integrate them out before sampling.

4.1 Gibbs Sampler Conditionals

Now we derive the complete conditionals for the collapsed Gibbs sampler used for inference in the model. There are four groups of variables that must be sampled during inference: the per-word topic labels z, the per word DPM component assignment variables s, the DPM component means $\boldsymbol{\gamma}$, and the per-topic DPM component variances $\boldsymbol{\sigma}^2$. Note that, because of normal-normal conjugacy, it would be possible to collapse the γ variables from the model. We choose to sample values for γ anyway because the parameters of the DPM are useful artifacts in their own right, as they enable rich posterior analyses of the per-topic metadata distributions.

4.1.1 Complete Conditional for z and s

We choose to sample z_{di} and s_{di} in a block, since the calculations necessary to sample z_{di} include those sufficient to sample both variables jointly.

$$\begin{aligned}[z_{di}, s_{di}] =& p(z_{di} = j, s_{di} = k | \mathbf{z}_{-di}, \mathbf{s}_{-di}, \boldsymbol{\sigma}^2, \mathbf{w}, \quad (1)\\ & \mathbf{t}, m, \boldsymbol{\gamma}, \alpha_\sigma, \beta_\sigma, G_0, \alpha, \beta)\\ \propto& \alpha_{\star dj} \frac{\beta_{\star j w_{di}}}{\sum_{v=1}^{V} \beta_{\star jv}} \quad (2)\\ & \cdot \begin{cases} \frac{n_{jk}}{n_j + m} f(t_{di}; \gamma_{jk}, \sigma_j^2) \\ \quad \text{if } k \leq |\gamma_j|, \\ \frac{m}{n_j + m} f(t_{di}; \mu_0, \sigma_0^2 + \sigma_j^2) \\ \quad \text{if } k = |\gamma_j| + 1 \end{cases}\end{aligned}$$

where $\alpha_{\star dj} = \alpha_j + n_{dj}$, $\beta_{\star jv} = \beta_v + n_{jv}$.

4.1.2 Complete Conditional for γ

When sampling a z_{di}, s_{di} pair if $s_{di} = K_{z_{di}} + 1$ (i.e., we are creating a new component for the DPM for that topic), then we need to draw a new γ for the z_{di}th DPM. Also, each γ_{jk} needs to be resampled each iteration of the Gibbs sampler.

Let $\tau^{(jk)} = \{t_{di} : z_{di} = j \text{ and } s_{di} = k\}$ ordered arbitrarily, which groups the t_{di} by the topic and DPM component that they are associated with. The complete conditional for each γ is:

$$\begin{aligned}[\gamma_{jk}] =& p(\gamma_{jk} | \mathbf{s}, \mathbf{t}, \mathbf{w}, \mathbf{z}, \boldsymbol{\gamma}_{-jk}, \boldsymbol{\sigma}^2, \quad (3)\\ & \alpha_\sigma, \beta_\sigma, m, \alpha, \beta, \mu_0, \sigma_0^2)\\ =& f\left(\gamma_{jk}; \mu_{jk\star}, \sigma^2_{jk\star}\right) \quad (4)\end{aligned}$$

where $\sigma_\star^2 = \left(\frac{1}{\sigma_0^2} + \frac{|\tau^{(jk)}|}{\sigma_j^2}\right)^{-1}$ and

$\mu_\star = \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{|\tau^{(jk)}|} \tau_i^{(jk)}}{\sigma_j^2}\right) \cdot \sigma_\star^2$

4.1.3 Complete Conditional for σ^2

The complete conditional is a common result for gamma-normal conjugacy. In this case, the likelihood is restricted to those t_{di} for which $z_{di} = j$:

$$\left[\sigma_j^2\right] = \textit{InverseGamma}\left(\alpha_{\sigma_\star}, \beta_{\sigma_\star}\right) \quad (5)$$

where $\alpha_{\sigma_\star} = \alpha_\sigma + \frac{n_j}{2}$,

$$\beta_{\sigma_\star} = \beta_\sigma + \frac{\sum_{d=1}^{D} \sum_{i=1}^{N_d} [\mathbf{1}_j(z_{di})(\gamma_{z_{di}, s_{di}} - t_{di})^2]}{2},$$

and $\mathbf{1}_j(x)$ is the Kronecker delta.

5 Experiments

We inferred topic assignments and metadata distributions for several real-world datasets using sLDA, TOT, TONPT, and a baseline method that we will refer to as PostHoc in which a vanilla LDA model is inferred over the dataset and then a linear model is fit to the metadata using the document topic proportions as predictors.

Because it is difficult to know a-priori what form the distributions over metadata given topics will take in real-world data, we also ran one experiment with synthetic data, where the metadata distributions were prespecified. Synthetic data was used in order to determine whether TONPT can accurately recover complex metadata distributions in conjunction with topic distributions.

The focus of our experiments was to measure quantitatively how well each model can predict metadata values on unseen data and to assess qualitatively (e.g., via inspection) whether the trained models capture human intuition and domain knowledge with respect to the correlations between topics and metadata values.

5.1 Data

We ran our experiments on three real-world datasets. For each dataset the timestamps of the documents were extracted and used as the metadata. For all real-world datasets, stopwords were removed using the stopwords file included in the MALLET topic modeling toolkit [10]. In addition, words that occurred in more than a half of the documents in a dataset and those that occurred in fewer than 1% were culled. Words were converted to lowercase, and documents that were empty after pre-processing were removed. Finally, only for the TOT model, the metadata were all normalized to the $(0, 1)$ interval to accommodate usage of the beta distribution.

The first dataset consists of the State of the Union Addresses delivered by Presidents of the United States from the first address by George Washington in 1790 to the second address by Barack Obama in 2010. The data was prepared in the manner similar to that of Wang and McCallum [12], in which addresses were subdivided into individual documents by paragraph, resulting in 7507 (three-paragraph) documents. The metadata for this dataset were address timestamps which were normalized to the interval $[0, 1]$.

The second dataset was the LDC-annotated portion of the Enron corpus [2]. This dataset consists of 4,935 emails, that were made public as part of the investigation of illegal activities by the Enron Corporation. It covers approximately one year of time (January 2001 through December 2001). The metadata timestamps for this dataset were extracted from the *Date* header field of the e-mail messages.

