A Comparison of Two MCMC Algorithms for Hierarchical Mixture Models

Russell G. Almond[*]
Florida State University

Abstract

Mixture models form an important class of models for unsupervised learning, allowing data points to be assigned labels based on their values. However, standard mixture models procedures do not deal well with rare components. For example, pause times in student essays have different lengths depending on what cognitive processes a student engages in during the pause. However, instances of student planning (and hence very long pauses) are rare, and thus it is difficult to estimate those parameters from a single student's essays. A hierarchical mixture model eliminates some of those problems, by pooling data across several of the higher level units (in the example students) to estimate parameters of the mixture components. One way to estimate the parameters of a hierarchical mixture model is to use MCMC. But these models have several issues such as non-identifiability under label switching that make them difficult to estimate just using off-the-shelf MCMC tools. This paper looks at the steps necessary to estimate these models using two popular MCMC packages: JAGS (random walk Metropolis algorithm) and Stan (Hamiltonian Monte Carlo). JAGS, Stan and R code to estimate the models and model fit statistics are published along with the paper.

Key words: Mixture Models, Markov Chain Monte Carlo, JAGS, Stan, WAIC

[*]Paper presented at the Bayesian Application Workshop at Uncertainty in Artificial Intelligence Conference 2014, Quebec City, Canada.

1 Introduction

Mixture models (McLachlan & Peel, 2000) are a frequently used method of unsupervised learning. They sort data points into clusters based on just their values. One of the most frequently used mixture models is a mixture of normal distributions. Often the mean and variance of each cluster is learned along with the classification of each data point.

As an example, Almond, Deane, Quinlan, Wagner, and Sydorenko (2012) fit a mixture of lognormal distributions to the pause time of students typing essays as part of a pilot writing assessment. (Alternatively, this model can be described as a mixture of normals fit to the log pause times.) Almond et al. found that mixture models seems to fit the data fairly well. The mixture components could correspond to different cognitive process used in writing (Deane, 2012) where each cognitive process takes different amounts of time (i.e., students pause longer when planning, than when simply typing).

Mixture models are difficult to fit because they display a number of pathologies. One problem is component identification. Simply swapping the labels of Components 1 and 2 produces a model with identical likelihoods. Even if a prior distribution is placed on the mixture component parameters, the posterior is multimodal. Second, it is easy to get a pathological solution in which a mixture component consists of a single point. These solutions are not desirable, and some estimation tools constrain the minimum size of a mixture component (Gruen & Leisch, 2008). Furthermore, if a separate variance is to be estimated for each mixture component, several data points must be assigned to each component in order for the variance estimate to have a reasonable standard error. Therefore, fitting a model with rare components requires a large data size.

Almond et al. (2012) noted these limitations in their conclusions. First, events corresponding to the highest-level planning components in Deane (2012)'s cognitive model would be relatively rare, and hence would lumped in with other mixture components due to size restrictions. Second, some linguistic contexts (e.g., between Sentence pauses) were rare enough that fitting a separate mixture model to each student would not be feasible.

One work-around is a hierarchical mixture model. As with all hierarchical models, it requires units at two different levels (in this example, students or essays are Level 2 and individual pauses are Level 1). The assumption behind the hierarchical mixture model is that the mixture components will look similar across the second level units. Thus, the mean and variance of Mixture Component 1 for Student 1 will look similar to those for Student 2. Li (2013) tried this on some of the writing data.

One problem that frequently arises in estimating mixture models is determining how many mixture components to use. What is commonly done is to estimate models for $K = 2, 3, 4, \ldots$ up to some small maximum number of components (depending on the size of the data). Then a measure of model–data fit, such as AIC, DIC or WAIC (see Gelman et al., 2013, Chapter 7), is calculated for each model and the model with the best fit index is chosen. These methods look at the *deviance* (minus twice the log likelihood of the data) and adjust it with a penalty for model complexity. Both DIC and WAIC require Markov chain Monte Carlo (MCMC) to compute, and require some custom coding for mixture models because of the component identification issue.

This paper is a tutorial for replicating the method used by Li (2013). The paper walks through a script written in the R language (R Core Team, 2014) which performs most of the steps. The actual estimation is done using MCMC using either Stan (Stan Development Team, 2013) or JAGS (Plummer, 2012). The R scripts along with the Stan and JAGS models and some sample data are available at `http://pluto.coe.fsu.edu/mcmc-hierMM/`.

2 Mixture Models

Let $i \in \{1, \ldots, I\}$ be a set of indexes over the second level units (students in the example) and let $j \in \{1, \ldots, J_i\}$ be the first level units (pause events in the example). A hierarchical mixture model is by adding a Level 2 (across student) distribution over the parameters of the Level 1 (within student) mixture model. Section 2.1 describes the base Level 1 mixture model, and Section 2.2 describes the Level 2 model. Often MCMC requires reparameterization to achieve better mixing (Section 2.3). Also, there are certain parameter values which result in infinite likelihoods. Section 2.4 describes prior distributions and constraints on parameters which keep the Markov chain away from those points.

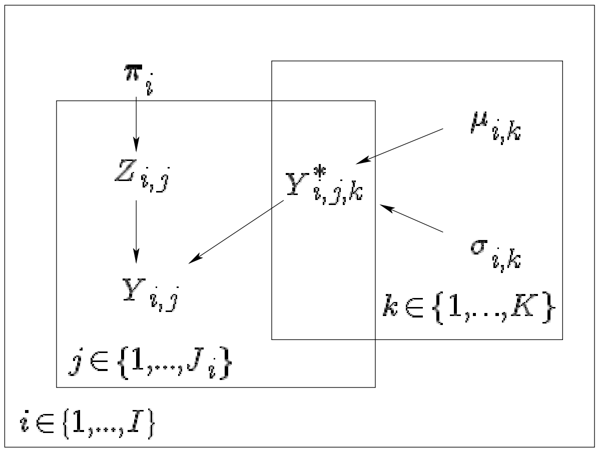


Figure 1: Non-hierarchical Mixture Model

2.1 Mixture of Normals

Consider a collection observations, $\mathbf{Y}_i = (Y_{i,1}, \ldots, Y_{i,J_i})$ for a single student, i. Assume that the process that generated these data is a mixture of K normals. Let $Z_{i,j} \sim \mathrm{cat}(\boldsymbol{\pi}_i)$ be a categorical latent index variable indicating which component Observation j comes from and let $Y^*_{i,j,k} \sim \mathcal{N}(\mu_{i,k}, \sigma_{i,k})$ be the value of $Y_{i,j}$ which would be realized when $Z_{i,j} = k$.

Figure 1 shows this model graphically. The plates indicate replication over categories (k), Level 1 (pauses, j) and Level 2 (students, i) units. Note that there is no connection across plates, so the model fit to each Level 2 unit is independent of all the other Level 2 units. This is what Gelman et al. (2013) call the no pooling case.

The latent variables Z and Y^* can be removed from the likelihood for $Y_{i,j}$ by summing over the possible values of Z. The likelihood for one student's data, $\mathbf{Y}_i$, is

$$L_i(\mathbf{Y}_i | \boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i) = \prod_{j=1}^{J_i} \sum_{k=1}^{K} \pi_{i,k} \phi(\frac{Y_{i,j} - \mu_{i,k}}{\sigma_{i,k}}) \qquad (1)$$

where $\phi(\cdot)$ is the unit normal density.

Although conceptually simple, there are a number of issues that mixture models can have. The first issue is that the component labels cannot be identified from data. Consider the case with two components. The model created by swapping the labels for Components 1 and 2 with new parameters $\boldsymbol{\pi}'_i = (\pi_{i,2}, \pi_{i,1})$, $\boldsymbol{\mu}'_i = (\mu_{i,2}, \mu_{i,1})$, and $\boldsymbol{\sigma}'_i = (\sigma_{i,2}, \sigma_{i,1})$ has an identical likelihood. For the K component model, any permutation of the component labels produces a model with identical likelihood. Consequently, the likelihood surface is multimodal.

A common solution to the problem is to identify the components by placing an ordering constraint on one of the three parameter vectors: $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ or $\boldsymbol{\sigma}$. Section 4.1 returns to this issue in practice.

A second issue involves degenerate solutions which contain only a single data point. If a mixture component has only a single data point, its standard deviation, $\sigma_{i,k}$ will go to zero, and $\pi_{i,k}$ will approach $1/J_i$, which is close to zero for large Level 1 data sets. Note that if either $\pi_{i,k} \to 0$ or $\sigma_{i,k} -> 0$ for any k, then the likelihood will become singular.

Estimating $\sigma_{i,k}$ requires a minimum number of data points from Component k ($\pi_{i,k} J_i > 5$ is a rough minimum). If $\pi_{i,k}$ is believed to be small for some k, then a large (Level 1) sample is needed. As K increases, the smallest value of $\pi_{i,k}$ becomes smaller so the minimum sample size increases.

2.2 Hierarchical mixtures

Gelman et al. (2013) explains the concept of a hierarchical model using a SAT coaching experiment that took place at 8 different schools. Let $X_i \sim \mathcal{N}(\mu_i, \sigma_i)$ be the observed effect at each school, where the school specific standard error σ_i is known (based mainly on the sample size used at that school). There are three ways to approach this problem: (1) *No pooling.* Estimate μ_i separately with no assumption about about the similarity of μ across schools. (2) *Complete pooling.* Set $\mu_i = \mu_0$ for all i and estimate μ_0 (3) *Partial pooling.* Let $\mu_i \sim \mathcal{N}(\mu_0, \nu)$ and now jointly estimate $\mu_0, \mu_1, \ldots, \mu_8$. The no pooling approach produces unbiased estimates for each school, but it has the largest standard errors, especially for the smallest schools. The complete pooling approach ignores the school level variability, but has much smaller standard errors. In the partial pooling approach, the individual school estimates are shrunk towards the grand mean, μ_0, with the amount of shrinkage related to the size of the ratio $\nu^{-2}/(\nu^{-2} + \sigma_i^{-2})$; in particular, there is more shrinkage for the schools which were less precisely measured. Note that the higher level standard deviation, ν controls the amount of shrinkage: the smaller ν is the more the individual school estimates are pulled towards the grand mean. At the limits, if $\nu = 0$, the partial pooling model is the same as the complete pooling model and if $\nu = \infty$ then the partial pooling model is the same as the no pooling model.

Li (2013) builds a hierarchical mixture model for the essay pause data. Figure 1 shows the no pool-

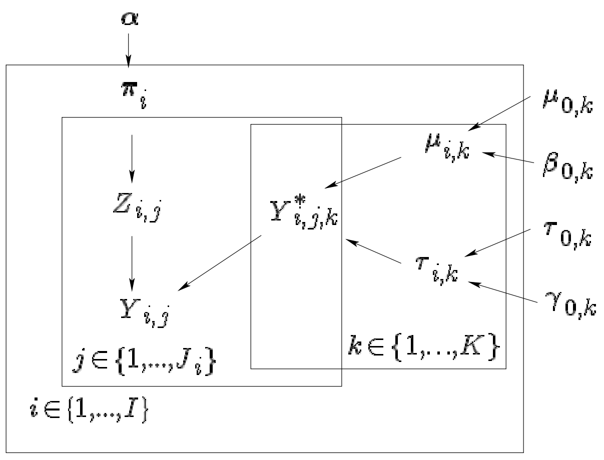


Figure 2: Hierarchical Mixture Model

ing mixture model. To make the hierarchical model, add across-student prior distributions for the student-specific parameters parameters, $\boldsymbol{\pi}_i$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$. Because JAGS parameterizes the normal distribution with precisions (reciprocal variances) rather than standard deviations, $\boldsymbol{\tau}_i$ are substituted for the standard deviations $\boldsymbol{\sigma}_i$. Figure 2 shows the hierarchical model.

Completing the model requires specifying distributions for the three Level 1 (student-specific) parameters. In particular, let

$$\boldsymbol{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,K}) \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_k) \quad (2)$$
$$\mu_{i,k} \sim \mathcal{N}(\mu_{0,k}, \beta_{0,k}) \quad (3)$$
$$\log(\tau_{i,k}) \sim \mathcal{N}(log(\tau_{0,k}), \gamma_{0,k}) \quad (4)$$

This introduces new Level 2 (across student) parameters: $\boldsymbol{\alpha}$, $\boldsymbol{\mu}_0$, $\boldsymbol{\beta}_0$, $\boldsymbol{\tau}_0$, and $\boldsymbol{\gamma}_0$. The likelihood for a single student, $L_i(\mathbf{Y}_i|\boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \boldsymbol{\tau}_i)$ is given (with a suitable chance of variable) by Equation 1. To get the complete data likelihood, multiply across the students units (or to avoid numeric overflow, sum the log likelihoods). If we let $\boldsymbol{\Omega}$ be the complete parameter ($\boldsymbol{\pi}_i$,$\boldsymbol{\mu}_i$, $\boldsymbol{\tau}_i$ for each student, plus $\boldsymbol{\alpha}$, $\boldsymbol{\mu}_0$, $\boldsymbol{\beta}_0$, $\boldsymbol{\tau}_0$, and $\boldsymbol{\gamma}_0$), then

$$\mathcal{L}(\mathbf{Y}|\boldsymbol{\Omega}) = \sum_{i=1}^{I} \log L_i(\mathbf{Y}_i|\boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \boldsymbol{\tau}_i) \; . \quad (5)$$

Hierarchical models have their own pathologies which require care during estimation. If either of the standard deviation parameters, $\beta_{0,k}$ or $\gamma_{0,k}$, gets too close to zero or infinity, then this could cause the log posterior to go to infinity. These cases correspond to the no pooling and complete pooling extremes of the hierarchical model. Similarly, the variance of the Dirichlet distribution is determined by $\alpha_N = \sum_{k=1}^{K} \alpha_k$. If α_N is too close to zero, this produces no pooling in the estimates of $\boldsymbol{\pi}_i$ and if α_N is too large, then there is nearly complete pooling in those estimates. Again, those values of α_N can cause the log posterior to become infinite.

2.3 Reparameterization

It is often worthwhile to reparameterize a model in order to make MCMC estimation more efficient. Both random walk Metropolis (Section 3.2) and Hamiltonian Monte Carlo (Section 3.3) work by moving a point around the parameter space of the model. The geometry of that space will influence how fast the Markov chain *mixes*, that is, moves around the space. If the geometry is unfavorable, the point will move slowly through the space and the autocorrelation will be high (Neal, 2011). In this case a very large Monte Carlo sample will be required to obtain reasonable Monte Carlo error for the parameter estimates.

Consider once more the Eight Schools problem where $\mu_i \sim \mathcal{N}(\mu_0, \nu)$. Assume that we have a sampler that works by updating the value of the μ_i's one at a time and then updating the values of μ_0 and ν. When updating μ_i, if ν is small then values of μ_i close to μ_0 will have the highest conditional probability. When updating ν if all of the values of μ_i are close to μ_0, then small values of ν will have the highest conditional probability. The net result is a chain in which the movement from states with small ν and the μ_i's close together to states with large ν and μ_i's far apart takes many steps.

A simple trick produces a chain which moves much more quickly. Introduce a series of auxiliary variables $\theta_i \sim \mathcal{N}(0, 1)$ and set $\mu_i = \mu_0 + \nu\theta_i$. Note that the marginal distribution of μ_i has not changed, but the geometry of the $\boldsymbol{\theta}, \mu_0, \nu$ space is different from the geometry of the $\boldsymbol{\mu}, \mu_0, \nu$ space, resulting in a chain that moves much more quickly.

A similar trick works for modeling the relationships between $\boldsymbol{\alpha}$ and $\boldsymbol{\pi}_i$. Setting $\alpha_k = \alpha_{0,k} * \alpha_N$, where $\boldsymbol{\alpha}_0$ has a Dirichlet distribution and α_N has a gamma distribution seems to work well for the parameters α of the Dirichlet distribution. In this case the parameters have a particularly easy to interpret meaning: $\boldsymbol{\alpha}_0$ is the expected value of the Dirichlet distribution and α_N is the effective sample size of the Dirichlet distribution.

Applying these two tricks to the parameters of the hierarchical model yields the following augmented parameterization.

$$\begin{aligned}
\alpha_k &= \alpha_{0,k} * \alpha_N && (6)\\
\mu_{i,k} &= \mu_{0,k} + \theta_{i,k}\beta_{0,k} && (7)\\
\theta_{i,k} &\sim \mathcal{N}(0,1) && (8)\\
\log(\tau_{i,k}) &= \log(\tau_{0,k}) + \eta_{i,k}\gamma_{0,k} && (9)\\
\eta_{i,k} &\sim \mathcal{N}(0,1) && (10)
\end{aligned}$$

2.4 Prior distributions and parameter constraints

Rarely does an analyst have enough prior information to completely specify a prior distribution. Usually, the analyst chooses a convenient functional form and chooses the hyperparameters of that form based on available prior information (if any). One popular choice is the *conjugate distribution* of the likelihood. For example, if the prior for a multinomial probability, $\boldsymbol{\pi}_i$ follows a Dirichlet distribution with hyperparameter, $\boldsymbol{\alpha}$, then the posterior distribution will also be a Dirichlet with a hyperparameter equal to the sum of the prior hyperparameter and the data. This gives the hyperparameter of a Dirichlet prior a convenient interpretation as pseudo-data. A convenient functional form is one based on a normal distribution (sometimes on a transformed parameter, such as the log of the precision) whose mean can be set to a likely value of the parameter and whose standard deviation can be set so that all of the likely values for the parameter are within two standard deviations of the mean. Note that for location parameters, the normal distribution is often a conjugate prior distribution.

Proper prior distributions are also useful for keeping parameter values away from degenerate or impossible solutions. For example, priors for standard deviations, $\beta_{0,k}$ and $\gamma_{0,k}$ must be strictly positive. Also, the group precisions for each student, $\boldsymbol{\tau}_i$, must be strictly positive. The natural conjugate distribution for a precision is a gamma distribution, which is strictly positive. The lognormal distribution, used in Equations 4 and 9, has a similar shape, but its parameters can be interpreted as a mean and standard deviation on the log scale. The mixing probabilities $\boldsymbol{\pi}_i$ must be defined over the unit simplex (i.e., they must all be between 0 and 1 and they must sum to 1), as must the expected mixing probabilities $\boldsymbol{\alpha}_0$; the Dirichlet distribution satisfies this constraint and is also a natural conjugate. Finally, α_N must be strictly positive; the choice of the chi-squared distribution as a prior ensures this.