The final dataset was the Reuters 21578 corpus[8]. We used the subset of the articles for which topical tags are available, which consists of 11,367 documents. The articles were written during a time interval that spans most of the year 1987. The documents were processed using the same feature selection as for the other two datasets with an additional step in which variants of the names of the months were removed. These words are especially common in this corpus (e.g., in datelines) and provide a strong signal that is not based on the topical content of the articles (i.e., they allowed the models to "cheat"). After feature selection there were 10,230 non-empty documents in the final dataset.

5.2 Procedure

In our prediction experiments, models were trained on 90% of the documents and then were used to predict the metadata values for the remaining 10%. This was repeated in a cross-validation scheme ten times, with the training and evaluation sets being randomly sampled each time. Prediction quality was evaluated using the formula for the coefficient of determination (R^2) used by Blei and McAuliff [3]:

$$R^2(\mathbf{t}, \hat{\mathbf{t}}) = 1 - \frac{\sum_d (t_d - \hat{t}_d)^2}{\sum_d (t_d - \bar{t})^2},$$

where t_d is the actual metadatum for document d, $\hat{t}_d$ is the prediction and $\bar{t}$ is the mean of the observed t_ds. For linear models this metric measures the proportion of the variability in the data that is accounted for by the model. More generally, it is one minus the relative efficiency of the supervised topic model predictor to a predictor that always predicts the mean of the observed data points. This value can be negative in cases where the model being evaluated performs worse than the mean predictor. The means and standard errors of the R^2 values across all ten folds were recorded. In order to assess the statistical significance of the results, a one-sided permutation test was used to calculate p-values for the hypothesis that the mean R^2 for the model with the highest mean R^2 was greater than the mean R^2 for each of the other models being tested. P-values less than 0.05 were considered significant.

As discussed above, prediction in the case of sLDA is quite simple. In the case of TOT and TONPT, prediction is complicated by the fact that these models have per-word metadata variables, and not per-document variables. In addition, they do not produce a prediction using a simple dot product, but instead they provide a distribution over predicted values given a topic assignment. In order to perform prediction in TOT one finds the metadata value with maximal posterior probability given the topic assignments for all of the

words in the test document[12]:

$$\hat{t}_d = \arg\max_t \prod_i p(t|z_{di})$$

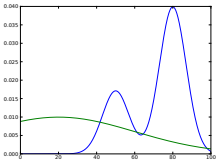
In order to approximate this value, we first infer topic assignments for each word in the document using a version of the model in which the metadata and related variables have been integrated out (i.e., vanilla LDA). Next, because the posterior is a fairly complicated product, and difficult to maximize directly, we approximate by choosing several discrete points and check the value of the posterior at each test point. In the original TOT paper, the candidate points were chosen to represent decades. In an attempt to be more general and to choose candidates that are likely to be of high posterior probability we generate candidates by sampling a metadata value for each word from the beta distribution for the topic assigned to that word. The mean of the sampled points is also added as a candidate. Finally, to generate a prediction the posterior density is calculated at each of the candidates and the one producing the greatest value is chosen.

We found that in the case of TONPT the multimodality of the $p(t|z_{di})$ distribution caused this prediction algorithm to perform poorly. For TONPT, predictions are determined by first estimating the θ_d parameter for the test document using samples obtained from the model with the metadata marginalized out, and then using θ_d to estimate the mean of $p(t|z_{di})$ as the θ_d weighted average of the means of the topic DPMs.

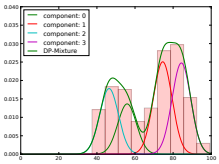
For the TONPT runs, G_0 was chosen to be a normal with mean and variance equal to the sample mean and variance for the observed metadata, α_σ was 2.0, β_σ was 1.0, and m was 1. For all runs, the document-topic parameter $\alpha = 0.1$, and the topic-word parameter $\beta = 0.01$.

5.3 Synthetic Data Results

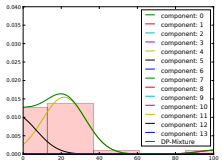
The synthetic dataset was created such that there are 2 topics and a vocabulary of 5 words: "common", "semicommon1", "semicommon2", "rare1" and "rare2". The "common" word occurs with 0.6 probability in both topics, "semicommon1" is slightly more likely than "semicommon2" in the first topic, and slightly less likely in the second topic. The "rare1" word is much more likely in the first topic than the second and "rare2" is much more likely in the second topic than the first, but both are much less common in general than the "semicommon"s.



(a) "True" metadata distributions.



(b) Distribution learned for Topic 0



(c) Distribution learned for Topic 1

Figure 4: The estimated metadata distributions discovered for the synthetic dataset.

Each topic was given a fixed metadata distribution:

$$t_0 \sim 0.3 \cdot f(50, 7) + 0.7 \cdot f(80, 7)$$
$$t_1 \sim f(20, 40)$$

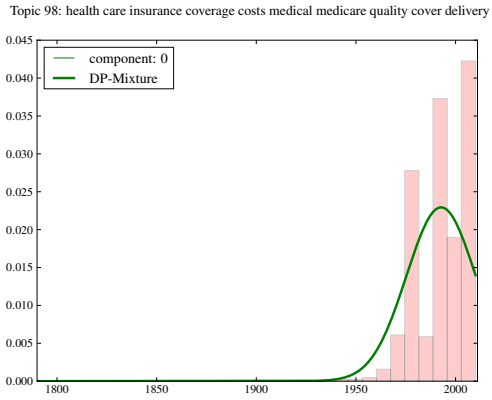
Figure 4 shows how, for one run of the inference procedure, the model was able to separate the two topics and recreate the original metadata distributions with high degree of fidelity. Some runs result in slightly better approximations, while others do worse, but these plots seem to be representative of TONPT's performance on this task.

5.4 Prediction Results

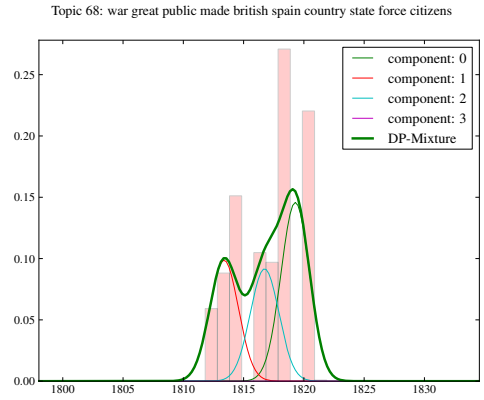
Table 2 shows the performance of the various models for the prediction task with 40 topics, which we found to be a number of topics at which peak performance was observed for most of the models. It can be seen that TONPT is significantly superior on the State of the Union and Reuters data, though TOT does come out ahead on the Enron dataset (but not significantly so).