There are other softer restraints on the parameters. If $\beta_{0,k}$ or $\gamma_{0,k}$ gets too high or too low for any value of k, the result is a no pooling or complete pooling solution on that mixture component. For both of these parameters $(.01, 100)$ seems like a plausible range. A lognormal distribution which puts the bulk of its probability mass in that range should keep the model away from those extremes. Values of $\tau_{i,k}$ that are too small represent the collapse of a mixture component onto a single point. Again, if $\tau_{0,k}$ is mostly in the range $(.01, 100)$ the chance of an extreme value of $\tau_{i,k}$ should be small. This yields the following priors:

$$\log(\beta_{0k}) \sim \mathcal{N}(0,1) \tag{11}$$
$$\log(\gamma_{0k}) \sim \mathcal{N}(0,1) \tag{12}$$
$$\log(\tau_{0k}) \sim \mathcal{N}(0,1) \tag{13}$$

High values of α_N also correspond to a degenerate solution. In general, the gamma distribution has about the right shape for the prior distribution of α_N, and we expect it to be about the same size as I, the number of Level-2 units. The choice of prior is a chi-squared distribution with $2 * I$ degrees of freedom.

$$\alpha_N \sim \chi^2(I * 2) \tag{14}$$

The two remaining parameters we don't need to constrain too much. For $\mu_{0,k}$ we use a diffuse normal prior (one with a high standard deviation), and for $\boldsymbol{\alpha}_0$ we use a Jeffrey's prior (uniform on the logistic scale) which is the Dirichlet distribution with all values set to 1/2.

$$\mu_{0,k} \sim N(0, 1000) \tag{15}$$
$$\boldsymbol{\alpha}_0 \sim \text{Dirichlet}(0.5, \ldots, 0.5) \tag{16}$$

The informative priors above are not sufficient to always keep the Markov chain out of trouble. In particular, it can still reach places where the log posterior distribution is infinite. There are two different places where these seem to occur. One is associated with high values of α_N. Putting a hard limit of $\alpha_N < 500$ seems to avoid this problem (when $I = 100$). Another possible degenerate spot is when $\alpha_k \approx 0$ for some k. This means that $\pi_{i,k}$ will be essentially zero for all students. Adding .01 to all of the α_k values in Equation 2 seems to fix this problem.

$$\boldsymbol{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,K}) \sim \text{Dirichlet}(\alpha_1 + .01, \ldots, \alpha_k + .01)$$

3 Estimation Algorithms

There are generally two classes of algorithms used with both hierarchical and mixture models. The expectation maximization (EM) algorithm (Section 3.1) searches for a set of parameter values that maximizes the log posterior distribution of the data (or if the prior distribution is flat, it maximizes the log likelihood). It comes up with a single parameter estimate but does not explore the entire space of the distribution. In contrast, MCMC algorithms explore the entire posterior distribution by taking samples from the space of possible values. Although the MCMC samples are not independent, if the sample is large enough it converges in distribution to the desired posterior. Two approaches to MCMC estimation are the random walk Metropolis algorithm (RWM; used by JAGS, Section 3.2) and the Hamiltonian Monte Carlo algorithm (HMC; used by Stan, Section 3.3).

3.1 EM Algorithm

McLachlan and Krishnan (2008) provides a review of the EM algorithm with emphasis on mixture models. The form of the EM algorithm is particularly simple for the special case of a non-hierarchical mixture of normals. It alternates between an E-step where the $p(Z_{i,j} = k) = p_{i,j,k}$ is calculated for every observation, j, and every component, k and an M-step where the maximum likelihood values for $\pi_{i,k}$, $\mu_{i,k}$ and $\sigma_{i,k}$ are found by taking moments of the data set $\mathbf{Y}_i$ weighted by the component probabilities, $p_{i,j,k}$.

A number of different problems can crop up with using EM to fit mixture models. In particular, if $\pi_{i,k}$ goes to zero for any k, that component essentially disappears from the mixture. Also, if $\sigma_{i,k}$ goes to zero the mixture component concentrates on a single point. Furthermore, if $\mu_{i,k} = \mu_{i,k'}$ and $\sigma_{i,k} = \sigma_{i,k'}$ for any pair of components the result is a degenerate solution with $K - 1$ components.

As the posterior distribution for the mixture model is multimodal, the EM algorithm only finds a local maximum. Running it from multiple starting points may help find the global maximum; however, in practice it is typically run once. If the order of the components is important, the components are typically relabeled after fitting the model according to a predefined rule (e.g., increasing values of $\mu_{i,k}$).

Two packages are available for fitting non-hierarchical mixture models using the EM algorithm in R (R Core Team, 2014): `FlexMix` (Gruen & Leisch, 2008) and `mixtools` (Benaglia, Chauveau, Hunter, & Young, 2009). These two packages take different approaches to how they deal with degenerate solutions. `FlexMix` will combine two mixture components if they get too close together or the probability of one component gets too small (by default, if $\pi_{i,k} < .05$). `Mixtools`, on the other hand, retries from a different starting point when the the EM algorithm converges on a degenerate solution. If it exceeds the allowed number of retries, it gives up.

Neither `mixtools` nor `FlexMix` provides standard errors for the parameter estimates. The `mixtools` package recommends using the bootstrap (resampling from the data distribution) to calculate standard errors, and provides a function to facilitate this.

3.2 Random-walk Metropolis Algorithm (RWM; used by JAGS)

Geyer (2011) gives a tutorial summary of MCMC. The basic idea is that a mechanism is found for constructing a Markov chain whose stationary distribution is the desired posterior distribution. The chain is run until the analyst is reasonably sure it has reach the stationary distribution (these early draws are discarded as *burn-in*). Then the the chain is run some more until it is believed to have *mixed* throughout the entire posterior distribution. At this point it has reached *pseudo-convergence* (Geyer calls this pseudo-convergence, because without running the chain for an infinite length of time, there is no way of telling if some part of the parameter space was never reached.) At this point the mean and standard error of the parameters are estimated from the the observed mean and standard deviation of the parameter values in the MCMC sample.

There are two sources of error in estimates made from the MCMC sample. The first arises because the observed data are a sample from the universe of potential observations. This sampling error would be present even if the posterior distribution could be computed exactly. The second is the Monte Carlo error that comes from the estimation of the posterior distribution with the Monte Carlo sample. Because the draws from the Markov chain are not statistically independent, this Monte Carlo error does not fall at the rate of $1/\sqrt{R}$ (where R is the number of Monte Carlo samples). It is also related to the autocorrelation (correlation between successive draws) of the Markov chain. The higher the autocorrelation, the lower the *effective sample size* of the Monte Carlo sample, and the higher the Monte Carlo error.

Most methods for building the Markov chain are based on the Metropolis algorithm (see Geyer, 2011, for details). A new value for one or more parameter is proposed and the new value is accepted or rejected randomly according to the ratio of the posterior distribution and the old and new points, with a correction factor for the mechanism used to generate the new sample. This is called a *Metropolis* or *Metropolis-Hastings* update (the latter contains a correction for asymmetric proposal distributions). *Gibbs sampling* is a special case in which the proposal is chosen in such a way that it will always be accepted.

As the form of the proposal distribution does not matter for the correctness of the algorithm, the most common method is to go one parameter at a time and add a random offset (a step) to its value, accepting or rejecting it according to the Metropolis rule. As this distribution is essentially a random walk over the parameter space, this implementation of MCMC is called *random walk Metropolis* (RWM). The step size is a critical tuning parameter. If the average step size is too large, the value will be rejected nearly every cycle and the autocorrelation will be high. If the step size is too small, the chain will move very slowly through the space and the autocorrelation will be high. Gibbs sampling, where the step is chosen using a conjugate distribution so the Metropolis-Hastings ratio always accepts, is not necessarily better. Often the effective step size of the Gibbs sampler is small resulting in high autocorrelation.

Most packages that use RWM do some adaptation on the step size, trying to get an optimal rejection rate. During this adaptation phase, the Markov chain does not have the correct stationary distribution, so those observations must be discarded, and a certain amount of burn-in is needed after the adaptation finishes.

MCMC and the RWM algorithm were made popular by their convenient implementation in the BUGS software package (Thomas, Spiegelhalter, & Gilks, 1992). With BUGS, the analyst can write down the model in a form very similar to the series of equations used to describe the model in Section 2, with a syntax derived from the R language. BUGS then compiles the model into pseudo-code which produces the Markov chain, choosing to do Gibbs sampling or random walk Metropolis for each parameter depending on whether or not a convenient conjugate proposal distribution was available. The output could be exported in a form that could be read by R, and the R package `coda` (Plummer, Best, Cowles, & Vines, 2006) could be used to process the output. (Later, WinBUGS would build some of that output processing into BUGS.)

Although BUGS played an important role in encouraging data analysts to use MCMC, it is no longer actively supported. This means that the latest developments and improvements in MCMC do not get incorporated into its code. Rather than use BUGS, analysts are advised to use one of the two successor software packages: OpenBUGS (Thomas, O'Hara, Ligges, & Sturtz, 2006) or JAGS (just another Gibbs sampler; Plummer, 2012). The R package `rjags` allows JAGS to be called from R, and hence allows R to be used as a scripting language for JAGS, which is important for serious analytic efforts. (Similar packages exist for BUGS and OpenBUGS.)

3.3 Hamiltonian Monte Carlo (HMC; used by Stan)

Hamiltonian Monte Carlo (HMC) (Neal, 2011) is a variant on the Metropolis Algorithm which uses a different proposal distribution than RWM. In HMC, the current draw from the posterior is imagined to be a small particle on a hilly surface (the posterior distribution). The particle is given a random velocity and is allowed to move for several discrete steps in that direction. The movement follows the laws of physics, so the particle gains speed when it falls down hills and loses speed when it climbs back up the hills. In this manner a proposal is generated that can be a great distance from the original starting point. The proposed point is then accepted or rejected according to the Metropolis rule.

The software package Stan (Stan Development Team, 2013) provides support for HMC. As with BUGS and JAGS, the model of Section 2 is written in pseudocode, although this time the syntax looks more like C++ than R. Rather than translate the model into interpreted code, Stan translates it into C++ then compiles and links it with existing Stan code to run the sampler. This has an initial overhead for the compilation, but afterwards, each cycle of the sampler runs faster. Also, as HMC generally has lower autocorrelation than random walk Metropolis, smaller run lengths can be used, making Stan considerably faster than JAGS in some applications. A package `rstan` is available to link Stan to R, but because of the compilation step, it requires that the user have the proper R development environment set up.

HMC has more tuning parameters than random walk Metropolis: the mass of the particle, the distribution of velocities and the number of steps to take in each direction must be selected to set up the algorithm. Stan uses a *warm-up* phase to do this adaptation. The recommended procedure is to use approximately 1/2 the samples for warm-up as a longer warm-up produces lower autocorrelations when actively sampling.

Stan has some interesting features that are not present in BUGS or JAGS. For one, it does not require every parameter to have a proper prior distribution (as long as the posterior is proper). It will simply put a uniform prior over the space of possible values for any parameter not given a prior distribution. However, using explicit priors has some advantages for the application to student pause data. In particular, when data for a new student become available, the posterior parameters for the previous run can be input into Stan (or JAGS) and the original calibration model can be reused to estimate the parameters for new student (Mislevy, Almond, Yan, & Steinberg, 1999).

3.4 Parallel Computing and Memory Issues

As most computers have multiple processors, parallel computing can be used to speed up MCMC runs. Often multiple Markov chains are run and the results are compared to assess pseudo-convergence and then combined for inference (Gelman & Shirley, 2011). This is straightforward using the output processing package `coda`. It is a little bit trickier using the `rstan` package, because many of the graphics require a full `stanfit` object. However, the conversion from Stan to coda format for the MCMC samples is straightforward.

In the case of hierarchical mixture models, there is an even easier way to take advantage of multiple processes. If the number of components, K, is unknown, the usual procedure is to take several runs with different values of K and compare the fit. Therefore, if 4 processors were available, one could run all of the chains for $K = 2$, one for $K = 3$, and one for $K = 4$, leaving one free to handle operating system tasks.

In most modern computers, the bottleneck is usually not available CPU cycles, but available memory. For running 3 chains with $I = 100$ students and $R = 5000$ MCMC samples in each chain, the MCMC sample can take up to 0.5GB of memory! Consequently, it is critically important to monitor memory usage when running multiple MCMC runs. If the computer starts requiring swap (disk-based memory) in addition to physical memory, then running fewer processes will probably speed up the computations.

Another potential problem occurs when storing the result of each run between R sessions in the `.RData` file. Note that R attempts to load the entire contents of that data file into memory when starting R. If there are the results of several MCMC runs in there, the `.RData` file can grow to several GB in size, and R can take several minutes to load. (In case this problem arises, it is recommended that you take a make a copy of the `.RData` after the data have been cleaned and all the auxiliary functions loaded but before starting the MCMC runs, and put it someplace safe. If the `.RData` file gets to be too large it can be simply replaced with the backup.)

In order to prevent the `.RData` file from growing unmanageably large, it recommended that the work space not be saved at the end of each run. Instead, run a block of code like this

```
assign(runName,result1)
outfile <-
     gzfile(paste(runName,"R","gz",sep="."),
         open="wt")
dump(runName,file=outfile)
close(outfile)
```

after all computations are completed. Here `result1` is a variable that gathers together the portion of the results to keep, and `runName` is a character string that provides a unique name for the run.

Assume that all of the commands necessary to perform the MCMC analysis are in a file `script.R`. To run this from a command shell use the command:

```
R CMD BATCH --slave script.R
```

This will run the R code in the script, using the `.RData` file in the current working directory, and put the output into `script.Rout`. The `--slave` switch performs two useful functions. First, it does not save the `.RData` file at the end of the run (avoiding potential memory problems). Second, it suppresses echoing the script to the output file, making the output file easier to read. Under Linux and Mac OS X, the command

```
nohup R CMD BATCH --slave script.R &
```

runs R in the background. The user can log out of the computer, but as long as the computer remains on, the process will continue to run.

4 Model Estimation

A MCMC analysis always follows a number of similar steps, although the hierarhical mixture model requires a couple of additional steps. Usually, it is best to write these as a script because the analysis will often need to be repeated several times. The steps are as follows:

1. *Set up parameters for the run.* This includes which data to analyze, how many mixture components are required (i.e., K), how long to run the Markov chain, how many chains to run, what the prior hyperparameters are.

2. *Clean the data.* In the case of hierarchical mixture models, student data vectors which are too short should be eliminated. Stan does not accept missing values, so `NA`s in the data need to be replaced with a finite value. For both JAGS and Stan the data are bundled with the prior hyperparameters to be passed to the MCMC software.

3. *Set initial values.* Initial values need to be chosen for each Markov chain (see Section 4.2).

4. *Run the Markov Chain.* For JAGS, this consists of four substeps: (a) run the chain in adaptation mode, (b) run the chain in normal mode for the burn-in, (c) set up monitors on the desired parameters, and (d) run the chain to collect the MCMC sample. For Stan, the compilation, warm-up and sampling are all done in a single step.

5. *Identify the mixture components.* Ideally, the Markov chains have visited all of the modes of the posterior distribution, including the ones which differ only by a permutation of the component labels. Section 4.1 describes how to permute the component labels are permuted so that the component labels are the same in each MCMC cycle.

6. *Check pseudo-convergence.* Several statistics and plots are calculated to see if the chains have reached pseudo-convergence and the sample size is adequate. If the sample is inadequate, then additional samples are collected (Section 4.3).

7. *Draw Inferences.* Summaries of the posterior distribution for the the parameters of interest are computed and reported. Note that JAGS offers some possibilities here that Stan does not. In particular, JAGS can monitor just the cross-student parameters ($\boldsymbol{\alpha}_0$, α_N, $\boldsymbol{\mu}_0$, $\boldsymbol{\beta}_0$, $\log(\boldsymbol{\tau}_0)$, and $\boldsymbol{\gamma}_0$) for a much longer time to check pseudo-convergence, and then a short additional run can be used to draw inferences about the student specific parameters, $\boldsymbol{\pi}_i$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\tau}_i$ (for a considerable memory savings).

8. *Data point labeling.* In mixture models, it is sometimes of interest to identify which mixture component each observation $Y_{i,j}$ comes from (Section 4.5).

9. *Calculate model fit index.* If the goal is to compare models for several different values of K, then a measure of model fit such as DIC or WAIC should be calculated (Section 5).

4.1 Component Identification

The multiple modes in the posterior distribution for mixture models present a special problem for MCMC. In particular, it is possible for a Markov chain to get stuck in a mode corresponding to a single component labeling and never mix to the other component labelings. (This especially problematic when coupled with the common practice of starting several parallel Markov chains.) Früthwirth-Schnatter (2001) describes this problem in detail.

One solution is to constrain the parameter space to follow some canonical ordering (e.g., $\mu_{i,1} \leq \mu_{i,2} \leq \cdots \leq \mu i, K$). Stan allows parameter vectors to be specified as ordered, that is restricted to be in increasing order. This seems tailor-made for the identification issue. If an order based on $\boldsymbol{\mu}_0$ is desired, the declaration:

```
ordered[K] mu0;
```

enforces the ordering constraint in the MCMC sampler. JAGS contains a `sort` function which can achieve a similar effect. Früthwirth-Schnatter (2001) recommends against this solution because the resulting Markov chains often mix slowly. Some experimentation with the ordered restriction in Stan confirmed this finding; the MCMC runs took longer and did not reach pseudo-convergence as quickly when the ordered restriction was not used.

Früthwirth-Schnatter (2001) instead recommends letting the Markov chains mix across all of the possible modes, and then sorting the values according to the desired identification constraints post hoc. JAGS provides special support for this approach, offering a special distribution `dnormmix` for mixtures of normals. This uses a specially chosen proposal distribution to encourage jumping between the multiple modes in the posterior. The current version of Stan does not provide a similar feature.

One result of this procedure is that the MCMC sample will have a variety of orderings of the components; each draw could potentially have a different labeling. For example, the MCMC sample for the parameter $\mu_{i,1}$ will in fact be a mix of samples from $\mu_{i,1}, \ldots, \mu_{i,K}$. The average of this sample is not likely to be a good estimate of $\mu_{i,1}$. Similar problems arise when looking at pseudo-convergence statistics which are related to the variances of the parameters.