5.5 Posterior Analysis

Figure 5 shows the distribution over time for two topics found during runs of TONPT on the State of the Union dataset. The topic shown in 5a is typical of



(a) Time distribution for a health care topic



(b) Time distribution for early 1800s conflicts topic

Figure 5: Ratings distributions for two topics found in different runs of TONPT.

Data	Model	Mean R^2	Std Err	p-val
SotU	PostHoc	0.8099	0.0053	0.004
	sLDA	0.8180	0.0046	0.029
	TOT	0.6945	0.0073	0.000
	TONPT	**0.8306**	0.0035	N/A
Enron	PostHoc	0.2434	0.0092	0.002
	sLDA	0.2638	0.0141	0.026
	TOT	**0.3137**	0.0179	N/A
	TONPT	**0.2836**	0.0175	0.137
Reuters	PostHoc	0.1031	0.0072	0.006
	sLDA	0.0775	0.0132	0.010
	TOT	-0.7873	0.0447	0.004
	TONPT	**0.1948**	0.0312	N/A

Table 2: Prediction results for the 3 real-world datasets. Values that are not statistically significantly different from the best are highlighted. P-values are from a 1-sided permutation test against the results from the model with the highest mean R^2.

the majority of the distributions we find (a DPM with only one observed component and thus very close to a simple symmetric distribution). It shows the relatively recent rise in prevalence of the topic of health care in U.S. politics.

The topic shown in 5b is an example of a more complex distribution. This particular example appears to capture several conflicts the United States was involved in during the early 1800s, including The War of 1812 and several conflicts related to the Seminole Wars in Florida (which was a Spanish territory until it was ceded to the U.S. in 1821).

6 Conclusion and Future Work

We have presented TONPT, a supervised topic model that models metadata using a nonparametric density estimator. The model accomplishes the goal of accommodating a wider range of metadata distributions and, in the case of the datasets that we evaluated against, prediction performance remains competitive with previous models. Future work could extend the model to multivariate metadata, such as temporal-spatial data including both timestamps and geolocation information. For example, a multidimensional version of TONPT could be used to capture the development of trends in Twitter data, identifying areas where topics originate and how they spread across the country over time. A multivariate normal component distribution would also capture correlations between metadata elements through a topic covariance matrix.

References

[1] David Aldous, Illdar Ibragimov, Jean Jacod, and David Aldous. Exchangeability and related topics. In *cole d't de Probabilits de Saint-Flour XIII 1983*, volume 1117 of *Lecture Notes in Mathematics*, pages 1–198. Springer Berlin / Heidelberg, 1985. 10.1007/BFb0099421.

[2] Michael W. Berry, Murray Brown, and Ben Signer. 2001 topic annotated Enron email data set, 2007.

[3] David M. Blei and Jon D. McAuliffe. Supervised topic models. In *arXiv:1003.0783*, March 2010.

[4] J. Chang and D. Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

[5] M.D. Escobar. Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, pages 268–277, 1994.

[6] M.D. Escobar and M. West. Computing nonparametric hierarchical models. *Practical nonparametric and semiparametric Bayesian statistics*, pages 1–22, 1998.

[7] S.M. Gerrish and D.M. Blei. Predicting legislative roll calls from text. In *Proceedings of the 28th Annual International Conference on Machine Learning*, number 11, 2011.

[8] D. Lewis. Reuters-21578 text categorization test collection. `http://www.research.att.com/~lewis`, 1997.

[9] A.Y. Lo. On a class of bayesian nonparametric estimates: I. density estimates. *The Annals of Statistics*, 12(1):351–357, 1984.

[10] Andrew Kachites McCallum. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`, 2002.

[11] D. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, pages 411–418, 2008.

[12] Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference(KDD'06)*, Philadelphia, PA, August 2006.

On Approximate Inference of Dynamic Latent Classification Models for Oil Drilling Monitoring

Shengtong Zhong
Department of Computer and Information Science
Norwegian University of Science and Technology
7491 Trondheim, Norway
shket@idi.ntnu.no

Abstract

We have been working with dynamic data from an oil production facility in the North sea, where unstable situations should be identified as soon as possible. Monitoring in such a complex domain is a challenging task. Not only is such a domain typically volatile and following non-linear dynamics, but sensor input to the monitoring system can also often be high dimensional, making it difficult to model and classify the domain's states. Dynamic latent classification models are dynamic Bayesian networks capable of effective and efficient modeling and classification. An approximate inference algorithm utilizing Gaussian collapse has been tailormade for this family of models, but the approximation's properties have not been fully explored. In this paper we compare alternatives approximate inference methods for the dynamic latent classification model, in particular focusing on traditional sampling techniques. We show that the approximate scheme based on Gaussian collapse is computationally more efficient than sampling, while offering comparable accuracy results.

1 Introduction

In the oil drilling, monitor the complex process and identify the current system state is actually very difficult. Monitoring the complex process often involves keeping an eye on hundreds or thousands of sensors to determine whether or not the process is stable. We report results on an oil production facility in the North sea, where unstable situations should be identified as soon as possible [12]. The oil drilling data that we are considering, consisting of some sixty variables, is captured every five seconds. The data is monitored in real time by experienced engineers, who have a number of tasks to perform ranging from understanding the situation on the platform (*activity recognition*) via avoiding a number of either dangerous or costly situations (*event detection*), to optimization of the drilling operation. The variables that are collected cover both topside measurements (like flow rates) and down-hole measurements (like, for instance, gamma rate). For the discussions to be concrete, we will tie the development to the task of *activity recognition* in this paper. The drilling of a well is a complex process, which consists of activities that are performed iteratively as the length of the well increases, and knowing which activity is performed at any time is important for the further *event detection*.

Motivated by this problem setting, a generative model called dynamic latent classification models (dLCM) [12] for dynamic classification in continuous domains is proposed to help the drilling engineers by automatically analyzing the data stream and classify the situation accordingly. Dynamic latent classification models are Bayesian networks which could model the complex system process and identify its system state.