To fix this problem, the component labels need to be permuted separately for each cycle of the MCMC sample. With a one-level mixture model, it is possible to identify the components by sorting on one of the student-level parameters, $\pi_{i,k}$, $\mu_{i,k}$ or $\tau_{i,k}$. For the hierarchical mixture model, one of the cross-student parameters, $\alpha_{0,k}$, $\mu_{0,k}$, or $\tau_{0,k}$, should be used. Note that in the hierarchical models some of the student-level parameters might not obey the constraints. In the case of the pause time analysis, this is acceptable as some students may behave differently from most of their peers (the ultimate goal of the mixture modeling is to identify students who may benefit from different approaches to instruction). Choosing an identification rule involves picking which parameter should be used for identification at Level 2 (cross-student) and using the corresponding parameter is used for student-level parameter identification.

Component identification is straightforward. Let $\boldsymbol{\omega}$ be the parameter chosen for model identification. Figure 3 describes the algorithm.

```
for each Markov chain $c$: do
  for each MCMC sample $r$ in Chain $c$: do
    Find a permutation of indexes $k'_1, \ldots, k'_K$ so that $\omega_{c,r,k'_1} \leq \cdots \leq \omega_{c,r,k'_1}$.
    for $\boldsymbol{\xi}$ in the Level 2 parameters $\{\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0, \boldsymbol{\beta}_0, \boldsymbol{\tau}_0, \boldsymbol{\gamma}_0\}$: do
      Replace $\boldsymbol{\xi}_{c,r}$ with $(\xi_{c,r,k'_1}, \ldots, \xi_{c,r,k'_K})$.
    end for{Level 2 parameter}
    if inferences about Level 1 parameters are desired then
      for $\boldsymbol{\xi}$ in the Level 1 parameters $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\tau},\}$ do
        for each Level 2 unit (student), $i$ do
          Replace $\boldsymbol{\xi}_{c,r,i}$ with $(\xi_{c,r,i,k'_1}, \ldots, \xi_{c,r,i,k'_K})$.
        end for{Level 1 unit}
      end for{Level 1 parameter}
    end if
  end for{Cycle}
end for{Markov Chain}
```

Figure 3: Component Identification Algorithm

4.2 Starting Values

Although the Markov chain should eventually reach its stationary distribution no matter where it starts, starting places that are closer to the center of the distribution are better in the sense that the chain should reach convergence more quickly. Gelman et al. (2013) recommend starting at the maximum likelihood estimate when that is available. Fortunately, off-the shelf software is available for finding the maximum likelihood estimate of the individual student mixture models. These estimates can be combined to find starting values for the cross-student parameters. The initial values can be set by the following steps:

1. *Fit a separate mixture model for students using maximum likelihood.* The package `mixtools` is slightly better for this purpose than `FlexMix` as it will try multiple times to get a solution with the requested number of components. If the EM algorithm does not converge in several tries, set the values of the parameters for that student to `NA` and move on.

2. *Identify Components.* Sort the student-level parameters, $\boldsymbol{\pi}_i$, $\boldsymbol{\mu}_i$, $\boldsymbol{\sigma}_i$ and $\boldsymbol{\tau}_i$ according to the desired criteria (see Section 4.1.

3. *Calculate the Level 2 initial values.* Most of the cross-student parameters are either means or variances of the student-level parameters. The initial value for $\boldsymbol{\alpha}_0$ is the mean of $\boldsymbol{\pi}_i$ across the students i (ignoring `NA`s). The initial values for $\boldsymbol{\mu}_0$ and $\boldsymbol{\beta}_0$ are the mean and standard deviation of $\boldsymbol{\mu}_i$. The initial values for $\log(\boldsymbol{\tau}_0)$ and $\boldsymbol{\gamma}_0$ are the mean and standard deviation of $\log(\boldsymbol{\tau}_i)$.

4. *Impute starting values for maximum likelihood estimates that did not converge.* These are the `NA`s

from Step 1. For each student i that did not converge in Step 1, set $\boldsymbol{\pi}_i = \boldsymbol{\alpha}_0$, $\boldsymbol{\mu}_i = \boldsymbol{\mu}_0$ and $\boldsymbol{\tau}_i = \boldsymbol{\tau}_0$.

5. *Compute initial values for $\boldsymbol{\theta}_i$ and $\boldsymbol{\eta}_i$.* These can be computed by solving Equations 7 and 9 for $\boldsymbol{\theta}_i$ and $\boldsymbol{\eta}_i$.

6. *Set the initial value of $\alpha_N = I$.* Only one parameter was not given an initial value in the previous steps, that is the effective sample size for $\boldsymbol{\alpha}$. Set this equal to the number of Level 2 units, I. The initial values of $\boldsymbol{\alpha}$ can now be calculated as well.

This produces a set of initial values for a single chain. Common practice for checking for pseudo-convergence involves running multiple chains and seeing if they arrive the same stationary distribution. Different starting values for the second and subsequent chains can be found by sampling some fraction λ of the Level 1 units from each Level 2 unit. When J_i is large, $\lambda = .5$ seems to work well. When J_i is small, $\lambda = .8$ seems to work better. An alternative would be to take a bootstrap sample (a new sample of size J_i drawn with replacement).

4.3 Automated Convergence Criteria

Gelman and Shirley (2011) describe recommended practice for assessing pseudo-convergence of a Markov chain. The most common technique is to run several Markov chains and look at the ratio of within-chain variance to between-chain variance. This ratio is called $\widehat{R}$, and it comes in both a univariate (one parameter at a time) and a multivariate version. It is natural to look at parameters with the same name together (e.g., $\boldsymbol{\mu}_0$, $\boldsymbol{\beta}_0$, $\boldsymbol{\tau}_0$, and $\boldsymbol{\gamma}_0$). Using the multivariate version of $\widehat{R}$ with $\boldsymbol{\alpha}_0$ and $\boldsymbol{\pi}_i$ requires some care because calculating the multivariate $\widehat{R}$ involves inverting a matrix that does not have full rank when the parameters are restricted to a simplex. The work-around is to only look at the first $K-1$ values of these parameters.

The other commonly used test for pseudo-convergence is to look at a trace plot for each parameter: a time series plot of the sampled parameter value against the MCMC cycle number. Multiple chains can be plotted on top of each other using different colors. If the chain is converged and the sample size is adequate, the traceplot should look like white noise. If this is not the case, a bigger MCMC sample is needed. While it is not particularly useful for an automated test of pseudo-convergence, the traceplot is very useful for diagnosing what is happening when the chains are not converging. Some sample traceplots are given below.

Even if the chains have reached pseudo-convergence, the size of the Monte Carlo error could still be an issue. Both `rstan` and `coda` compute an *effective sample size* for the Monte Carlo sample. This is the size of a simple random sample from the posterior distribution that would have the same Monte Carlo error as the obtained dependent sample. This is different for different parameters. If the effective sample size is too small, then additional samples are needed.

Note that there are $K(3I + 5)$ parameters whose convergence needs to be monitored. It is difficult to achieve pseudo-convergence for all of these parameters, and exhausting to check them all. A reasonable compromise seems to be to monitor only the $5K$ cross-student parameters, $\boldsymbol{\alpha}$, $\boldsymbol{\mu}_0$, $\boldsymbol{\beta}_0$, $\log(\boldsymbol{\tau}_0)$ and $\boldsymbol{\gamma}_0$. JAGS makes this process easier by allowing the analyst to pick which parameters to monitor. Using JAGS, only the cross-student parameters can be monitored during the first MCMC run, and then a second shorter sample of student-level parameters can be be obtained through an additional run. The `rstan` package always monitors all parameters, including the $\boldsymbol{\theta}_i$ and η_i parameters which are not of particular interest.

When I and J_i are large, a run can take several hours to several days. As the end of a run might occur during the middle of the night, it is useful to have a automated test of convergence. The rule I have been using is to check to see if all $\widehat{R}$ are less than a certain maximum (by default 1.1) and if all effective sample sizes are greater than a certain minimum (by default 100) for all cross-student parameters. If these criteria are not met, new chains of twice the length of the old chain can be run. The traceplots are saved to a file for later examination, as are some other statistics such as the mean value of each chain. (These traceplots and outputs are available at the web site mentioned above.) It is generally not worthwhile to restart the chain for a third time. If pseudo-convergence has not been achieved after the second longer run, usually a better solution is to reparameterize the model.

It is easier to extend the MCMC run using JAGS than using Stan. In JAGS, calling `update` again samples additional points from the Markov chain, starting at the previous values, extending the chains. The current version of Stan (2.2.0)[1] saves the compiled C++ code, but not the warm-up parameter values. So the second run is a new run, starting over from the warm-up phase. In this run, both the warm-up phase and the sampling phase should be extended, because a longer

[1]The Stan development team have stated that the ability to save and reuse the warm-up parameters are a high priority for future version of Stan. Version 2.3 of Stan was released between the time of first writing and publication of the paper, but the release notes do not indicate that the restart issue was addressed.

warm-up may produce a lower autocorrelation. In either case, the data from both runs can be pooled for inferences.

4.4 A simple test

To test the scripts and the viability of the model, I created some artificial data consistent with the model in Section 2. To do this, I took one of the data sets use by Li (2013) and ran the initial parameter algorithm described in Section 4.2 for $K = 2$, 3 and 4. I took the cross-student parameters from this exercise, and generated random parameters and data sets for 10 students. This produced three data sets (for $K = 2$, 3 and 4) which are consistent with the model and reasonably similar to the observed data. All three data sets and the sample parameters are available on the web site, `http://pluto.coe.fsu.edu/mcmc-hierMM/`.

For each data set, I fit three different hierarchical mixture models ($K' = 2$, 3 and 4, where K' is the value used for estimation) using three different variations of the algorithm: JAGS (RWM), Stan (HMC) with the cross-student means constrained to be increasing, and Stan (HMC) with the means unconstrained. For the JAGS and Stan unconstrained run the chains were sorted (Section 4.1) on the basis of the cross-students means ($\boldsymbol{\mu}_0$) before the convergence tests were run. In each case, the initial run of 3 chains with 5,000 observations was followed by a second run of 3 chains with 10,000 observations when the first one did not converge. All of the results are tabulated at the web site, including links to the complete output files and all traceplots.

Unsurprisingly, the higher the value of K' (number of components in the estimated model), the longer the run took. The runs for $K' = 2$ to between 10–15 minutes in JAGS and from 30–90 minutes in Stan, while the runs for $K' = 4$ too just less than an hour in JAGS and between 2–3 hours in Stan. The time difference between JAGS and Stan is due to the difference in the algorithms. HMC uses a more complex proposal distribution than RWM, so naturally each cycle takes longer. The goal of the HMC is to trade the longer run times for a more efficient (lower autocorrelation) sample, so that the total run time is minimized. The constrained version of Stan seemed to take longer than the unconstrained.

In all 27 runs, chain lengths of 15,000 cycles were not sufficient to reach pseudo-convergence. The maximum (across all cross-student parameters) value for $\widehat{R}$ varied between 1.3 and 2.2, depending on the run. It seemed to be slightly worse for cases in which K (number of components for data generation) was even and K' (estimated number of components) was odd or vice versa. The values of $\widehat{R}$ for the constrained Stan model were substantially worse, basically confirming the findings of Früthwirth-Schnatter (2001).

The effective sample sizes were much better for Stan and HMC. For the $K' = 2$ case, the smallest effective sample size ranged from 17–142, while for the unconstrained Stan model it ranged from 805–3084. Thus, roughly 3 times the CPU time is producing a 5-fold decrease in the Monte Carlo error.

The following graphs compare the JAGS and Stan (Unconstrained model) outputs for four selected cross-student parameters. The output is from `coda` and provides a traceplot on the left and a density plot for the corresponding distribution on the right. Recall that the principle difference between JAGS and Stan is the proposal distribution: JAGS uses a random walk proposal with a special distribution for the mixing parameters and Stan uses the Hamiltonian proposal. Also, note that for Stan the complete run of 15,000 samples is actually made up of a sort run of length 5,000 and a longer run of length 10,000; hence there is often a discontinuity at iteration 5,000.

Although not strictly speaking a parameter, the deviance (twice the negative log likelihood) can easily be monitored in JAGS. Stan does not offer a deviance monitor, but instead monitors the log posterior, which is similar. Both give an impression of how well the model fits the data. Figure 4 shows the monitors for the deviance or log posterior. This traceplot is nearly ideal white noise, indicating good convergence for this value. The value of $\widehat{R}$ is less than the heuristic threshold of 1.1 for both chains, and the effective sample size is about 1/6 of the 45,000 total Monte Carlo observations.

Figure 5 shows the grand mean of the log pause times for the first component. The JAGS output (upper row) shows a classic slow mixing pattern: the chains are crossing but moving slowly across the support of the parameter space. Thus, for JAGS even though the chains have nearly reached pseudo-convergence ($\widehat{R}$ is just slightly greater than 1.1), the effective sample size is only 142 observations. A longer chain might be needed for a good estimate of this parameter. The Stan output (bottom row) looks much better, and the effective sample size is a respectable 3,680.

The Markov chains for $\alpha_{0,1}$ (the proportion of pause times in the first component) have not yet reached pseudo convergence, but they are close (a longer run might get them there). Note the black chain often ranges above the values in the red and green chains in the JAGS run (upper row). This is an indication that the chains may be stuck in different modes; the

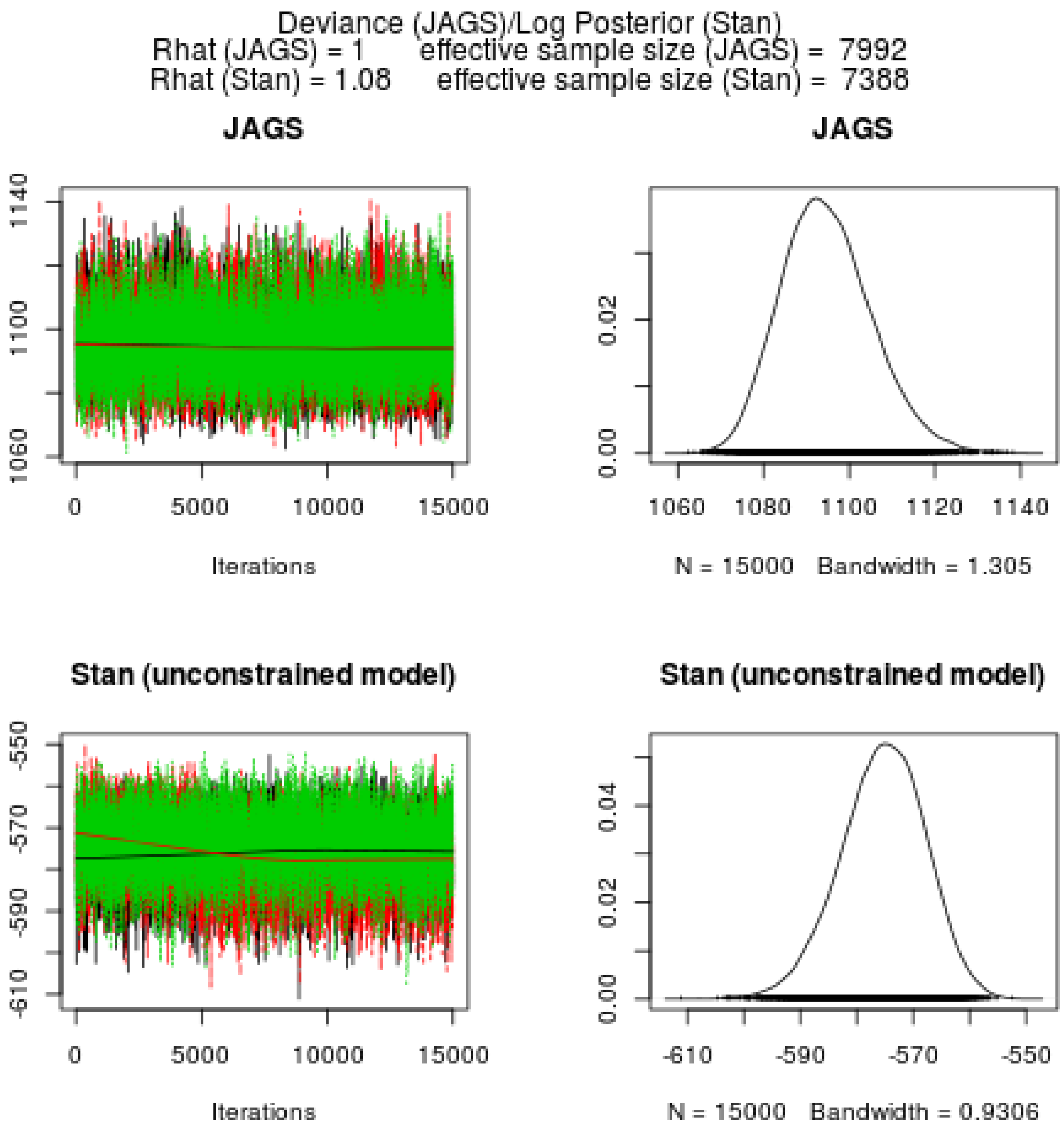


Figure 4: Deviance (JAGS) and Log Posterior (Stan) plots
Note: To reduce the file size of this paper, this is a bitmap picture of the traceplot. The original pdf version is available at `http://pluto.coe.fsu.edu/mcmc-hierMM/DeviancePlots.pdf`.