A dynamic latent classification model can be seen as combining a naïve Bayes model with a mixture of factor analyzers at each time point. The latent variables of the factor analyzers are used to capture the state-specific dynamics of the process as well as modeling dependencies between attributes. As exact inference for the model is intractable, an approximate inference scheme based on Gaussian collapse is proposed in our previous study [12]. Although the previous experiments demonstrated that the proposed approximate inference is functioned well the learning of dynamic latent classification models as well as the classification work, we further investigate the approximation's properties by introducing alternative sampling techniques.

The remaining of the paper is organized as follows. In Section 2, we introduce the detail of dLCM. The importance of Gaussian collapse in the inference of dLCM

is discussed in Section 3. Next, alternative sampling techniques are proposed for dLCM in Section 4. After the experiment results are illustrated and discussed in Section 5, the conclusion of the paper is presented in Section 6.

2 Dynamic Latent Classification Models

Dynamic Latent classification models [12] are dynamic Bayesian networks, which can model the complex system process and identify its system state. The complex system process is highly dynamical and complex, which makes it difficult to model and idetentify with the static models and standard dynamic model. The framework of dLCM is specified incrementally by examining its expressivity relative to the oil drilling data.

The dLCM is established from naïve Bayes model (NB), which is one of the simplest static models. In the first step, temporal dynamics of the class variables (a first order Markov chain) is added as considerable correlation between the class variable of consecutive time slices are evidenced from the oil drilling data [12]. This results in a dynamic version of naïve Bayes, which is also equivalent to a standard first order hidden Markov model (HMM) shown in Figure 1, where C^t denotes the class variable at time t and Y_i^t denotes the i-th attribute at time t. This model type has a long history of usage in monitoring, see e.g. [10].

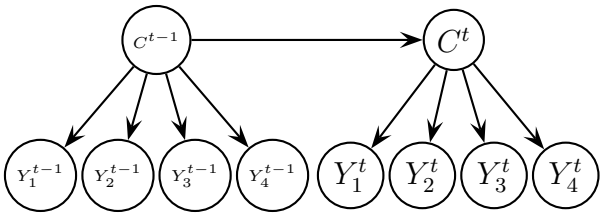


Figure 1: An example of dynamic version of naïve Bayes with 4 attributes using 2-time slice dynamic Bayesian networks representation (2TDBN). Attributes are assumed to be conditionally independent given the class variable.

This model is described by a prior distribution over the class variable $P(c^0)$, a conditional observation distribution $P(\boldsymbol{y}^t|c^t)$, and transition probabilities for the class variable $P(c^t|c^{t-1})$; we assume that the model is *stationary*, i.e., $P(\boldsymbol{y}^t|c^t) = P(\boldsymbol{y}^{t-1}|c^{t-1})$ and $P(c^t|c^{t-1}) = P(c^{t+1}|c^t)$, for all t. For the continuous observation vector, the conditional distribution may be specified by a class-conditional multivariate Gaussian distribution with mean μ_{c^t} and covariance matrix $\boldsymbol{\Sigma}_{c^t}$, i.e., $\boldsymbol{Y}|\{C^t = c^t\} \sim N(\boldsymbol{\mu}_{c^t}, \boldsymbol{\Sigma}_{c^t})$

In a standard HMM, it assumes that the class variable and attributes at different time points are independent given the class variables at the current time, which is violated in many real world setting. In our oil drilling data, there is also a strong correlation between attributes given the class [12]. Modeling the dependence between attributes is then the next step in creating the dLM.

Following [7], we introduce latent variables to encode conditional dependence among the attributes. Specifically, for each time step t we have the vector $\boldsymbol{Z}^t = (Z_1^t, \ldots, Z_k^t)$ of latent variables that appear as children of the class variable and parents of all the attributes (see Figure 2). It can be seen as combining the NB model with a factor analysis model at each time step.

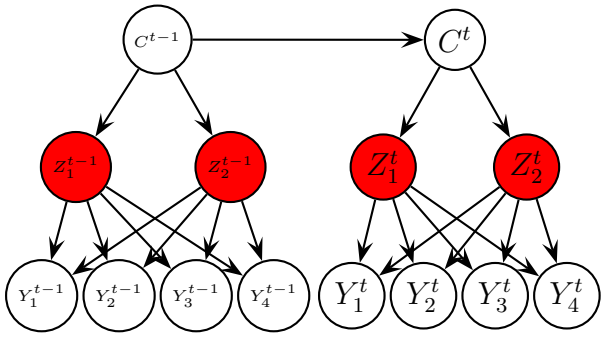


Figure 2: An example model by adding latent variables to dynamic version of naïve Bayes using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4. In each time step, the conditional dependencies between the attributes are encoded by the latent variables $(Z_1^t, \ldots, Z_k^t)$.

The latent variable $\boldsymbol{Z}^t$ is assigned a multivariate Gaussian distribution conditional on the class variable and the attributes $\boldsymbol{Y}^t$ are assumed to be linear multivariate Gaussian distributions conditional on the latent variables:

$$\boldsymbol{Z}^t|\{C^t = c^t\} \sim N(\boldsymbol{\mu}_{c^t}, \boldsymbol{\Sigma}_{c^t}),$$
$$\boldsymbol{Y}^t|\{\boldsymbol{Z}^t = \boldsymbol{z}^t\} \sim N(\boldsymbol{L}\boldsymbol{z}^t + \boldsymbol{\Phi}, \boldsymbol{\Theta}),$$

where $\boldsymbol{\Sigma}_{c^t}$ and $\boldsymbol{\Theta}$ are diagonal covariance matrix and $\boldsymbol{L}$ is the transition matrix, $\boldsymbol{\Phi}$ is the offset from the latent space to attribute space; note that the stationarity assumption encoded in the model.

In this model, the latent variables capture the dependence between the attributes. They are conditionally independent given the class but marginally dependent. Furthermore, the same mapping, $\boldsymbol{L}$, from the latent space to the attribute space is used for all classes, and hence, the relation between the class and the attributes is conveyed by the latent variables only.

At this step, the temporal dynamics of the model is assumed to be only captured at the class level. When the state specification of the class variable is coarse, then this assumption will rarely hold. This assumption does not hold in our oil drilling data, as the conditional correlation of the attribute in successive time slices is evident [12]. we address this by modeling the dynamics of the system at the level of the latent variables. The state specific dynamics is encoded by assuming that the latent variable at the current time slice follows a linear Gaussian distribution conditioned on previous time slice. Specifically, we encode the state specific dynamics by assuming that the multivariate latent variable $\boldsymbol{Z}^t$ follows a linear Gaussian distribution conditioned on $\boldsymbol{Z}^{t-1}$, and the transition dynamics between latent variable is denoted by a diagonal matrix $\boldsymbol{A}_{c^t}$:

$$\boldsymbol{Z}^t|\{\boldsymbol{Z}^{t-1}=\boldsymbol{z}^{t-1}, C^t=c^t\} \sim N(\boldsymbol{A}_{c^t}\boldsymbol{z}^{t-1}, \boldsymbol{\Sigma}_{c^t})$$

A graphical representation of the model is given in Figure 3.

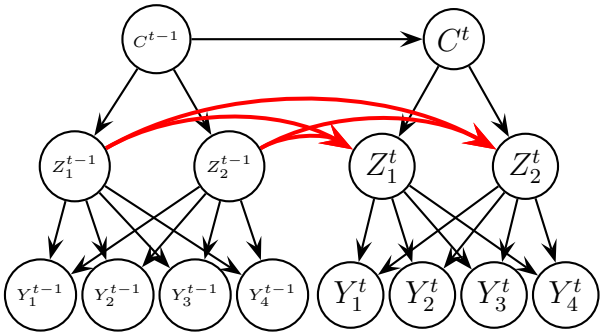


Figure 3: An example model by incrementally adding dynamics on latent variables using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4. The state specific dynamics are encoded at the level of the latent variables.

A discrete mixture variable M is further introduced to the model at each time slice for the purpose of reducing the computational cost while maintaining the representational power [12]. Similar situation is done by [7] for static domains, and in the dynamic domains can be seen from [3, 6] where a probabilistic model called switching state-space model is proposed that combining discrete and continuous dynamics. In this case, the mixture variable follows a multinomial distribution conditioned on the class variable. and the attributes $\boldsymbol{Y}^t$ follow a multivariate Gaussian distribution conditioned on the latent variables and the discrete mixture variable,

$$M^t|\{C^t=c^t\} \sim P(M^t|C^t=c^t),$$
$$\boldsymbol{Y}^t|\{\boldsymbol{Z}^t=\boldsymbol{z}^t, M^t=m^t\} \sim N(\boldsymbol{L}_{m^t}\boldsymbol{z}^t+\boldsymbol{\Phi}_{m^t}, \boldsymbol{\Theta}_{m^t}),$$

where $1 \le m^t \le |sp(M)|$ ($|sp(M)|$ denotes the dimension of variable M space), $P(M^t=m^t|C^t=c^t) \ge 0$ and $\sum_{m^t=1}^{|sp(M)|} P(M^t=m^t|C^t=c^t)=1$ for all $1 \le c^t \le |sp(C)|$, $\boldsymbol{\Phi}_{m^t}$ is the offset from the latent space to attribute space.

The final model is then called dynamic latent classification model which is shown in Figure 4. The dynamic latent classification model is shown to be effective and efficient through the experiment with our oil drilling data, and the significant improvement is also demonstrated when comparing dLCM with static models (such as NB or decision tree) and HMM [12].

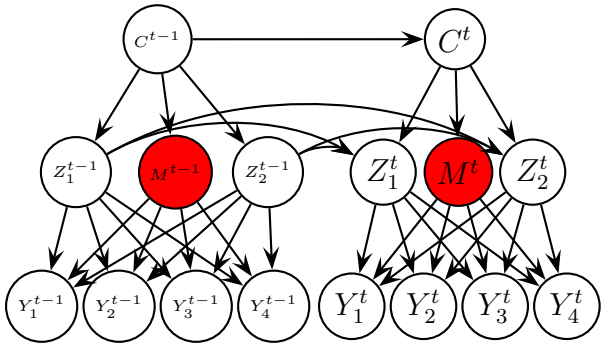


Figure 4: An example of dynamic latent classification model using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4.

3 Approximate inference in dLCM

The exact inference for dLCM is intractable. To make dLCM applicable and effective in practice, approximate inference is then proposed.

3.1 Intractability of exact inference in dLCM

Seen from the dLCM in Figure 4, an equivalent probabilistic model is

$$p(\boldsymbol{y}^{1:T}, \boldsymbol{z}^{1:T}, m^{1:T}, c^{1:T}) = p(\boldsymbol{y}^1|\boldsymbol{z}^1, m^1)p(\boldsymbol{z}^1|c^1)p(m^1|c^1)p(c^1) \cdot \prod_{t=2}^{T} p(\boldsymbol{y}^t|\boldsymbol{z}^t, m^t)p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, c^t)p(m^t|c^t)p(c^t|c^{t-1}).$$

In dLCM, exact filtered and smoothed inference is shown to be intractable (scaling exponentially with T [8]) as neither the class variables nor the mixture variables are observed: At time step 1, $p(\boldsymbol{z}^1|\boldsymbol{y}^1)$ is

a mixture of $|sp(C)| \cdot |sp(M)|$ Gaussian. At time-step 2, due to the summation over the classes and mixture variables, $p(\boldsymbol{z}^2|\boldsymbol{y}^{1:2})$ will be a mixture of $|sp(C)|^2 \cdot |sp(M)|^2$ Gaussian; at time-step 3 it will be a mixture of $|sp(C)|^3 \cdot |sp(M)|^3$ Gaussian and so on until the generation of a mixture of $|sp(C)|^T \cdot |sp(M)|^T$ Gaussian at time-point T. To control this explosion in computational complexity, approximate inference techniques are adopted to the inference of dLCM.

3.2 Approximate inference: Forward pass

The structure of the proposed dLCM is similar to the linear dynamical system (LDS) [2], the standard Rauch-Tung-Striebel (RTS) smoother [9] and the expectation correction smoother [3] for LDS provide the basis for the approximate inference of dLCM. As for the RTS, the filtered estimate of dLCM $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is obtained by a forward recursion, and then following a backward recursion to calculate the smoothed estimate $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$. The inference of dLCM is then achieved by a single forward recursion and a single backward recursion iteratively. Gaussian collapse is incorporated into both the forward recursion and the backward recursion to form the approximate inference. The Gaussian collapse in the forward recursion is equivalent to assumed density filtering [4], and the Gaussian collapse in the backward recursion mirrors the smoothed posterior collapse from [3].