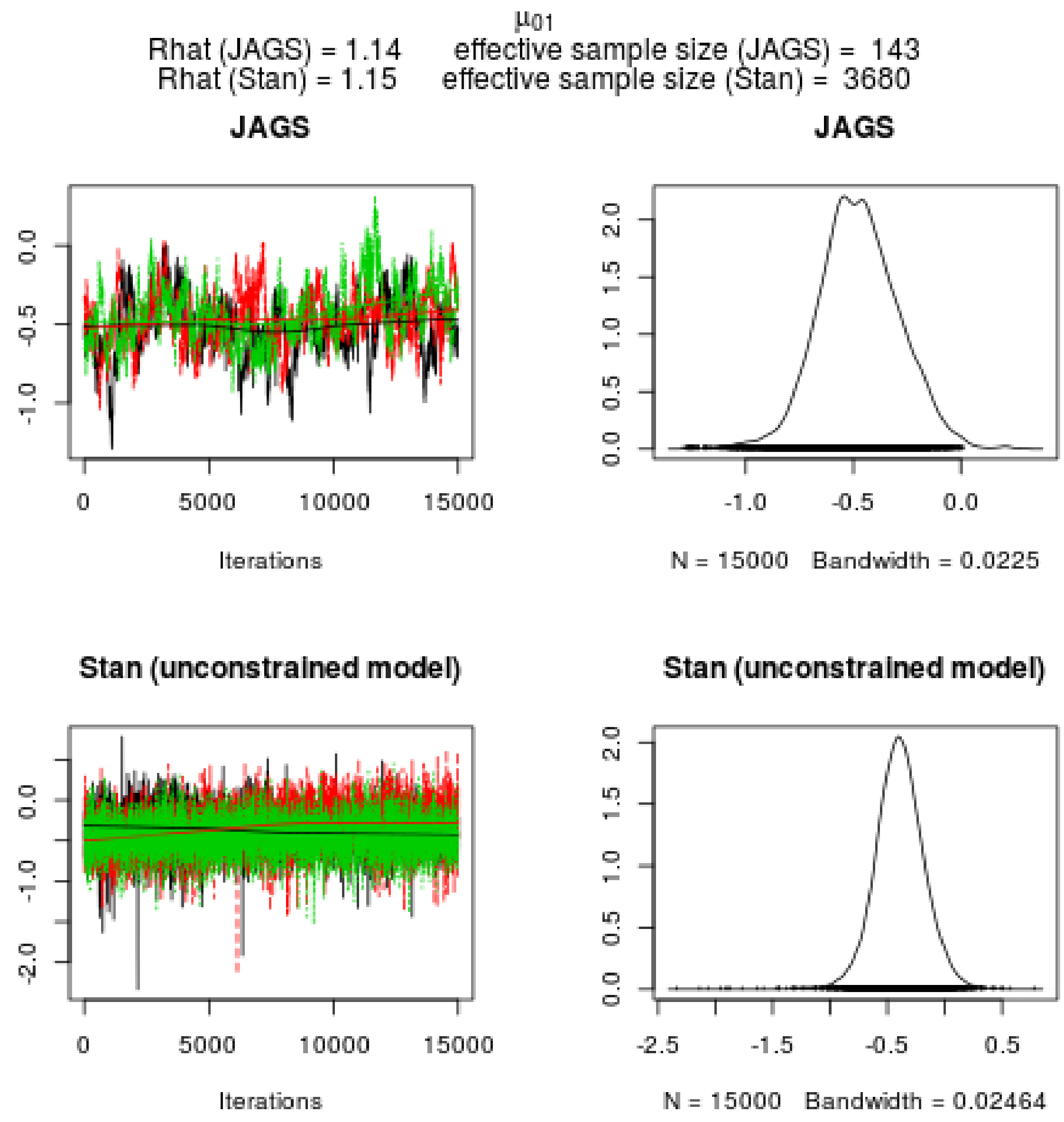


Figure 5: Traceplots and Density plots for $\mu_{0,1}$
Note: To reduce the file size of this paper, this is a bitmap picture of the traceplot. The original pdf version is available at `http://pluto.coe.fsu.edu/mcmc-hierMM/mu0Plots.pdf`.

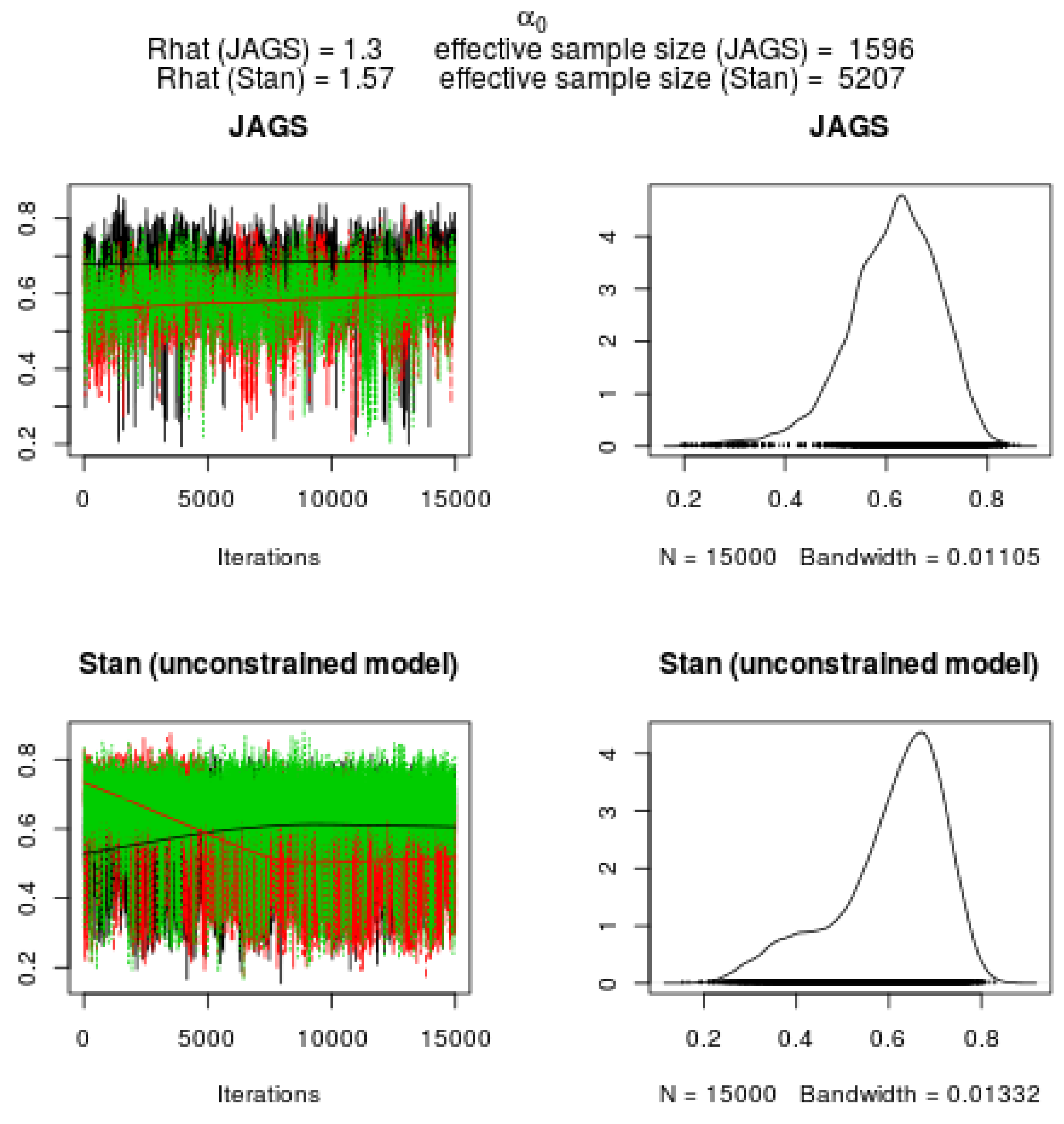


Figure 6: Traceplots and Density plots for $\alpha_{0,1}$

Note: To reduce the file size of this paper, this is a bitmap picture of the traceplot. The original pdf version is available at `http://pluto.coe.fsu.edu/mcmc-hierMM/alpha0Plots.pdf`.

shoulder on the density plot also points towards multiple modes. The plot for Stan looks even worse, after iteration 5,000 (i.e., after the chain was restarted), the red chain has moved into the lower end of the support of the distribution. This gives a large shoulder in the corresponding density plot.

The chains for the variance of the log precisions for the first component, $\gamma_{0,1}$ (Figure 7), are far from pseudo-convergence with $\widehat{R} = 1.73$. In the JAGS chains (top row) the black chain seems to prefer much smaller values for this parameter. In the Stan output, the green chain seems to be restricted to the smaller values in both the first and second run. However, between the first and second run the black and red chains swap places. Clearly this is the weakest spot in the model, and perhaps a reparameterization here would help.

Looking at Figures 6 and 7 together, a pattern emerges. There seems to be (at least) two distinct modes. One mode (black chain JAGS, green chain in Stan) has higher value for $\alpha_{0,1}$ and a lower value for $\gamma_{0,1}$, and the other one has the reverse pattern. This is an advantage of the MCMC approach over an EM approach. In particular, if the EM algorithm was only run once from a single starting point, the existence of the other mode might never be discovered.

4.5 Data Point Labeling

For some applications, the analysis can stop after estimating the parameters for the students units, $\boldsymbol{\pi}_i$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\tau}_i$ (or $\boldsymbol{\sigma}_i$ as desired). In other applications it is interesting to assign the individual pauses to components, that is, to assign values to $Z_{i,j}$.

When $\boldsymbol{\pi}_i$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are known, it is simple to calculate $p_{i,j,k} = \Pr(Z_{i,j} = k)$. Let $\boldsymbol{\pi}_i^{(c,r)}$, $\boldsymbol{\mu}_i^{(c,r)}$ and $\boldsymbol{\sigma}_i^{(c,r)}$ be the draws of the Level 1 parameters from Cycle r of Chain c (after the data have been sorted to identify the components). Now define,

$$q_{i,j,k}^{(c,r)} = \pi_{i,k}^{(c,r)} \phi\left(\frac{Y_{i,j} - \mu_{i,k}^{(c,r)}}{\sigma_{i,k}^{(c,r)}}\right), \qquad (17)$$

and set $Q_{i,j}^{(c,r)} = \sum_{k=1}^{K} q_{i,j,k}^{(c,r)}$. The distribution for $Z_{i,j}$ for MCMC draw (c,r) is then $p_{i,j,k}^{(c,r)} = q_{i,j,k}^{(c,r)}/Q_{i,j}^{(c,r)}$. The MCMC estimate for $p_{i,j,k}$ is just the average over all the MCMC draws of $p_{i,j,k}^{(c,r)}$, that is $\frac{1}{CR}\sum_{c=1}^{C}\sum_{r=1}^{R} p_{i,j,k}^{(c,r)}$.

If desired, $Z_{i,j}$ can be estimated as the maximum over k of $p_{i,j,k}$, but for most purposes simply looking at the probability distribution $\mathbf{p}_{i,j}$ provides an adequate, if not better summary of the available information about the component from which $Y_{i,j}$ was drawn.

4.6 Additional Level 2 Units (students)

In the pause time application, the goal is to be able to estimate fairly quickly the student-level parameters for a new student working on the same essay. As the MCMC run is time consuming, a fast method for analyzing data from new student would be useful.

One approach would be to simply treat Equations 2, 3 and 4 as prior distributions, plugging in the point estimates for the cross-student parameters as hyperparameters, and find the posterior mode of the observations from the new data. There are two problems with this approach. First, the this method ignores any remaining uncertainty about the cross-student parameters. Second, the existing mixture fitting software packages (`FlexMix` and `mixtools`) do not provide any way to input the prior information.

A better approach is to do an additional MCMC run, using the posterior distributions for the cross-student parameters from the previous run as priors from the previous run (Mislevy et al., 1999). If the hyperparameters for the cross-student priors are passed to the MCMC sampler as data, then the same JAGS or Stan model can be reused for additional student pause records. Note that in addition to the MCMC code Stan provides a function that will find the maximum of the posterior. Even if MCMC is used at this step, the number of Level 2 units, I, will be smaller and the chains will reach pseudo-convergence more quickly.

If the number of students, I, is large in the original sample (as is true in the original data set), then a sample of students can be used in the initial model calibration, and student-level parameters for the others can be estimated later using this method. In particular, there seems to be a paradox estimating models using large data sets with MCMC. The larger the data, the slower the run. This is not just because each data point needs to be visited within each cycle of each Markov chain. It appears, especially with hierarchical models, that more Level 2 units cause the chain to have higher autocorrelation, meaning larger runs are needed to achieve acceptable Monte Carlo error. For the student keystroke logs, the "initial data" used for calibration was a sample of 100 essays from the complete collection. There is a obvious trade-off between working with smaller data sets and being able to estimate rare events: $I = 100$ seemed like a good compromise.

5 Model Selection

A big question in hierarchical mixture modeling is "What is the optimal number of components, K?" Although there is a way to add a prior distribution

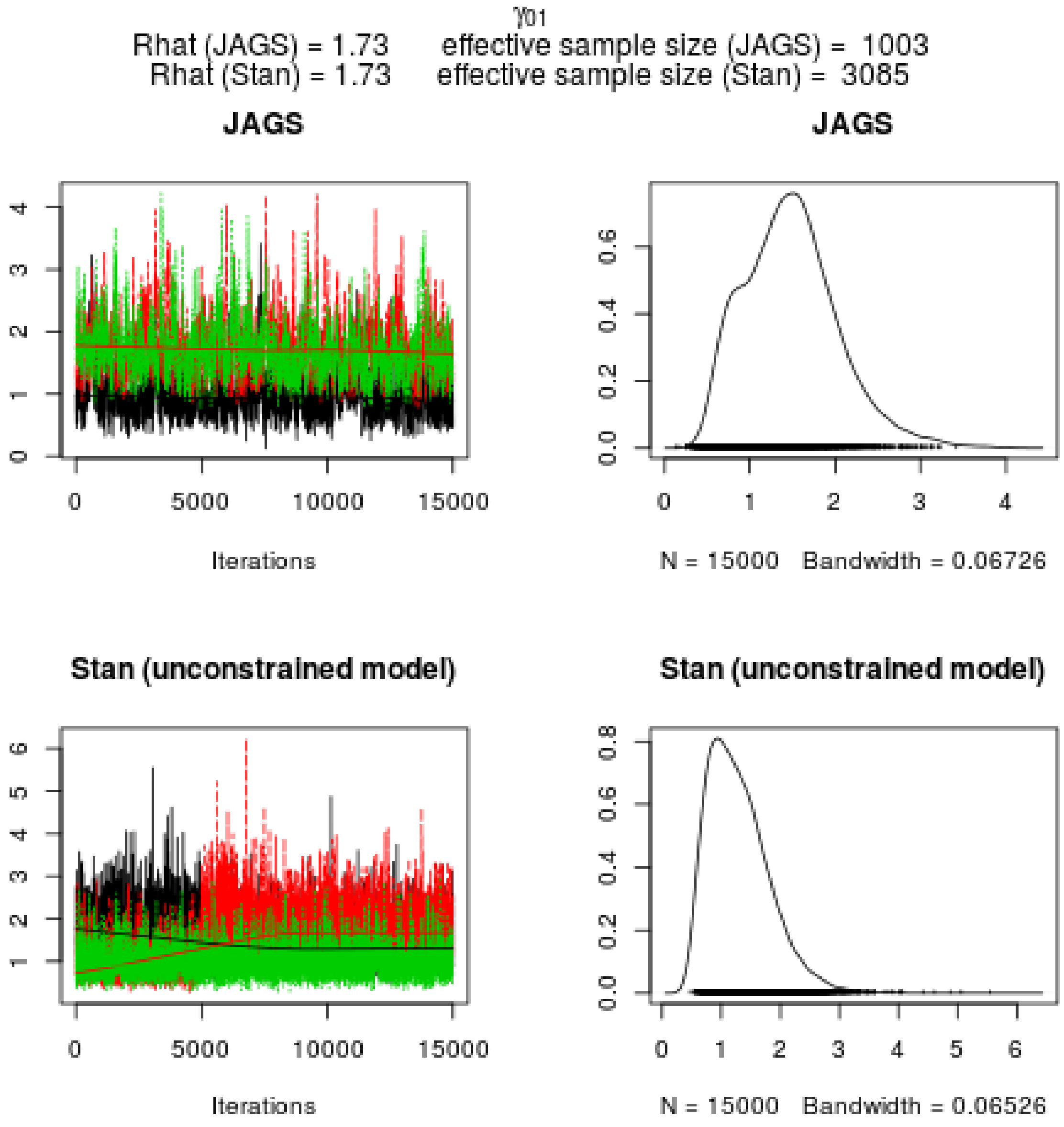


Figure 7: Traceplots and Density plots for $\gamma_{0,1}$
Note: To reduce the file size of this paper, this is a bitmap picture of the traceplot. The original pdf version is available at `http://pluto.coe.fsu.edu/mcmc-hierMM/gamma0Plots.pdf`.

over K and learn this as part of the MCMC process, it is quite tricky and the number of parameters varies for different values of K. (Geyer, 2011, describes the Metropolis-Hastings-Green algorithm which is required for variable dimension parameter spaces). In practice, analysts often believe that the optimal value of K is small (less than 5), so the easiest way to determine a value of K is to fit separate models for $K = 2, 3, 4, \ldots$ and compare the fit of the models.

Note that increasing K should always improve the fit of the model, even if the extra component is just used to trivially fit one additional data point. Consequently, analysts use penalized measures of model fit, such as AIC to chose among the models.

Chapter 7 of Gelman et al. (2013) gives a good overview of the various information criteria, so the current section will provide only brief definitions. Recall Equation 5 that described the log likelihood of the complete data $\mathbf{Y}$ as a function of the complete parameter $\mathbf{\Omega}$. By convention, the *deviance* is twice the negative log likelihood: $D(\mathbf{\Omega}) = -2\mathcal{L}(\mathbf{Y}|\mathbf{\Omega})$. The lower the deviance, the better the fit of the data to the model, evaluated at that parameter. As students are independent given the cross-student parameters, and pause times are independent given the student-level parameters, $D(\widetilde{\mathbf{\Omega}})$, for some value of the parameters $\widetilde{\mathbf{\Omega}}$ can be written as a sum:

$$D(\widetilde{\mathbf{\Omega}}) = \sum_{i=1}^{I}\sum_{j=1}^{J_i} \log Q_{i,j}(\widetilde{\mathbf{\Omega}}) \ , \tag{18}$$

where $Q_{i,j}(\widetilde{\mathbf{\Omega}})$ is the likelihood of data point $Y_{i,j}$, that is:

$$Q^{i,j}(\widetilde{\mathbf{\Omega}}) = \sum_{k=1}^{k} \widetilde{\pi_{i,k}}\phi(\frac{Y_{i,j} - \widetilde{\mu_{i,k}}}{\widetilde{\sigma_{i,k}}}). \tag{19}$$

Note that the cross-student parameters drop out of these calculations.

The Akaike information criterion (AIC) takes the deviance as a measure of model fit and penalizes it for the number of parameters in the model. Let $\widehat{\mathbf{\Omega}}$ be the maximum likelihood estimate of the parameter. Then AIC is defined as:

$$\text{AIC} = D(\widehat{\mathbf{\Omega}}) + 2 * m \tag{20}$$

where $m = K(2IK + 4) + (K - 1)(I + 1) + 1$ is the number of free parameters in the model. Note that the MCMC run does not provide a maximum likelihood estimate. A pseudo-AIC can be computed by substituting $\overline{\mathbf{\Omega}}$, the posterior mean of the parameters $\mathbf{\Omega}$. Note that $\overline{\mathbf{\Omega}}$ must be calculated after the parameters in each MCMC cycle are permuted to identify the components.