During the forward recursion of dLCM, the filtered posterior $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is computed with a recursive form. By a simple decomposition,

$$\begin{aligned} p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t}) &= p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})/p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) \\ &\propto p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}). \end{aligned}$$

Dropping the normalization constant $p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$, $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is proportional to the new joint probability $p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$, where

$$\begin{aligned} p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) = p(\boldsymbol{y}^t, \boldsymbol{z}^t|m^t, c^t, \boldsymbol{y}^{1:t-1}) \cdot \\ p(m^t|c^t, \boldsymbol{y}^{1:t-1}) p(c^t|\boldsymbol{y}^{1:t-1}). \end{aligned} \quad (1)$$

To build the forward recursion, a recursive form for each of the factors in Equation 1 is required. Given the filtered results of the previous time-step, the recursive form for each of the factors are shown to be feasible [12]. On the way to devise the recursive form, one term $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ is required, which can be directly obtained since it is the filtered probability from the previous time step. However, the number of mixture components of $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{t-1})$ is increasing exponentially over time as we discussed earlier, so is the case for $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$. In our Gaussian collapse implementation [12], the term $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ is collapsed into a single Gaussian, parameterized with mean $\boldsymbol{\nu}_{m^{t-1}, c^{t-1}}$ and covariance $\boldsymbol{\Gamma}_{m^{t-1}, c^{t-1}}$, and then propagate this collapsed Gaussian for next time slice. With this approximation, the recursive computation of the forward pass becomes tractable.

3.3 Approximate inference: Backward pass

Similar to the forward pass, the backward pass also relies on a recursion computation of the smoothed posterior $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$. In detail, $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ is computed from its smoothed result of the previous step $p(\boldsymbol{z}^{t+1}, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T})$, together with some other quantities obtained from forward pass. The first smoothed posterior is $p(\boldsymbol{z}^T, m^T, c^T|\boldsymbol{y}^{1:T})$, which can be directly obtained as it is also the last filtered posterior from the forward pass. To compute $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$, factorize it as

$$\begin{aligned} &p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T}) \\ &\quad = \sum_{m^{t+1}, c^{t+1}} p(\boldsymbol{z}^t, m^t, c^t, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T}) \\ &\quad = \sum_{m^{t+1}, c^{t+1}} p(\boldsymbol{z}^t|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) \cdot \\ &\quad p(m^t, c^t|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) p(m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T}). \end{aligned}$$

Due to the fact that $\boldsymbol{z}^t \perp\!\!\!\perp \{\boldsymbol{y}^{t+1:T}, m^{t+1}, c^{t+1}\}|\{\boldsymbol{z}^{t+1}, m^t, c^t\}$, the term $p(\boldsymbol{z}^t|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ can be found from

$$\begin{aligned} &p(\boldsymbol{z}^t|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) \\ &\quad = \int_{\boldsymbol{z}^{t+1}} p(\boldsymbol{z}^t|\boldsymbol{z}^{t+1}, m^t, c^t, \boldsymbol{y}^{1:t}) \cdot \\ &\quad p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) \mathrm{d}\boldsymbol{z}^{t+1}. \end{aligned}$$

To complete the backward recursive form, two essential assumptions are further made in the backward pass that makes the approximate inference applicable and effective. The first assumption is to approximate $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ [3]. This is due to that although $\{m^t, c^t\} \not\perp\!\!\!\perp \boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T}$, the influence of $\{m^t, c^t\}$ on $\boldsymbol{z}^{t+1}$ through $\boldsymbol{z}^t$ is 'weak' as $\boldsymbol{z}^t$ will be mostly influenced by $\boldsymbol{y}^{1:t}$. The benefit of this simple assumption lies in that $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ can be directly obtained from the previous backward recursion. Meanwhile $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is a Gaussian mixture whose components increase exponentially in $T - t$.

The second assumption is also a Gaussian collapse process. $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is collapsed into a single Gaussian and then pass this collapsed Gaussian for the next step. This will guarantee that the back propagated term $p(\boldsymbol{z}^t|m^t, c^t, \boldsymbol{y}^{1:T})$ will be Gaussian mixture with fixed $|sp\,(C)| \cdot |sp\,(M)|$ components at next time step. With this Gaussian collapse process at each time slice, a tractable recursion in backward pass is established.

3.4 The importance of approximate inference

The exact inference is not applicable in practise as its computation cost is increasing exponentially over time. The approximate inference is then essential to dLCM. Gaussian collapse is adopted during building the recursive form for both forward and backward pass. At the same time, $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is also approximated by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1})$ in dLCM. As the approximations are made within the inference, the quality of the overall learning and inference for dLCM is rather sensitive to these approximations. Our experimental results [12] showed that the overall performance of dLCM is satisfactory, which indicate that the chosen approximations are reasonable.

Even though the proposed approximate inference in dLCM is satisfactory, is there any improvement space with alternative approximation methods? With this question in mind, we decide to investigate the approximation's properties by incorporating new approximation method. The traditional sampling techniques (e.g., [5]) are commonly used in a similar approximation situation. Next section we will briefly introduce sampling technique, and then we will explain how it is integrated in dLCM. Meanwhile the approximation of $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1})$ is kept unchanged.

In general, we will replace Gaussian collapse by sampling in the approximate inference of dLCM, and further investigate the effectiveness and efficiency of this proposal through a comparison experiment between original Gaussian collapse based dLCM and sampling techniques based dLCM.

4 Sampling

4.1 Background

The sampling is to select a subset of samples from within a population to estimate the characteristics of the original population. There is a commonly known tradeoff in sampling. When less samples are selected from within a population, which means the sampling process takes shorter time, the estimation of the characteristics to the original population is relatively worse. On the other hand, if more samples are selected from within the same population, which of course is much more time consuming, the characteristics of the original population is better estimated. The efficiency (time consuming) and effectiveness (characteristics estimation) are the essential concerns in the sampling techniques. In general, more samples should be selected within the tolerable time, and the better estimation of characteristics of the population can be expected.

This feature of traditional sampling techniques makes it attractive to the approximate inference of dLCM, a balance between efficiency and effectiveness is expected to be achieved according to application requirement. Meanwhile sampling can approximate any distribution as long as the sample number is sufficient. Sampling is expected to replace the Gaussian collapse for the approximation in the both forward and backward pass. We introduce particle filtering next, which will further motivate our discussion on the utilizations detail of the sampling in the inference of dLCM.

4.2 Particle Filtering

Particle filtering (PF) [1] is a technique for implementing a recursive Bayesian filter by Monte Carlo simulation, which is an efficient statistical method to estimate the system state. The Monte Carlo simulation relies on repeated random sampling techniques.