Gelman et al. (2013) advocate using a new measure of model selection called WAIC. WAIC makes two substitutions in the definition of AIC above. First, it uses $\overline{D(\mathbf{\Omega})}$ in place of $D(\overline{\mathbf{\Omega}})$. Second, it uses a new way of calculating the dimensionality of the model. There are two possibilities:

$$p_{\text{WAIC}_1} = 2\sum_{i=1}^{I}\sum_{j=1}^{J_i} (\log E[Q_{i,j}(\mathbf{\Omega})] - E[\log Qi, j(\mathbf{\Omega})]) \ , \tag{21}$$

$$p_{\text{WAIC}_2} = 2\sum_{i=1}^{I}\sum_{j=1}^{J_i} \text{Var}(Q_{i,j}(\mathbf{\Omega})) \ . \tag{22}$$

The expectation and variance are theoretically defined over the posterior distribution and are approximated using the MCMC sample. The final expression for WAIC is:

$$\text{WAIC} = \overline{D(\mathbf{\Omega})} + 2 * m_{\text{WAIC}} \ . \tag{23}$$

For all of the model fit statistics discussed here: AIC, WAIC$_1$ and WAIC$_2$, the model with the lowest value is the best. One way to chose the best value of K is to calculate all of these statistics for multiple values of K and choose the value of K which has the lowest value for all 5 statistics. Hopefully, all of the statistics will point at the same model. If there is not true, that is often evidence that the fit of two different models is too close to call.

In the 27 runs using the data from known values of K, simply using the lowest WAIC value was not sufficient to recover the model. As the Markov chains have not yet reached pseudo-convergence, the WAIC values may not be correct, but there was fairly close agreement on both WAIC$_1$ and WAIC$_2$ for both the JAGS and Stan (unconstrained) models. However, the values of both WAIC$_1$ and WAIC$_2$ were also fairly close for all values of K' (number of estimated components). For example when $K = 2$ the WAIC$_1$ value was 1132, 1132 and 1135 for $K' = 2$, 3 and 4 respectively when estimated with JAGS and 1132, 1131, and 1132 when estimated with Stan. The results for WAIC$_2$ were similar. It appears as if the common practice of just picking the best value of WAIC to determine K is not sufficient. In particular, it may be possible to approximate the data quite well with a model which has an incorrect value of K.

6 Conclusions

The procedure described above is realized as code in R, Stan and JAGS and available for free at `http://pluto.coe.fsu.edu/mcmc-hierMM/`. Also available, are the simulated data for trying out the models.

These scripts contain the functions necessary to automatically set starting parameters for these problems, identify the components, and to calculate WAIC model fit statistics.

The scripts seem to work fairly well for a small number of students units ($5 \leq I \leq 10$). With this sample size, although JAGS has the faster run time, Stan has a larger effective sample size. Hamiltonian Monte Carlo. With large sample sizes $I = 100$, $J_i \approx 100$ even the HMC has high autocorrelation and both JAGS and Stan are extremely slow. Here the ability of JAGS to allow the user to selectively monitor parameters and to extend the chains without starting over allows for better memory management.

The code supplied at the web site solves two key technical problems: the post-run sorting of the mixture components (Früthwirth-Schnatter, 2001) and the calculation of the WAIC model-fit indexes. Hopefully, this code will be useful for somebody working on a similar application.

The model proposed in this paper has two problems, even with data simulated according to the model. The first is the slow mixing. This appears to be a problem with multiple modes, and indicates some possible non-identifiability in the model specifications. Thus, further work is needed on the form of the model. The second problem is that the WAIC test does not appear to be sensitive enough to recover the number of components in the model. As the original purpose of moving to the hierarchical mixture model was to discover if there were rare components that could not be detected in the single-level hierarchical model, the current model does not appear to be a good approach for that purpose.

In particular, returning to the original application of fitting mixture models to student pauses while writing, the hierarchical part of the model seems to be creating as many problems as it solves. It is not clearly better than fitting the non-hierarchical mixture model individually to each student. Another problem it has for this application is the assumption that each pause is independent. This does not correspond with what is known about the writing process: typically writers spend long bursts of activities simply doing transcription (i.e., typing) and only rarely pause for higher level processing (e.g., spelling, grammar, word choice, planning, etc). In particular, rather than getting this model to work for this application, it may be better to look at hidden Markov models for individual student writings.[2]

The slow mixing of the hierarchical mixture model for large data sets indicates that there may be additional transformation of this model that are necessary to make it work properly. Hopefully, the publication of the source code will allow other to explore this model more fully.

[2]Gary Feng and Paul Deane, private communication.

References

Almond, R. G., Deane, P., Quinlan, T., Wagner, M., & Sydorenko, T. (2012). *A preliminary analysis of keystroke log data from a timed writing task* (Research Report No. RR-12-23). Educational Testing Service. Retrieved from `http://www.ets.org/research/policy_research_reports/publications/report/2012/jgdg`

Benaglia, T., Chauveau, D., Hunter, D. R., & Young, D. (2009). mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, *32*(6), 1–29. Retrieved from `http://www.jstatsoft.org/v32/i06/`

Brooks, S., Gelman, A., Jones, G. L., & Meng, X. (Eds.). (2011). *Handbook of markov chain monte carlo*. CRC Press.

Deane, P. (2012). Rethinking K-12 writing assessment. In N. Elliot & L. Perelman (Eds.), *Writing assessment in the 21st century. essays in honor of Edward M. white* (pp. 87–100). Hampton Press.

Früthwirth-Schnatter, S. (2001). Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. *Journal of the American Statistical Association*, *96*(453), 194–209.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis* (3rd ed.). Chapman and Hall. (The third edition is revised and expanded and has material that the earlier editions lack.)

Gelman, A., & Shirley, K. (2011). Inference from simulations and monitoring convergence. In S. Brooks, A. Gelman, G. L. Jones, & X. Meng (Eds.), *Handbook of markov chain monte carlo* (pp. 163–174). CRC Press.

Geyer, C. J. (2011). Introduction to Markov chain Monte Carlo. In S. Brooks, A. Gelman, G. L. Jones, & X. Meng (Eds.), *Handbook of markov chain monte carlo* (pp. 3–48). CRC Press.

Gruen, B., & Leisch, F. (2008). FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, *28*(4), 1–35. Retrieved from `http://www.jstatsoft.org/v28/i04/`

Li, T. (2013). *A Bayesian hierarchical mixture approach to model timing data with application to* *writing assessment* (Unpublished master's thesis). Florida State University.

McLachlan, G., & Krishnan, T. (2008). *The EM algorithm and extensions* (2nd ed.). Wiley.

McLachlan, G., & Peel, D. (2000). *Finite mixture models*. Wiley.

Mislevy, R. J., Almond, R. G., Yan, D., & Steinberg, L. S. (1999). Bayes nets in educational assessment: Where the numbers come from. In K. B. Laskey & H. Prade (Eds.), *Uncertainty in artificial intelligence '99* (p. 437-446). Morgan-Kaufmann.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In MCMCHandbook (Ed.), (pp. 113–162).

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

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News*, *6*(1), 7–11. Retrieved from `http://CRAN.R-project.org/doc/Rnews/`

R Core Team. (2014). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/`

Stan Development Team. (2013). Stan: A C++ library for probability and sampling (Version 2.2.0 ed.) [Computer software manual]. Retrieved from `http://mc-stan.org/`

Thomas, A., O'Hara, B., Ligges, U., & Sturtz, S. (2006). Making BUGS open. *R News*, *6*, 12-17. Retrieved from `http://www.rni.helsinki.fi/~boh/publications/Rnews2006-1.pdf`

Thomas, A., Spiegelhalter, D. J., & Gilks, W. R. (1992). BUGS: A program to perform Bayesian inference using Gibbs sampling. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian statistics 4*. (pp. 837–842). Clarendon Press.

Acknowledgments

Tingxuan Li wrote the first version of much of the R code as part of her Master's thesis. Paul Deane and Gary Feng of Educational Testing Service have been collaborators on the essay keystroke log analysis work which is the inspiration for the present research. Various members of the JAGS and Stan users groups and development teams have been helpful in answering emails and postings related to these models, and Kathy Laskey and several anonymous reviewers have given me useful feedback (to which I should have paid closer attention) on the draft of this paper.

Multi-Process Models – An Application for the Construction of Financial Factor Models

Kevin R. Keane and **Jason J. Corso**
Computer Science and Engineering
University at Buffalo, The State University of New York
Buffalo, NY 14260

Abstract

We present an unsupervised, comprehensive methodology for the construction of financial risk models. We offer qualitative comments on incremental functionality and quantitative measures of superior performance of component and mixture dynamic linear models relative to alternative models. We apply our methodology to a high dimensional stream of daily closing prices for approximately 7,000 US traded stocks, ADRs, and ETFs for the most recent 10 years. Our methodology automatically extracts an evolving set of explanatory time series from the data stream; maintains and updates parameter distributions for component dynamic linear models as the explanatory time series evolve; and, ultimately specifies time-varying asset specific mixture models. Our methodology utilizes a hierarchical Bayesian approach for the specification of component model parameter distributions and for the specification of the mixing weights in the final model. Our approach is insensitive to the exact number of factors, and "effectively" sparse, as irrelevant factors (time series of pure noise) yield posterior parameter distributions with high density around zero. The statistical models obtained serve a variety of purposes, including: outlier detection; portfolio construction; and risk forecasting.

1 INTRODUCTION

We propose a time varying Bayesian statistical model for individual stock returns depicted in Figure 1. Our goal is to accurately model the high dimensional probability distribution underlying stock returns. We demonstrate success by constructing better performing investment portfolios, where the performance goal is to minimize the standard deviation of returns. Investment professionals seek to minimize the variation in investment outcomes because doing so results in higher economic utility for their clients. To illustrate this point, consider which of two pension scenarios with equal expected value is preferred: one that pays 80% salary with probability 1; or, the other that pays 120% salary with probability 1/2 (if things "go well") and pays 40% salary with probability 1/2 (if things "go poorly"). The economic concept of declining marginal utility, expressed mathematically with concave utility functions $\mathrm{U}(\mathrm{E}(x)) > \mathrm{E}(\mathrm{U}(x))$, implies that given investment scenarios with equal expected return, individuals prefer the scenario with lowest variation. Portfolio managers are concerned with generating acceptable returns with minimal risk; and, risk managers monitor the portfolio managers, verifying financial risks remain within authorized limits. Risk models, defining the $n \times n$ asset covariance matrix $\mathbf{\Sigma}$ and precision matrix $\mathbf{\Sigma}^{-1}$, are used by portfolio managers in conjunction with expected return vectors $\boldsymbol{\alpha}$ to *construct* optimal portfolios weights $\mathbf{\Sigma}^{-1}\alpha$; and, are used by risk managers given portfolio weights $\mathbf{w}$ to *forecast* portfolio variance $\mathbf{w}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{w}$.

We describe the construction of a set of Bayesian switching state-space models and qualitatively analyze the on-line behavior of the various component models and mixture models. We focus our discussion on the qualitative aspects then conclude by providing quantitative measures of our model's superior performance relative to competing models. We look at the tradeoff of adaption rate and stability of parameter estimates, evaluating model responsiveness with both synthetic and real data series. We show that dynamic linear models are robust [1] with respect to pure noise explanatory variables, appropriately generating param-

[1] We use the term *robust* to describe a desirable trait where an estimation method is stable in the presence of outlier observations and irrelevant explanatory variables (noise).

eter distributions very concentrated around zero. We comment on the behavior of component models relative to the mixture consensus. We illustrate component and mixture response to outlier observations. In periods with ambiguous posterior model probabilities, we describe the diffusive impact to the mixture distribution; and, we note surprisingly distinct behavior when an outlier component is selected with near certainty. We find unexpectedly similar updating behavior across a range of component models, bringing into question the necessity of more than two components. We inspect the impact of implementing intervention in the estimation process by greatly inflating the variance of a stock's posterior distribution subsequent to a merger event. The intervention is shown to result in extremely rapid convergence to new dynamics, while the same model without intervention is shown to maintain bias for an unacceptably long period. Lastly, we compare our two component mixture model against several alternatives. Analyzing results in Table 1, we show the positive impact of regularization provided by the Bayesian framework relative to PCA, and further improvement of the mixture model as compared to the single process model.

2 BACKGROUND

2.1 RISK MODELS

High dimensional statistical models, central to modern portfolio management, present significant technical challenge in their construction. There is extensive literature on the topic of constructing *factor models* or *risk models* as they are interchangeably known by practitioners. Consider a matrix of observations

$$\mathbf{X} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,t} \\ \vdots & \ddots & \vdots \\ r_{n,1} & \cdots & r_{n,t} \end{bmatrix} \quad , \tag{1}$$

representing log price returns

$$r_{i,j} = \log\left(\frac{p_{i,j}}{p_{i,j-1}}\right) \quad , \tag{2}$$

for assets $i \in 1 \dots n$, trading days $j \in 1 \dots t$, and end of day prices $p_{i,j}$. The general approach to financial risk modeling specifies the covariance of asset returns as a structured matrix. A factor model with p explanatory time series is specified for the $n \times t$ matrix $\mathbf{X}$ with an $n \times p$ matrix of common factor loadings $\mathbf{L}$, a $p \times t$ matrix of common factor returns time series $\mathbf{F}$, and an $n \times t$ matrix of residual error time series $\boldsymbol{\epsilon}$:

$$\mathbf{X} = \mathbf{LF} + \boldsymbol{\epsilon} \quad . \tag{3}$$

An orthogonal factor model (Johnson and Wichern, 1998, Ch. 9) implies diagonal covariance matrices

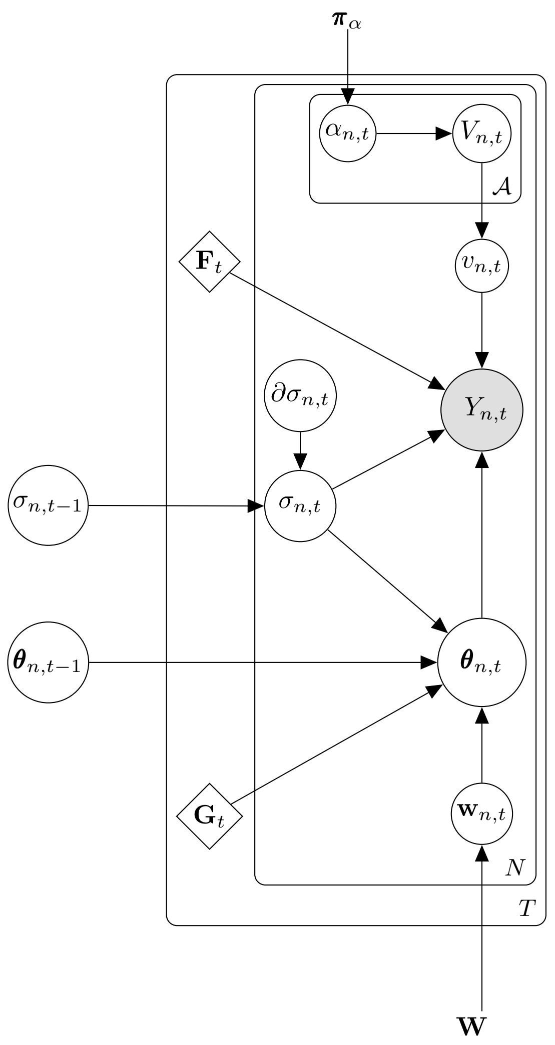


Figure 1: Our final two process Bayesian switching state-space model for log price returns $Y_{n,t} = \log(p_{n,t}/p_{n,t-1})$ for asset n in time period t. We specify the prior probabilities π_α of the switch variable $\alpha_{n,t}$ that controls the observation variance: $V_{n,t} = 1$ if $\alpha_{n,t} = \{\textit{regular model}\}$ and $V_{n,t} = 100$ if $\alpha_{n,t} = \{\textit{outlier model}\}$. In our final model, we specify the evolution variance $\mathbf{W}$. Other variables are obtained or inferred from the data in an unsupervised manner. The return of an individual asset is modeled as a time varying regression, with regression coefficient vector $\boldsymbol{\theta}_{n,t}$, common factor explanatory vector $\mathbf{F}_t$, and noise: $Y_{n,t} = \boldsymbol{\theta}_{n,t}^{\mathsf{T}}\mathbf{F}_t + \sigma_{n,t}v_{n,t}$, $v_{n,t} \sim \mathrm{N}(0, V_{n,t})$. The regression coefficients are a hidden state vector, evolving as a Markov chain: $\boldsymbol{\theta}_{n,t} = \mathbf{G}_t\boldsymbol{\theta}_{n,t-1} + \sigma_{n,t}\mathbf{w}_{n,t}$, $\mathbf{w}_{n,t} \sim \mathrm{N}(\mathbf{0}, \mathbf{W})$. $\mathbf{G}_t$ captures rotation and scaling of the explanatory vector $\mathbf{F}_t$ over time, permitting computation of prior distributions for $\boldsymbol{\theta}_{n,t}$ from posterior distributions for $\boldsymbol{\theta}_{n,t-1}$. $\sigma_{n,t}$ is an asset and time specific scale factor applied to both the observation noise $v_{n,t}$ and the state evolution noise $\mathbf{w}_{n,t}$.

$\text{Cov}(\mathbf{F}) = \mathbf{I}$ and $\text{Cov}(\boldsymbol{\epsilon}) = \boldsymbol{\Psi}$. In an orthogonal factor model, the covariance of the observation matrix $\mathbf{X}$ is simply:

$$\text{Cov}(\mathbf{X}) = \mathbf{L}\mathbf{L}^{\mathsf{T}} + \boldsymbol{\Psi} \quad . \tag{4}$$

By construction, risk models based on principal component and singular value decomposition (Wall et al., 2003), including ours, possess this simple structure of orthogonal common factors and residual errors.

2.2 COMMON FACTORS

In pursing an unsupervised methodology, we utilize an SVD based approach to identifying characteristic time series. SVD identifies common variation, both row-wise and column-wise, in a matrix of data (Wall et al., 2003). SVD decomposes a rectangular matrix, such as the returns $\mathbf{X}$, into two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$; and, a diagonal matrix $\mathbf{D}$:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}} \quad . \tag{5}$$

Given the format of the $n \times t$ returns matrix $\mathbf{X}$ in Equation 3, the mutually orthogonal, unit-length right singular vectors comprising matrix $\mathbf{V}$ that represent common variation across the columns (t trading days) are the *characteristic time series*; the mutually orthogonal, unit-length left singular vectors comprising matrix $\mathbf{U}$ that represent common variation across rows (n assets) are the *characteristic portfolios*; and, the diagonal singular values matrix $\mathbf{D}$ captures scale. The entries of $\mathbf{D}$ are ordered by magnitude, therefore a p-factor model would use the first p-rows of $\mathbf{V}^{\mathsf{T}}$ for the factor return time series.