In particle filtering, let a weighted particle set $\{(\boldsymbol{s}_n^t, \boldsymbol{\pi}_n^t)\}_{n=1}^N$ at each time t denotes an approximation of required posterior probability of the system state. Each of N particles has the state s_n^t and its weight $\boldsymbol{\pi}_n^t$, the weights are normalized such that $\sum_n \boldsymbol{\pi}_n^t = 1$. The particle filtering has three operation stages: sampling (selection), prediction and observation. In the sampling stage, N particles are chosen from the prior probability according to the set $\{(\boldsymbol{s}_n^{(t-1)}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$. Then predict the state of the chosen particles by the dynamic model $p(\boldsymbol{s}^t|\boldsymbol{s}^{t-1})$. In observation stage, the predicted particles are weighted according to observation model $p(\boldsymbol{y}^t|\boldsymbol{s}^t)$. After obtaining the weights of particles, the state at time t can be estimated based on the weighted particle set.

4.3 Sampling in the dCLM

The sampling process that we required for the inference of dCLM is similar to the PF. In the forward pass, we know that the mixture components of $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{t-1})$ is increasing exponentially over time in the exact inference. Instead of a recursive approximation on $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ in the Gaussian collapse scheme, an recursive approximation on $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ by sampling is adopted. With the obtained approxi-

mated distribution $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ at time slice $t-1$, N weighted samples $\{(\boldsymbol{s}_n^{t-1}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$ are selected from this approximated distribution. These selected samples are propagated to the next time slice t with a linear transition dynamics $\boldsymbol{A}_{c^t}$. As the discrete class variable C^t has the size of $|sp\,(C)|$, then each of the selected samples will become $|sp\,(C)|$ new samples. These $|sp\,(C)|\cdot N$ propagated samples are further updated by the observation $\boldsymbol{y}^t$. The updating rule is the same as the Kalmar filter updating [11]. Due to the mixture component has size of $|sp\,(M)|$, each of these propagated samples will become $|sp\,(M)|$ new samples again. In general, the N selected samples from time slice $t-1$ will become $|sp\,(C)|\cdot|sp\,(M)|\cdot N$ samples at time slice t and its weight are updated accordingly. The weighted sample set $\{(\boldsymbol{f}_n^t, \boldsymbol{\gamma}_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$ is then the approximation to $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$. For next time step recursion, a new weighted sample set $\{(\boldsymbol{s}_n^t, \boldsymbol{\pi}_n^t)\}_{n=1}^N$ containing N samples will be selected from the approximated $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$. The recursive process is summarized in Algorithm 1.

Algorithm 1 Sampling in the forward pass

1: **for** $t = 2 : T$ **do**
2: Select N samples from the previous appxoximated distribution $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ to form a weighted sample set $\{(\boldsymbol{s}_n^{t-1}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$
3: Propagate these selected samples to the next time slice t by a transition dynamics $\boldsymbol{A}_{c^t}$
4: Update the propagated samples by the observation Y^t.
5: Then the updated samples form a new weighted sample set $\{(\boldsymbol{f}_n^t, \boldsymbol{\gamma}_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$, which is an approximation of $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$
6: **end for**

In the backward pass, with a similar sampling process as the forward pass, samples are firstly selected from the approximated distribution $p(\boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T})$. $p(\boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T})$ is approximated by a weighted sample set denoted as $\{(\boldsymbol{b}_n^{t+1}, \boldsymbol{\rho}_n^{t+1})\}_{n=1}$. Next, the selected samples are then back-propagated to the previous time slice t (which is the next step of the backward pass) with the reverse transition dynamics of $\boldsymbol{A}_c^t$. The back-propagated samples is later updated by the observation Y^{t-1} [11]. Similar to the forward pass, the approximation to $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:T})$. $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:T})$ is a weighted sample set $\{(\boldsymbol{g}_n^t, \boldsymbol{\tau}_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$. For the recursive calculation of next time slice, a new weighted sample set $\{(\boldsymbol{b}_n^t, \boldsymbol{\rho}_n^t)\}_{n=1}^N$ containing N samples will be selected from this approximated distribution. The required term $p(\boldsymbol{z}^T|c^T, \boldsymbol{y}^{1:T})$ at the beginning of the backward pass is also the last time step result from the forward pass, which indicates that $\{(\boldsymbol{g}_n^T, \boldsymbol{\tau}_n^T)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$ is the same sample set as $\{(\boldsymbol{f}_n^T, \boldsymbol{\gamma}_n^T)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$. Finally the approximate inference for dLCM is completely established with sampling technique based scheme.

The number of samples selected from the approximated distribution at each step is fixed which is dependent on the time consumption requirement and estimation quality requirement of corresponding application. There is a balance need to be addressed according to the practical application requirement. Generally, the more samples we select, the more time it costs while the estimation quality is better.

In the discussion of this section, the approximate inference of dLCM based on sampling is established by mimicking the particle filtering process both in the forward and backward pass. To investigate the effectiveness and efficiency of sampling based dLCM, a comparison experiments test will be conducted in the next section.

5 Experiment Results

In this section, the comparison experiments on simulation data and oil drilling data are conducted and their results are discussed.

5.1 Experiments on simulation data

A set of simulation data is firstly generated from dLCM, and we investigate the classification accuracy and time-consumption between Gaussian collapse scheme and sampling scheme.

5.1.1 Experiments settings on simulation data generation

The simulation data-set are generated from dLCM with parameters that is chosen by a "semi random" process. The model parameters of dLCM have two parts: model structure and model parameters with fixed model structure. For each time slice, the model structure is decided by four factors: the size of class variable C (activity state), the dimension of latent variable space $\boldsymbol{Z}$, the dimension of attribute space $\boldsymbol{Y}$ and the size of mixture components M. These values are fixed as described in Table 1. After choosing the model structure, its associated model parameters were randomly generated. The above process of choosing model structure and model parameters together is the "semi random" process. We then generate a data set with this model.

For convenience we call the model structure and its parameters as model parameters in the remaining of the paper. The generated data set and the model parameters are used as true model for the classification test purpose next. A comparison experiment between

data	$\|sp(C)\|$	$\|sp(M)\|$	$\|sp(Z)\|$	$\|sp(Y)\|$	T
*set*1	2	1	12	6	500
*set*2	2	2	18	9	1000

Table 1: The simulation data set with its model structure information.