We exploit the fact that SVD methodically extracts common variation, ordered by magnitude. The intuition is that "pervasive" sources of return important to modeling portfolio level return characteristics will be reliably captured in the first several factors. Points of concern for some practitioners with regards to the use of SVD or PCA include:

1. the factors are not readily identifiable (to the human analyst);
2. the factors can be permuted in order and sign for data samples adjacent in time; and
3. it's not clear how many factors to "keep".

With respect to the first concern of (human) identifiability, we reiterate our goal is an unsupervised process yielding a high dimensional statistical model with adequate explanatory power as opposed to semantically meaningful groupings. Examination of assets with significant weight in the characteristic portfolios typically yields meaningful portfolio themes (Johnson and Wichern, 1998, Ex. 8.5). With respect to the second concern, the rotation and scaling of factors in different sample periods, our application incorporates the method of (Keane and Corso, 2012) to identify these rotations and maintain parameter distributions in the presence of rotation and scaling. Although much discussion surrounds the third concern, the identification of the "correct" number of factors (Roll and Ross, 1980; Trzcinka, 1986; Connor and Korajczyk, 1993; Onatski, 2010), we find the regularization provided by Bayesian dynamic linear models results in regression coefficients densely centered around zero for factors that are pure noise. The functioning of our process in this regard is analogous to regularized least squares (RLS) (Bishop, 2006, Ch. 3) and the ability of RLS to successfully estimate regression coefficients when confronted with a large number of candidate explanatory variables.

2.3 DYNAMIC LINEAR MODELS

The Bayesian dynamic linear model (DLM) framework elegantly addresses our need to process *streams* of data. DLMs are state space models very similar to Kalman filters (Kalman, 1960) and linear dynamical systems (Bishop, 2006, Ch. 13). We summarize the matrix variate notation and results from (West and Harrison, 1997, Ch. 16.4). Define t the time index, p the number of common factors, and n the number of assets. The observations $\mathbf{Y}_t$ are generated by matrix variate dynamic linear models characterized by four time varying parameters, $\{\mathbf{F}_t, \mathbf{G}_t, V_t, \mathbf{W}_t\}$ that define the observation and state evolution equations. We now define and comment on the DLM parameters as they pertain to our application:

- $\mathbf{Y}_t = \left[Y_{t,1}, \ldots, Y_{t,n}\right]^{\mathsf{T}}$, log price returns at time t, common to all component DLMs;
- $\mathbf{F}_t$ a $p \times 1$ dynamic regression vector, factor returns at time t, common to all component DLMs;
- $\mathbf{G}_t$ a $p \times p$ state evolution matrix, accounts for rotation and scaling of factor return time series at time t, common to all component DLMs;
- V_t an observational variance scalar, individually specified for each component DLM, greatly inflated in the DLMs generating "outliers" at time t;
- $\mathbf{W}_t$ an evolution variance matrix, individually specified for each component DLM, controls rate of change in factor loadings at time t;
- $\boldsymbol{\Sigma}_t = \begin{bmatrix} \sigma^2_{t,1} & & \\ & \ddots & \\ & & \sigma^2_{t,n} \end{bmatrix}$, unknown diagonal ma-

trix composed of n-asset specific variance scales at time t;

- $\boldsymbol{\nu}_t$, $n \times 1$ vector of unknown observation errors at time t;
- $\boldsymbol{\Theta}_t = \left[\theta_{t,1}, \ldots, \theta_{t,n}\right]$, a $p \times n$ unknown state matrix whose columns are factor loadings of individual assets on common factor returns at time t; and,
- $\boldsymbol{\Omega}_t = \left[\omega_{t,1}, \ldots, \omega_{t,n}\right]$, a $p \times n$ unknown evolution errors matrix applied to the state matrix at time t.

We specify our model variances in scale free form (West and Harrison, 1997, Ch. 4.5), implying multiplication by asset specific scales $\sigma^2_{t,i}$ in univariate DLMs: $\mathbf{C}_t = \sigma^2_{t,i}\mathbf{C}^*_t$, $V_t = \sigma^2_{t,i}V^*_t$, and $\mathbf{W}_t = \sigma^2_{t,i}\mathbf{W}^*_t$. When using matrix variate notation, $\boldsymbol{\Sigma}_t$ is a right variance parameter, discussed below, scaling the n columns of the matrix on which it operates. When we specify models in § 3.5, we will specify scale free parameters $V^*_t \in \left\{1, 100\right\}$ and $\mathbf{W}^*_t \in \left\{.00001, .001, .1\right\}$. For simplicity, we shall omit scale free notation.

The observation equation is:

$$\mathbf{Y}_t = \boldsymbol{\Theta}_t^{\mathsf{T}}\mathbf{F}_t + \boldsymbol{\nu}_t, \quad \boldsymbol{\nu}_t \sim \mathrm{N}[\mathbf{0}, V_t\boldsymbol{\Sigma}_t] \quad . \tag{6}$$

The distribution of the observation errors $\boldsymbol{\nu}_t$ is multivariate normal, with mean vector $\mathbf{0}$ and unknown variance $V_t\boldsymbol{\Sigma}_t$.

The state evolution is:

$$\boldsymbol{\Theta}_t = \mathbf{G}_t\boldsymbol{\Theta}_{t-1} + \boldsymbol{\Omega}_t, \quad \boldsymbol{\Omega}_t \sim \mathrm{N}[\mathbf{0}, \mathbf{W}_t, \boldsymbol{\Sigma}_t] \quad . \tag{7}$$

The distribution of the state matrix evolution errors $\boldsymbol{\Omega}_t$ is *matrix normal* (Dawid, 1981), with mean matrix $\mathbf{0}$, left variance matrix $\mathbf{W}_t$ controlling variation across rows (factors) of $\boldsymbol{\Theta}_t$, and right variance matrix $\boldsymbol{\Sigma}_t$ controlling variation across columns (assets) of $\boldsymbol{\Theta}_t$. As our implementation uses diagonal matrices for both $\mathbf{W}_t$ and $\boldsymbol{\Sigma}_t$, we implicitly assume independence in the evolution of factor loadings across the factors $i \in 1 \ldots p$ and across the assets $j \in 1 \ldots n$.

Mapping factor model notation of (Johnson and Wichern, 1998) in Equation 3 to DLM notation of (West and Harrison, 1997) in Equation 6: $\mathbf{X} \rightarrow \left[\mathbf{Y}_1 \ldots \mathbf{Y}_t\right]$; $\mathbf{L} \rightarrow \boldsymbol{\Theta}_t^{\mathsf{T}}$; $\mathbf{F} \rightarrow \left[\mathbf{F}_1 \ldots \mathbf{F}_t\right]$; and, $\boldsymbol{\epsilon} \rightarrow \left[\boldsymbol{\nu}_1 \ldots \boldsymbol{\nu}_t\right]$. The crucial change in perspective involves the regression coefficients, $\mathbf{L} \rightarrow \boldsymbol{\Theta}_t^{\mathsf{T}}$. Where as the other matrices in Equation 3 are simply collections of columns present in Equation 6, the static regression coefficients $\mathbf{L}$ now evolve with time in Equation 7 as $\boldsymbol{\Theta}_t^{\mathsf{T}}$.

As typical with a Bayesian approach, our process begins with a prior distribution reflecting our belief about the unknown state parameter matrix $\boldsymbol{\Theta}_0$ and the unknown variance scale matrix $\boldsymbol{\Sigma}_0$ before data arrives. Our initial belief is expressed as a *matrix normal/inverse Wishart* distribution:

$$(\boldsymbol{\Theta}_0, \boldsymbol{\Sigma}_0) \sim \mathrm{NW}^{-1}_{\delta_0}[\mathbf{m}_0, \mathbf{C}_0, \mathbf{S}_0] \quad , \tag{8}$$

with mean matrix $\mathbf{m}_0$, left variance matrix $\mathbf{C}_0$, right variance matrix $\mathbf{S}_0$, and degrees of freedom δ_0. We allow our estimate of observational variance to vary over time by decaying our sample variance degrees of freedom parameter δ_{t-1} immediately before computation of the prior distribution Equation 10.

The marginal distribution for the state matrix $\boldsymbol{\Theta}_0$ is a *matrix T* distribution:

$$\boldsymbol{\Theta}_0 \sim \mathrm{T}_{\delta_0}[\mathbf{m}_0, \mathbf{C}_0, \mathbf{S}_0] \quad . \tag{9}$$

Let $\mathbf{D}_t = \left[\mathbf{Y}_t \ldots \mathbf{Y}_0\right]$ refer to the information available subsequent to observing $\mathbf{Y}_t$. The conjugate parameter distributions are updated as follows.

Prior distribution at t:

$$(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}_t | \mathbf{D}_{t-1}) \sim \mathrm{NW}^{-1}_{\delta_{t-1}}[\mathbf{a}_t, \mathbf{R}_t, \mathbf{S}_{t-1}] \quad , \tag{10}$$

where $\mathbf{a}_t = \mathbf{G}_t\mathbf{m}_{t-1}$ and $\mathbf{R}_t = \mathbf{G}_t\mathbf{C}_{t-1}\mathbf{G}_t^{\mathsf{T}} + \mathbf{W}_t$.

Forecast distribution at t given the dynamic regression vector $\mathbf{F}_t$:

$$(\mathbf{Y}_t | \mathbf{D}_{t-1}) \sim \mathrm{T}_{\delta_{t-1}}[\mathbf{f}_t, Q_t\mathbf{S}_{t-1}] \quad , \tag{11}$$

where $\mathbf{f}_t = \mathbf{a}_t^{\mathsf{T}}\mathbf{F}_t$ and $Q_t = \left\{V_t + \mathbf{F}_t^{\mathsf{T}}\mathbf{R}_t\mathbf{F}_t\right\}$.

In our application, $\mathbf{F}_t$ is not available until $\mathbf{Y}_t$ is observed. Therefore, we accommodate random regression vectors $\mathbf{F}_t$ (Wang et al., 2011, § 7). Define $\mu_{\mathbf{F}_t} = \mathrm{E}(\mathbf{F}_t | \mathbf{D}_{t-1})$ and $\boldsymbol{\Sigma}_{\mathbf{F}_t} = \mathrm{Cov}(\mathbf{F}_t | \mathbf{D}_{t-1})$. The forecast distribution with $\mathbf{F}_t$ unknown is $(\mathbf{Y}_t | \mathbf{D}_{t-1}) \sim \mathrm{T}_{\delta_{t-1}}[\hat{\mathbf{f}}_t, \hat{Q}_t\mathbf{S}_{t-1}]$, where the moment parameters of the multivariate T forecast distribution are now $\hat{\mathbf{f}}_t = \mathbf{a}_t^{\mathsf{T}}\mu_{\mathbf{F}_t}$ and $\hat{Q}_t\mathbf{S}_{t-1} = \left\{V_t + \mu_{\mathbf{F}_t}^{\mathsf{T}}\mathbf{R}_t\mu_{\mathbf{F}_t} + \mathrm{tr}\left(\mathbf{R}_t\boldsymbol{\Sigma}_{\mathbf{F}_t}\right)\right\}\mathbf{S}_{t-1} + \mathbf{a}_t^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{F}_t}\mathbf{a}_t$.

Posterior distribution at t:

$$(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}_t | \mathbf{D}_t) \sim \mathrm{NW}^{-1}_{\delta_t}[\mathbf{m}_t, \mathbf{C}_t, \mathbf{S}_t] \quad , \tag{12}$$

with $\mathbf{m}_t = \mathbf{a}_t + \mathbf{A}_t\mathbf{e}_t^{\mathsf{T}}$, $\mathbf{C}_t = \mathbf{R}_t - \mathbf{A}_t\mathbf{A}_t^{\mathsf{T}}Q_t$, $\delta_t = \delta_{t-1} + 1$ and $\mathbf{S}_t = \delta_t^{-1}\left[\delta_{t-1}\mathbf{S}_{t-1} + \mathbf{e}_t\mathbf{e}_t^{\mathsf{T}}/Q_t\right]$ where $\mathbf{A}_t = \mathbf{R}_t\mathbf{F}_t/Q_t$ and $\mathbf{e}_t = \mathbf{Y}_t - \mathbf{f}_t$.

2.4 UNIVARIATE DLMS

In § 2.3, we summarized results for matrix variate DLMs. Setting the number of assets $n = 1$, results for univariate DLMs immediately follow.

2.5 MULTI-PROCESS MODELS

(West and Harrison, 1997, Ch. 12) define multi-process models composed of component DLMs. Consider a set of DLMs $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$. Let α_t reference the component DLM realized at time t, $\mathcal{A}_{\alpha_t} \in \mathcal{A}$. If the observations Y_t are generated with one unknown DLM $\alpha_t = \alpha$ for all time, the observations are said to follow a *multi-process, class I model.* If at different times $s \neq t$, the observations are generated by distinct DLMs $\alpha_s \neq \alpha_t$, the observations are said to follow a *multi-process, class II model.* In an unsupervised modeling process, we need to accommodate the arrival of both typical and outlier observations. We accomplish this with a multi-process class II model, where various component DLMs are appropriate for various subsets of the observations. We assume fixed model selection probabilities, $\pi_t(\mathcal{A}_\alpha) = \pi(\mathcal{A}_\alpha)$.

With class II models, there are $|\mathcal{A}|^t$ potential model histories for *each* asset. We avoid this explosion in model sequences by considering only two periods, $t-1$ and t, thereby limiting distinct model sequences in our mixtures to $|\mathcal{A}|^2$. As each asset has its own history, no longer sharing common scale free posterior variance $\mathbf{C}_t$, our mixture models are asset specific, forcing the use of univariate component DLMs. Parameters 1×1 in univariate DLMs are now displayed with scalar notation. To avoid clutter, we omit implicit asset subscripts.

Inference with multi-process models is based upon manipulation of various model probabilities: the posterior model probabilities for the last model $p_{t-1}(\alpha_{t-1})$; the prior model probabilities for the current model $\pi(\alpha_t)$; and, the model sequence likelihoods $p(Y_t|\alpha_t, \alpha_{t-1}, D_{t-1})$. Posterior model sequence probabilities for current model α_t and last model α_{t-1} upon observing Y_t are:

$$\begin{aligned} p_t(\alpha_t, \alpha_{t-1}) &= \Pr[\alpha_t, \alpha_{t-1}|D_t] \\ &\propto p_{t-1}(\alpha_{t-1})\pi(\alpha_t)p(Y_t|\alpha_t, \alpha_{t-1}, D_{t-1}) \quad . \end{aligned} \tag{13}$$

The unconditional posterior parameter distributions are computed as mixtures of the $|\mathcal{A}|^2$ component DLM sequences

$$p(\boldsymbol{\theta}_t|D_t) = \sum_{\alpha_t=1}^{k} \sum_{\alpha_{t-1}=1}^{k} p_t(\boldsymbol{\theta}_t|\alpha_t, \alpha_{t-1}, D_t) p_t(\alpha_t, \alpha_{t-1}) \; . \tag{14}$$

The posterior model probabilities are

$$p_t(\alpha_t) = \Pr[\alpha_t|D_t] = \sum_{\alpha_{t-1}=1}^{k} p_t(\alpha_t, \alpha_{t-1}) \quad . \tag{15}$$

The posterior probabilities for the last model α_{t-1} given the current model α_t and information D_t are

$$\Pr[\alpha_{t-1}|\alpha_t, D_t] = \frac{p_t(\alpha_t, \alpha_{t-1})}{p_t(\alpha_t)} \quad . \tag{16}$$

After each time step, the posterior mixture distribution for each component DLM is approximated with an analytic distribution using the methodology described in (West and Harrison, 1997, Ch. 12.3.4). The Kullback-Leibler directed divergence between the approximation and the mixture is minimized in the parameters: $\mathbf{m}_t(\alpha_t)$, $\mathbf{C}_t(\alpha_t)$, $S_t(\alpha_t)$, and $\delta_t(\alpha_t)$. Let $S_t(\alpha_t, \alpha_{t-1})$ refer to the variance scale estimate obtained with the DLM sequence α_{t-1}, α_t. The parameters of the approximating distributions are as follows. The variance scale estimates $S_t(\alpha_t)$ are:

$$S_t(\alpha_t)^{-1} = \frac{1}{p_t(\alpha_t)} \sum_{\alpha_{t-1}=1}^{k} \frac{p_t(\alpha_t, \alpha_{t-1})}{S_t(\alpha_t, \alpha_{t-1})} \quad . \tag{17}$$

The weights for computing the moments of the KL divergence minimizing approximation to the posterior distribution are:

$$p_t^*(\alpha_{t-1}) = \frac{S_t(\alpha_t)}{p_t(\alpha_t)} \frac{p_t(\alpha_t, \alpha_{t-1})}{S_t(\alpha_t, \alpha_{t-1})} \quad . \tag{18}$$

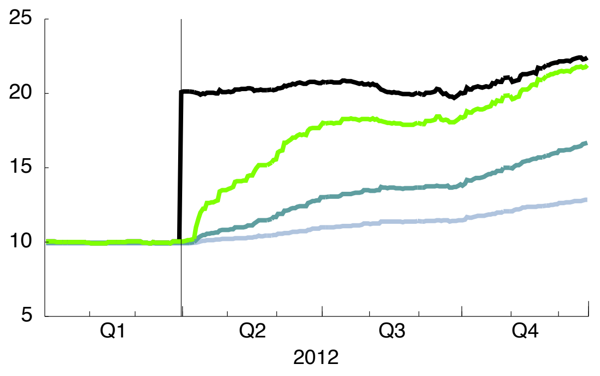
The mean vector $\mathbf{m}_t(\alpha_t)$ for DLM α_t is:

$$\mathbf{m}_t(\alpha_t) = \sum_{\alpha_{t-1}=1}^{k} p_t^*(\alpha_{t-1})\mathbf{m}_t(\alpha_t, \alpha_{t-1}) \quad . \tag{19}$$

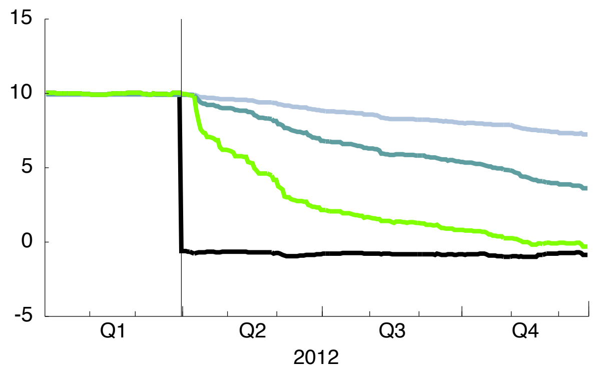
The variance matrix $\mathbf{C}_t(\alpha_t)$ for DLM α_t is:

$$\begin{aligned} \mathbf{C}_t(\alpha_t) = \sum_{\alpha_{t-1}=1}^{k} p_t^*(\alpha_{t-1}) \Big\{ & \mathbf{C}_t(\alpha_t, \alpha_{t-1}) + \\ & [\mathbf{m}_t(\alpha_t) - \mathbf{m}_t(\alpha_t, \alpha_{t-1})] \quad \times \\ & [\mathbf{m}_t(\alpha_t) - \mathbf{m}_t(\alpha_t, \alpha_{t-1})]^{\mathsf{T}} \Big\} \quad . \end{aligned} \tag{20}$$

The degrees of freedom parameter, $\delta_t(\alpha_t)$ is important to the KL minimization. Intuitively, if the component DLMs are in discord, the resulting mixture may be described as "fat-tailed", and the precision of the unknown variance scale parameter reduced. We compute $\delta_t(\alpha_t)$ using the procedure described by (West and Harrison, 1997, Ex. 12.7), with a further correction term. The approximation West and Harrison utilize, based upon an algorithm for computing the digamma function $\gamma(x)$ discussed in (Bernardo, 1976), is appropriate when $x \to \infty$. However, when estimating the *reciprocal* of the KL minimizing $\delta_t(\alpha_t)$, we find the error in the approximation remains rather constant, and we apply a correction to eliminate this constant.