Gaussian collapse and sampling is then conducted.

5.1.2 Results and discussion

The comparison experiment is conducted with both Gaussian collapse based and sampling based dLCM. Among sampling based scheme, there are three chosen sample sizes $40, 200, 1000$ respectively. The classification results on simulation set1 and set2 are summarized in Table 2. The classification accuracy results scheme are recorded with the average results of ten runs of each scheme, and it shows that Gaussian collapse based dLCM performs better than three sampling based dLM in both set1 and set2. Among sampling based scheme, scheme with larger sample size achieves better classification accuracy in a general sense.

scheme (samples)	set1/accuracy	set2/accuracy
Gaussian collapse	**99.60**%	**99.90**%
Sampling (40)	96.75%	95.55%
Sampling (200)	97.60%	97.95%
Sampling (1000)	97.75%	98.40%

Table 2: The average classification accuracy on simulation data set with Gaussian collapse based and sampling based (varying the number of samples) dLCM.

After investigating the effectiveness of each scheme, we continue to discuss the efficiency. The efficiency of each scheme is evaluated by the average time-cost of ten run that is required to accomplish the classification task, and the time-cost detail is shown in Table 3. In set1, the classification task is accomplished 0.47 second with Gaussian collapse, whereas the sampling scheme with 40 samples cost 2.01 second. The larger sample size in sampling scheme, the more time it costs to accomplish the classification task. Meanwhile it is clear that Gaussian collapse requires much less time to accomplish the classification task.

Compared to sampling based scheme, Gaussian collapse based scheme achieves comparable (slightly better) classification results with much less time on simulation data test.

5.2 Experiments on oil drilling data

Next the same comparison experiment is conducted with the oil drilling data from North sea.

scheme (samples)	set1/time-cost	set2/time-cost
Gaussian Collapse	**0.47(*s*)**	**1.91(*s*)**
Sampling (40)	2.01 (s)	5.97(s)
Sampling (200)	7.31(s)	17.56 (s)
Sampling (1000)	36.24 (s)	80.70 (s)

Table 3: The average time-cost (second) on simulation data set with both Gaussian collapse based and sampling based (varying the number of samples) dLCM.

5.2.1 Experiment settings on oil drilling data

As we mentioned in the introduction section, we will tie the development to the task of *activity recognition* in this paper. In total, there are 5 drilling acclivities in the dataset used for classification task. These activities are "drilling", "connection", "tripping in", "tripping out" and "other". The original oil drilling data contains more than 60 variables. Advised by oil drilling domain expert, 9 variables for the classification task here. There are two chosen data set, which contains 80000 and 50000 time slices with all 5 activities presented respectively.

For classification purpose in this paper, we combine these 5 activities into 2 activities and conduct the classification test on the combined data set. Three activities including "drilling", "connection" and "other" activities are combined as one activity, and we do the similar combination for "tripping in" and "tripping out" activities. The reason behind is that these combined actives are physically close and may have quite similar dynamics. This combination also simplify our experiments with the oil drilling data, while maintaining the comparison experiment purpose.

Before we can compare the inference of each scheme on the oil drilling data set, we learn a dLCM with the learning method proposed in [12] with the oil drilling data set containing 80000 time slices. The model structure is chosen by experience, with 2 mixture component and 16 latent variables. After learning its parameters with the chosen model structure, the dLCM for further classification experiment is then finalized. With the learnt dLM, the classification experiment will be conducted on another oil drilling data set containing 50000 time slices.

5.2.2 Results and discussion

With the fixed dLCM, the average (by ten runs) classification accuracy and average time-cost for each scheme are obtained. There are 4 scheme are presented, Gaussian collapsed based scheme and sampling techniques based scheme with $40, 200, 1000$ samples respectively. The experiments results are summarized in Table 4.

scheme (samples)	accuracy	time-cost
Gaussian Collapse	**82.9**%	**110.15(s)**
Sampling (40)	69.87%)	335.93(s)
Sampling (200)	76.56%	1130.85 (s)
Sampling (1000)	79.07%	4469.33 (s)

Table 4: The average classification accuracy and time-cost (second) on oil drilling data set with both Gaussian collapse based and sampling based (varying the number of samples) dLCM.

Among the sampling techniques based scheme, more samples achieves higher classification accuracy. However, with more samples in sampling techniques based scheme, the computation cot for the classification task is much more expensive. It is clearly shown in the table that sampling with 1000 samples requires more than one hour to accomplish the classification task which is around 40 times than that of Gaussian collapse, and they achieve a similar classification accuracy. In general Gaussian collapse still achieves comparable results (slightly better than Sampling), while keeping the computation cost in a rather low standard compared to sampling based scheme.

6 Conclusion

In the approximate inference of the dLCM, the Gaussian collapse is originally adopted as the core of the approximation method. In this paper, alternatively sampling technique is proposed to do the approximation. A process similar to particle filtering, utilizing sampling as the basis, is then incorporated into the approximate inference of the dLCM. We then conduct the comparison experiment results on both simulated data and real oil drilling data. The experimental results from both sets show that the approximate scheme based on Gaussian collapse is computationally more efficient than sampling, while offering comparable accuracy results.

Acknowledgements

I would like to thank Helge Langseth who helped a lot during the whole process of organizing and writing this paper.

References

[1] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques and software*. Artech House Publishers, 1993.

[3] David Barber. Expectation correction for smoothed inference in switching linear dynamical systems. *Journal of Machine Learning Research*, 7:2515–2540, 2006.

[4] Xavier Boyen and Daphne Koller. Approximate learning of dynamic models. In *Advances in Neural Information Processing Systems 12*, pages 396–402, 1999.

[5] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[6] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:963–996, 1998.

[7] Helge Langseth and Thomas D. Nielsen. Latent classification models. *Machine Learning*, 59(3):237–265, 2005.

[8] Uri Lerner. Hybrid Bayesian networks for reasoning about complex systems. *PhD thesis, Dept. of Comp. Sci. Stanford University, Stanford*, 2002.

[9] H.E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3:1445–1450, 1965.

[10] Padhraic Smyth. Hidden Markov models for fault detection in dynamic system. *Pattern Recognition*, 27(1):149–164, 1994.

[11] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[12] Shengtong Zhong, Helge Langseth, and Thomas D. Nielsen. Bayesian networks for dynamic classification. Working Paper, `http://idi.ntnu.no/~shket/dLCM.pdf`, 2012.

Author Index