(a) Synthetic scenario 1: abrupt increase in factor loading θ_t, units are percent per annum. Observations Y_t are synthesized returns using SPY (S&P 500 ETF) until March 30, 2012; and, 2×SPY thereafter. Note "true" factor loading doubles from approximately 10 to 20.



(b) Synthetic scenario 2: abrupt decrease in factor loading θ_t, units are percent per annum. Observations Y_t are synthesized returns using SPY until March 30, 2012; and, AGG (Barclays Aggregate Bond ETF) thereafter. Note "true" factor loading drops from approximately 10 to 0.

Figure 2: Unmonitored mixture models responding to abrupt change. Black line is true value of latent state variable $\theta_{1,t}$. Other lines are the posterior mean $\mathbf{m}_{1,t}$ (first common factor loading) obtained with three different mixture models discussed in § 3.5. None of the three above models responded quickly enough to the dramatic change in dynamics. A method of intervention to improve responsiveness is discussed in § 4.5.

3 APPLICATION DESIGN

3.1 END USERS

Our application enables a proprietary trading group at a financial services firm to better understand the aggregate behavior of stock portfolios. The models provide a statistical framework required for constructing portfolios, assessing portfolio risk, and clustering assets. The assets of primary interest are the common shares of the largest 1000 - 2000 companies in the US stock market. The group focuses on larger companies because the liquidity of larger stocks generally makes them cheaper and easier to transact. The group's strategies include short selling: borrowing stock, selling borrowed shares; and, attempting to profit by repurchasing the shares at a lower price. Larger stocks are generally easier to borrow.

3.2 BIAS-VARIANCE TRADEOFF

In implementing our application, one of the first issues encountered is a machine learning classic, the bias-variance tradeoff [Ch. 2.9](Hastie et al., 2009). With respect to DLMs, a trade off is incurred in the effective number of observations as the evolution variance is varied. A model with greater evolution variance will generate parameter distributions with greater variance but lower bias. A model with lower evolution variance will generate parameter distributions with lower variance but greater bias. A relatively smooth, lethargic, slowly adapting model does not track evolving dynamics as quickly as a rapidly adapting model; on the other hand, the quickly adapting model delivers a noisier sequence of parameter distributions. Outside our applied context, the loss function might be specified as squared error or absolute error. In the context of a risk model, the loss function should consider a portfolio manager's cost of over-trading due to a model adapting excessively (variance); as well a risk manager's problems arising in a system adapting too slowly (bias). The appropriate loss function depends critically on the intended end use. A quantitative trader constructing portfolios with quadratic optimization tends to magnify errors in a model, as the optimization process responds dynamically to parameter estimates (Muller, 1993). In contrast, a firm-wide risk manager, who typically evaluates sums of individual asset exposures, but does not dynamically respond to individual asset risk attributes, may prefer less bias and more variance, as error in the factor loadings of one asset may be offset by error in another asset in the summation process. We construct a variety of mixtures along the bias-variance continuum as discussed in § 3.5 and as illustrated in Figure 2.

3.3 UNIVERSE OF ASSETS

We identify two universes of assets: a relatively narrow set that will be used to construct explanatory time series; and, an all inclusive set for which we will generate factor loading and residual volatility estimates. It is desirable that the assets used to construct the common factor returns trade frequently and with adequate liquidity to minimize pricing errors. We also avoid *survivor bias*, the methodological error of omitting companies no longer in existence at the time a historical analysis is performed. We eliminate this hazard by defining our common factor estimation universe using daily exchange traded fund (ETF) create / redeem portfolios for the last 10 years. Brokers *create* ETF shares (in units of 50,000 ETF shares) by delivering a defined portfolio of stock and cash in exchange for ETF shares; or, they *redeem* ETF shares and receive the defined portfolio of stock and cash. Given the create / redeem definitions that were used as the basis for large transactions during the historical period, the assets in an ETF portfolio represent an institutionally held, survivor bias free, tradeable universe. Its likely the component shares were readily available to borrow, as the component shares were held by custodian banks for the ETF shareholders. Our data base permits us to obtain data for surviving and extinct stocks. The ETF we select for our factor estimation universe is the Vanguard Total Stock Market ETF (VTI) (Vanguard Group, Inc., 2014). As of April 2014, including both mutual fund and ETF share classes, the Vanguard Total Stock Market fund size was approximately USD 330 billion. The VTI constituents closely approximates our desired universe, with the number of component stocks typically ranging from 1300 - 1800.

3.4 DATA PREPARATION

Data preparation involves constructing the artifacts demanded by § 2.3: $\mathbf{Y}_t$, $\mathbf{F}_t$, and $\mathbf{G}_t$. Each trading day, using price, dividend, and corporate action data for all 7000 - 8000 stocks, ADRs, and ETFs in our US pricing data base (MarketMap Analytic Platform, 2014), we construct dividend and split adjusted log price return observation vectors $\mathbf{Y}_t$. For stocks in the VTI ETF on that day, we construct a variance equalized historical returns matrix $\mathbf{r}_t = \left[\mathbf{Y}_{t-T+1} \dots \mathbf{Y}_t\right] \hat{\mathbf{\Sigma}}_t^{-\frac{1}{2}}$ where $\hat{\mathbf{\Sigma}}_t$ is the diagonal matrix of sample variance for the period $t - T + 1$ to T. Using (Keane and Corso, 2012, §3.c), we compute $\mathbf{F}_t$, from the first p right singular vectors from a singular value decomposition of $\mathbf{r}_t$. As the vectors are unit length, and we desire unit variance per day, $\text{Cov}(\mathbf{F}_t) = \mathbf{I}$, we scale the right singular vectors by $\sqrt{T}$. The scaled characteristic time series from adjacent data windows, $\mathbf{r}_{t-1} = \left[\mathbf{Y}_{t-T} \dots \mathbf{Y}_{t-1}\right] \hat{\mathbf{\Sigma}}_{t-1}^{-\frac{1}{2}}$ and $\mathbf{r}_t = \left[\mathbf{Y}_{t-T+1} \dots \mathbf{Y}_t\right] \hat{\mathbf{\Sigma}}_t^{-\frac{1}{2}}$ are then used to compute $\mathbf{G}_t$ as described in (Keane and Corso, 2012, §3.e):

$$\mathbf{G}_t = \left(\mathbf{F}_t \mathbf{F}_t^{\mathsf{T}}\right)^{-1} \mathbf{F}_t \mathbf{F}_{t-1}^{\mathsf{T}} \quad . \tag{21}$$

We are a little flexible with the notation in Equation 21, where $\mathbf{F}_t$ and $\mathbf{F}_{t-1}$ are $p \times (T-1)$ sub-matrices representing time aligned subsets of two factor return matrices, the scaled right singular vectors obtained from the decomposition of $\mathbf{r}_{t-1}$ and $\mathbf{r}_t$. Elsewhere, viz. Equation 11, $\mathbf{F}_t$ refers to a $p \times 1$ dynamic regression vector, the right most column of the transposed and scaled right singular vectors, corresponding to the desired vector of common factor returns for day t.

3.5 MODEL COMPONENTS

The component DLMs in our mixture model share observations $\mathbf{Y}_t$, common factor scores $\mathbf{F}_t$, and state evolution matrices $\mathbf{G}_t$. The component DLMs are differentiated by the variance parameters: the observational variance scale V_t, and the evolution variance matrix $\mathbf{W}_t$. We construct a set of component DLMs following the approach of (West and Harrison, 1997, Ch. 12.4). For component DLMs that will accommodate "typical" observation variance, we set $V_t = 1$; for component DLMs that will accommodate outlier observations, we set $V_t = 100$. For the evolution variance, we similarly select a base rate of evolution $\mathbf{W}_t = .00001$; and, inflate $\mathbf{W}_t$ by a factor of 100 and 100^2 to permit increasingly rapid changes in the factor loadings.

3.6 OUTPUT

The format of the output risk model will be a $n \times p$ factor loading matrix and a $n \times 1$ residual volatility vector. These $p+1$ numeric attributes for n stocks are stored in various formats for subsequent use throughout the organization.

4 EVALUATION

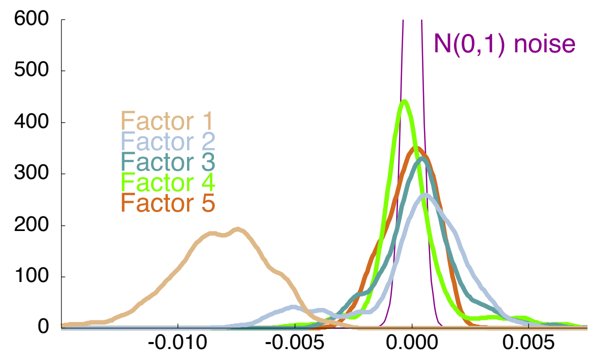
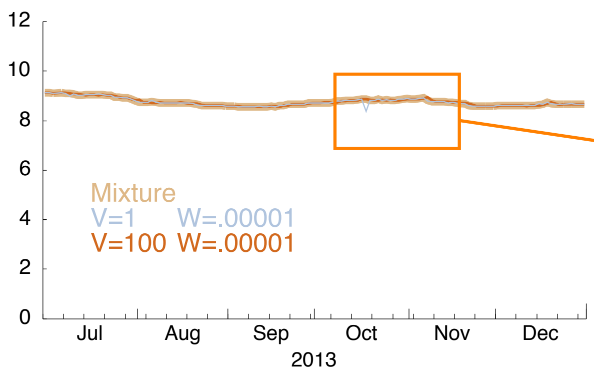


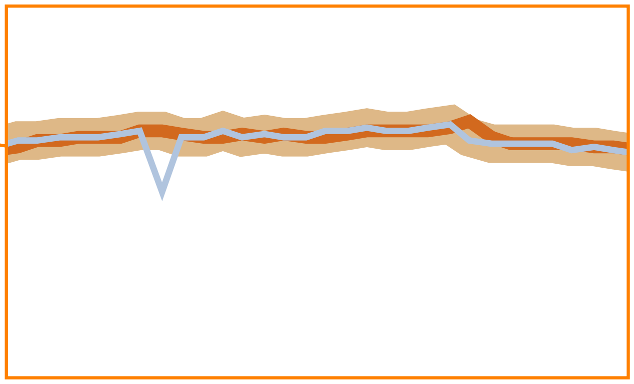
Figure 3: Distribution of factor loadings for the Vanguard Total Market Index portfolio on April 30, 2014. Left axis is density, $\int dx = 1$.

4.1 NUMBER OF FACTORS

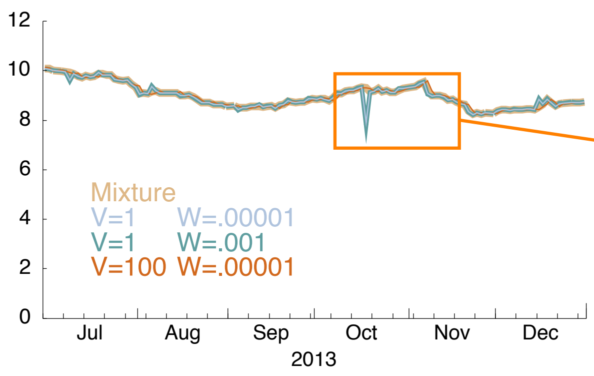
A Bayesian DLM updated with an "explanatory" series of pure noise is expected to generate posterior



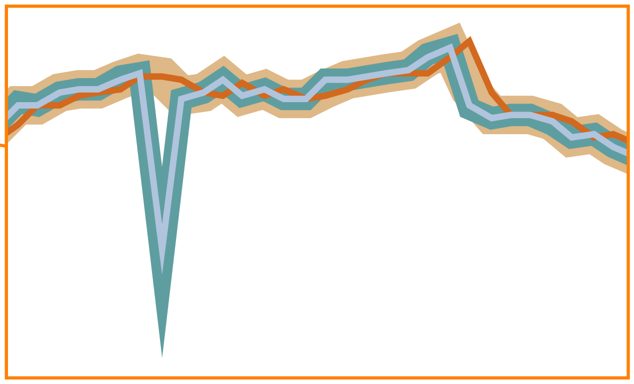
(a)



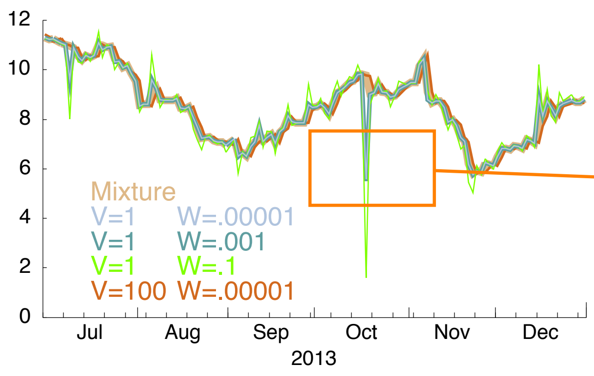
(b)



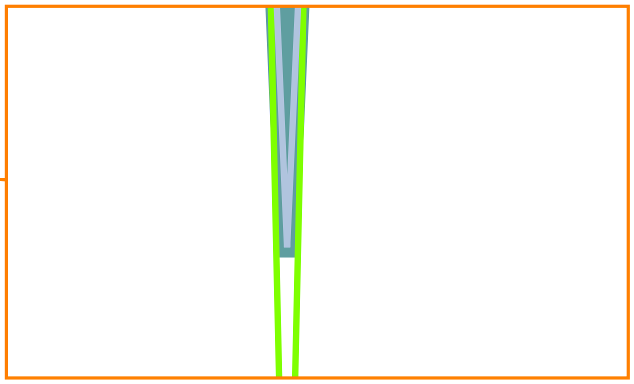
(c)



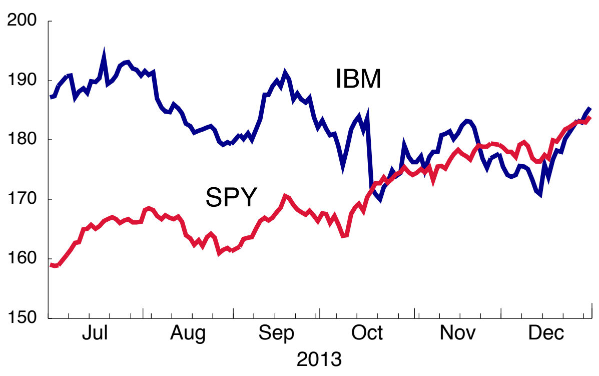
(d)



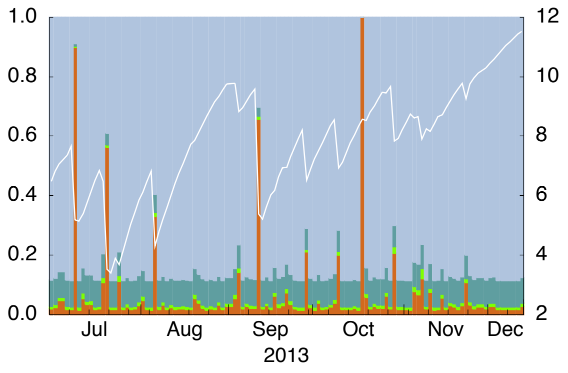
(e)



(f)



(g) Price histories for IBM and SPY, units are USD.



(h) Posterior component probabilities *(left)*, color key in (c); mixture model degrees of freedom δ_t *(right)*, white line.

Figure 4: Multi-Process Models. See § 4.3 for discussion. Figure 4(a),(c),(e) units are percent per annum.

regression parameter distributions densely surrounding zero given sufficient observations. Further, a linear combination of independent DLMs is a DLM (West and Harrison, 1997, *Principle of Superposition*, p. 188). We use these two points to justify the inclusion of a relatively large number of independent explanatory series. Given the regularization inherent in Bayesian DLMs, we believe the risk of omitting a common factor far exceeds the risk of including noise. In our application, we focus on aggregate (portfolio level) forecasts, therefore it is extremely important to identify common sources of variation that may appear insignificant at the individual asset level. (West and Harrison, 1997, Ch. 16.3) discuss in detail the hazard of omitted common factors in aggregate forecasts. In Figure 3, we show the distribution of factor loadings for the VTI constituents on April 30, 2014. The distribution of factor loadings for the first five common factors obtained from our two component mixture model are shown in comparison to the distribution of loadings on Gaussian noise, $F_t \sim \mathrm{N}[0,1]$. Figure 3 is consistent with our viewpoint, note the high density of zero loadings for the noise series, and the relatively diffuse factor loadings for the first five common factor series extracted with SVD. The factor loadings on the noise series have a mean loading $\mu_m = 0$ bp[2], and a standard deviation of $\sigma = 2$ bp. In contrast, the loadings on the first factor have a mean loading of $\mu_m = -82$bp and a standard deviation of $\sigma_m = 21$bp.

4.2 MIXTURE DYNAMICS

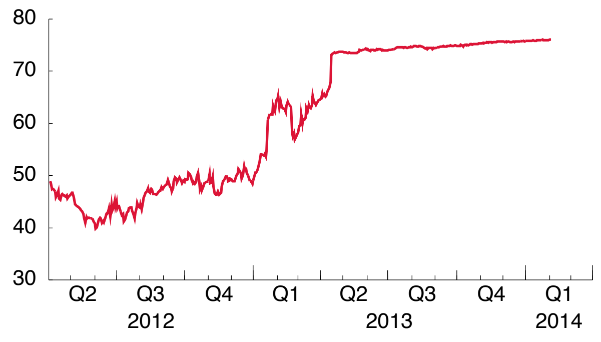
Figure 4 shows the price movement and model response for IBM during the second half of 2013. In Figure 4(g), the price histories for IBM and the S&P 500 ETF SPY are displayed. Note the sharp drop in IBM's price on October 17, 2013 corresponding to an earnings announcement. This event is helpful in understanding the interaction of components in our multi-process models. We construct three multi-process models, the simplest of which is a two component mixture (the "base model"), comprised of a standard component DLM to handle the majority of the observations Y_t, and an outlier component DLM. We specify common evolution variance $\mathbf{W}_t = .00001\ \mathbf{I}$; observation variance $\mathbf{V}_t = 1$ for the standard component; and observation variance $\mathbf{V}_t = 100$ for the outlier component. The base model and component estimates for the first factor loading $\mathbf{m}_{1,t}$ appear in Figure 4(a) and magnified in Figure 4(b). We specify a three component mixture (the "adaptive model") by adding a component DLM with inflated evolution variance $\mathbf{W}_t = .001\ \mathbf{I}$. The adaptive model and component estimates for the first factor loading $\mathbf{m}_{1,t}$ appear in Figure 4(c) and magnified in Figure 4(d). Finally, we specify a four component mixture (the "very adaptive model") by adding a component with further inflated evolution variance $\mathbf{W}_t = .1\ \mathbf{I}$. The very adaptive model and component estimates for the first factor loading $\mathbf{m}_{1,t}$ appear in Figure 4(e) and magnified in Figure 4(f). The posterior component model probabilities for the very adaptive model appear as a bar chart in Figure 4(h), where the bottom bar corresponds to the probability of the outlier component, the second bar corresponds to the very adaptive component DLM, the third bar corresponds to the adaptive component DLM, and the top bar corresponds to the base component DLM. We specified the fixed DLM selection (prior) probabilities as $\{.01543, .00887, .0887, .887\}$ for the components $\{$*outlier, very adaptive, adaptive, base*$\}$ respectively. Note several occurrences where the posterior probability of an outlier observation significantly exceeds the DLM selection probability. The white line in Figure 4(h) corresponds to the degrees of freedom parameter δ_t for the T-distribution that approximates the mixture model's posterior parameter distribution.

4.3 QUALITATIVE COMMENTS

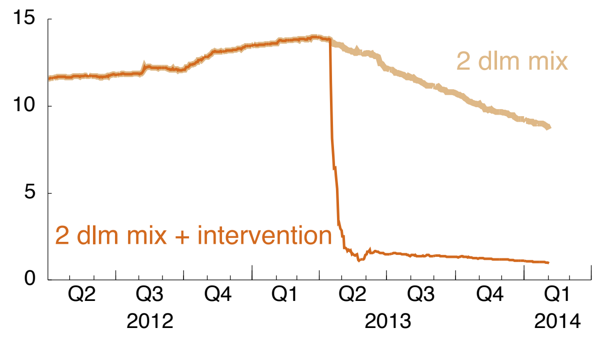
To supplement the more analytically precise discussion in § 2.5, we make the following qualitative comments as to the interaction of the mixture components:

- the mixture in Figure 4(e) with larger evolution variance adapts faster; the mixture in Figure 4(a) with smaller evolution variance appears smoother;
- time t component posteriors are 1-period departures from the $t-1$ consensus, see Figure 4(a), (b), (c), (d), and (e);
- an outlier component "ignores" current observations Y_t and forecast error $|\mathrm{e}_t|$, responding to the $t-1$ posterior consensus $\mathbf{m}_{t-1}$, see Figure 4(b) and (d);
- in periods of noise, the other components return to the outlier component's estimate with 1-period lag, see left-hand side of Figure 4(b) and (d);
- in periods of level change, the outlier follows the other components' estimate with 1-period lag, see right-hand side of Figure 4(b) and (d);
- when the posterior probability of the outlier component spikes up, the degrees of freedom parameter δ_t usually drops, reducing the precision of the variance scale estimate S_t, see Figure 4(h);
- *however*, when the outlier component is selected with very high probability, there is no impact to δ_t as the observation is ignored, see October 17,

[2]A basis point (bp) is $10,000^{-1}$.



(a) Price history, units are USD.



(b) Posterior mean $\mathbf{m}_{1,t}$, units are percent per annum.

Figure 5: In a deal announced April 15, 2013, Life Technologies Corporation was acquired by Thermo Fisher Scientific Inc. for USD 14 billion. The change in price behavior is noticable in Figure 5a. The factor loading estimates for two mixture models are compared in Figure 5b. One mixture model is unmonitored, the other benefits from intervention subsequent to the news event. See discussion in § 4.5

2013 in Figure 4(h), noting that the white line does not drop when Pr{ *outlier* } ≈ 1;

- except for the very adaptive component, the response does not vary with $\mathbf{W}_t$, see Figure 4(d) and (f), where W=.001 and W=.00001 responses are nearly identical.

4.4 UNRESPONSIVENESS TO $\mathbf{W}_t$

The phenomenon we find most surprising is the insensitivity of the components to $\mathbf{W}_t$ below a certain threshold. Digging further into this phenomenon, in a mixture, the various component models view of the $t-1$ posterior parameter distribution are very similar. Thinking about univariate DLMs, and assuming for discussion $F = 1$ and $G = 1$, the magnitude of the adaptive scalar $A_t = R_t/Q_t = (C_{t-1} + W_t) / (C_{t-1} + W_t + V_t)$. When $W_t \ll C_{t-1}$, as describes our situation, $A_t \approx C_{t-1}/ (C_{t-1} + V_t)$ as seen in Figure 4(d) and (f). Only when W_t is significant relative to C_{t-1} does response vary noticeably.

4.5 INTERVENTION

Our discussion of IBM focused on the mixture models' processing of unusual observations. We now explore an example where the data generating process changes abruptly, similar to our synthetic illustrations in Figure 2. In April 2013, the acquisition of Life Technologies Corporation by Thermo Fisher Scientific Inc. was announced. The stock's sensitivity to the market, as expressed in its first common factor loading, dropped abruptly, as shown in Figure 5. While our goal is an unsupervised estimation processes, the Bayesian DLM framework facilitates structured intervention when necessary. For one of the mixture models in Figure 5b, we intervene and inflate the prior parameter variance following the April 15th announcement, $\mathbf{R}_t^{++} = \mathbf{G}_t\,(\mathbf{C}_{t-1} + \mathbf{I})\,\mathbf{G}^{\mathsf{T}} + \mathbf{W}_t$, where the identity matrix reflects the increased uncertainty in the parameter distribution relative to the usual prior variance in Equation 10. When subsequent updates occur, the DLM with the inflated prior variance adapts to the new dynamics rapidly and satisfactorily.

Table 1: Risk Model Performance

	Volatility	s.e.	*t*-stat
GMV portfolio			
PCA (1)	4.62	0.07	40.48
PCA (10)	3.41	0.05	29.87
DLM (10)	2.17	0.03	6.82
Mixture (10)	**1.96**	0.03	
MSR portfolio			
PCA (1)	4.07	0.06	49.38
PCA (10)	2.06	0.03	28.84
DLM (10)	1.30	0.02	4.39
Mixture (10)	**1.22**	0.02	
Cap Weight	20.31	0.29	
Equal Weight	24.62	0.35	

4.6 RISK MODEL PERFORMANCE

To access the performance of our two component mixture model, we construct daily portfolios from the VTI universe for the most recent ten years, May 2004 to April 2014. The number of trading days during this period was 2,516. In Table 1, we report realized out-of-sample volatility and the standard error (se) of the volatility measure for two strategies: global minimum variance (GMV); and, maximum Sharpe ratio (MSR) (Demey et al., 2010). We implement four risk models: 1-factor PCA; 10-factor PCA; 10-factor DLM; and, 10-factor 2-component mixture model. We permit short positions, and do not constrain position weights. The PCA models are constructed using (Connor and Korajczyk, 1988). The mixture model is the same model we presented earlier, with noise "factor ten" present. For each strategy, we report the t-statistics for the various models' realized volatility compared to the mixture model's realized volatility. For context on the ambient volatility of the ten year period, we provide realized volatility for two portfolios that are long only and do not use a risk model: the capitalization weighted portfolio; and the equal weighted portfolio.

5 CONCLUSION

The ability to integrate SVD with Bayesian methods allows our application to process large data streams in an unsupervised fashion. We demonstrate that a two component multi-process model achieved better reduction in volatility than alternative models. The two component model out-performed alternative models including a single process model. We find the robustness of Bayesian DLMs with respect to noise inputs of great practical value, allowing us to favor inclusion of factors, potentially capturing pervasive sources of common movement important to aggregate forecasting. The inclusion of an outlier model adds great functionality, delivering robustness to the estimation process. The insensitivity of the mixture models to multiple evolution variance values leads us to favor mixtures of just two components, a typical evolution variance value for both components, and an inflated observation variance in the outlier component. We would recommend generating several models of varying adaptiveness, evaluating the variance-bias tradeoff in light of a user's specific situation. We favor the judicious use of intervention for events such as mergers. We would like to explore using news feeds to systematically intervene for events known to impact an assets dynamics.

References

J.M. Bernardo. Psi (digamma) function. *Applied Statistics*, 25(3):315–317, 1976.

C.M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

G. Connor and R.A. Korajczyk. Risk and return in an equilibrium APT. *Journal of Financial Economics*, 21(2):255–289, 1988.

G. Connor and R.A. Korajczyk. A test for the number of factors in an approximate factor model. *The Journal of Finance*, 48(4):1263–1291, 1993.

A.P. Dawid. Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, 68(1):265, 1981.

P. Demey, S. Maillard, and T. Roncalli. Risk based indexation. Technical report, Lyxor Asset Management, Paris, 2010.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*, volume 2. Springer, 2009.

R.A. Johnson and D.W. Wichern. *Applied multivariate statistical analysis*, volume 4. Prentice Hall, 1998.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

K.R. Keane and J.J. Corso. Maintaining prior distributions across evolving eigenspaces. In *International Conference on Machine Learning and Applications.* IEEE, 2012.

MarketMap Analytic Platform. *North American Pricing.* SunGard Data Systems, Inc., 2014.

P. Muller. Empirical tests of biases in equity portfolio optimization. In S.A. Zenios, editor, *Financial optimization.* Cambridge University Press, 1993.

A. Onatski. Determining the number of factors from empirical distribution of eigenvalues. *The Review of Economics and Statistics*, 92(4):1004–1016, 2010.

R. Roll and S.A. Ross. An empirical investigation of the arbitrage pricing theory. *The Journal of Finance*, 35(5):1073–1103, 1980.

C. Trzcinka. On the number of factors in the arbitrage pricing model. *the Journal of Finance*, 41(2):347–368, 1986.

The Vanguard Group, Inc. Prospectus. Vanguard Total Stock Market ETF. `https://www.vanguard.com`, 2014.

M.E. Wall, A. Rechtsteiner, and L.M. Rocha. Singular value decomposition and principal component analysis. In D.P. Berrar, W. Dubitzky, and M. Granzow, editors, *A practical approach to microarray data analysis.* Springer, 2003.

H. Wang, C. Reeson, and C. Carvalho. Dynamic financial index models: Modeling conditional dependencies via graphs. *Bayesian Analysis*, 6(4):639–664, 2011.

M. West and J. Harrison. *Bayesian forecasting and dynamic models.* Springer Verlag, 1997.

Hydrologic Predictions using Probabilistic Relational Models

Max Metzger
Charles River Analytics.
625 Mt. Auburn St.
Cambridge, MA 02138

Alison O'Connor
Charles River Analytics
625 Mt. Auburn St.
Cambridge, MA 02138

David F. Boutt
University of Massachusetts
611 North Pleasant Street
Amherst, MA 01003

Joe Gorman
Charles River Analytics.
625 Mt. Auburn St.
Cambridge, MA 02138

Abstract

The US Army faces a significant burden in planning sustainment operations. Currently, logistics planners must manually evaluate potential emplacement sites to determine their terrain suitability. Sites subject to rainfall-runoff responses such as flooding are ill-suited for emplacements, but evaluating the likelihood of such responses requires significant time and expertise. To reduce the time and to ease the difficulty of logistics site selection we demonstrated a series of Terrain Impact Decision Extensions (TIDE) for use in logistics planning tools and processes. TIDE performs data-fusion over a variety of terrain and weather data sets using probabilistic relational models (PRMS), providing a high-performance alternative to physics-based hydrologic models.

1. INTRODUCTION

Maintaining a constant supply of water and fuel is critical to sustaining the US Army's forces in the field. (Department of the Army, 2008). To provide access to these critical resources, logistics planners must deliver those resources with minimal failure to establish and maintain emplacements (e.g., tanks, fuel lines) capable of storing these crucial commodities. Water and fuel supplies must not be susceptible to disruption, damage, and contamination by water due to rainfall-runoff responses such as flooding, overland flow, and ponding (i.e., the temporary accumulation of surface water).

Currently, the risks posed by rainfall-runoff responses to potential emplacement sites are manually evaluated, and require considerable expertise and time. Site evaluation is further complicated for areas lacking detailed data that describe terrain, soil properties, and subsurface conditions (e.g., the presence of aquifers). This occurs due to the significant role played by terrain, soil, and subsurface factors in the effects of rainfall run-off on terrain. A decision aid capable of automatically evaluating the suitability of emplacement sites would reduce the time needed for evaluation by logistics planners and improve the quality of sites selected.

To reduce the time and the difficulty of logistics site selection we designed and demonstrated a series of Terrain Impact Decision Extensions (TIDE) for logistics planning tools and processes. TIDE performs data-fusion over a variety of terrain and weather data sets, and uses probabilistic relational models (PRMs) to reason with uncertainty to evaluate the suitability of potential logistics sites against a series of expert rules for a variety of emplacement systems. By using PRMs to rank the severity of potential rainfall-runoff responses, TIDE was able to site determine suitability much faster than by rigorous physical simulation. Additionally, PRMs can reason with incomplete data (e.g., a lack of detailed soil information), making them useful even when evaluating data-poor regions.

1.1 PROBLEM DESCRIPTION

The rainfall-runoff response of landscapes is a fundamental problem in the field of hydrology (Singh, 1988). The accumulation of water at a particular time-space location on the Earth's surface (i.e., terrain ponding) is the result of the confluence of many climatologic, hydrologic, and physical factors and parameters. During a liquid precipitation event (e.g., rain), water is transported in three main ways: water can run-off/on in the form of overland flow; infiltrate into the soil and become ground water; or be transferred back into the atmosphere via evapotranspiration. Overland flow can in turn lead to the accumulation of surface water (e.g., flooding), which poses a risk to US Army emplacements.

The terrain assessment model must account for multiple aspects of the area of interest. First, overall climatic conditions (i.e., arid, semi-arid, humid) have an important influence over the relative distribution of water in the three pathways. The model must account for uncertainty in weather predictions and climatologic predictions. Second, the model must account for local factors within the area of interest that influence the rainfall-runoff response. There are many such factors, including rainfall intensity and duration, slope of the land, land use and land cover characteristics (i.e., vegetation and impervious surfaces), soil and air temperature, soil hydraulic properties, and soil moisture conditions. The data sources for these factors may be incomplete or inaccurate, introducing additional uncertainty.

The prediction of where, when, and how long water will accumulate on the land surface is reliant on constraining parameters that describe the above processes and conditions. Fortunately, hydrologists have been developing tools to both quantify these factors and develop quantitative models for predicting rainfall-runoff response to precipitation events.

These models are often based on solving complex equations that govern the physics of surface and subsurface water (Abbott, Bathurst, Cunge et al., 1986; Panday & Huyakorn, 2004) or assign statistical values to terrain based on observation (Yoram, 2003). These models are not practical for US Army planning because they require complete data sets, are extremely time-consuming to compute, and do not scale to the levels of detail and scope required by US Army logistics planners.

2. APPROACH

Given the potential incompleteness of input parameters (including terrain, soil and subsurface data), our approach uses a probability-based method to track the inferences made about data through the model. For TIDE to be useful, the system must infer terrain characteristics, soil properties, and subsurface conditions from limited data. While terrain elevation data is available for most of the world at varying levels of detail, soil data is less prevalent. Land use, land cover (e.g., vegetation), as well as the soil's hydrologic properties and moisture conditions are all factors in predicting rainfall-runoff response. When this information is not directly available, it needs to be estimated or inferred. For example, soil properties for a given region within the United States may be well-known and stored in a Geographic Information System (GIS) database, but this data may be unknown for many rural regions around the world. An exhaustive geological survey of potential sites within that region is not possible given time and personnel constraints. Even when terrain, soil, and subsurface data are present, it may not be at resolutions high enough to be relevant to the emplacements (e.g., a map with soil data at a resolution of 500m is of limited use when selecting a site for a fuel line less than a meter across). In cases where data describing terrain characteristics, soil properties, and subsurface conditions are absent, purely rule-based approaches are insufficient, as rules alone are poorly-suited to handling incomplete data. The system must be capable of reasoning with limited or incomplete data before executing any impact assessment rules. The PRM model developed under the TIDE effort is capable of reasoning with incomplete data and inferring data that may be absent. Additionally, while our initial model is very simple, further work may expand the model to be very complex. The object-oriented PRM approach is well-suited for such complexity.

The PRM output is used to generate maps showing the likelihood for flow accumulation at a given location for a certain amount of time. We based our models on the Hortonian Infiltration and Runoff/On (HIRO2) model, which was originally developed for the USDA (Meng, Green, Salas et al., 2008). This model predicts rainfall-runoff responses, including runoff channels (in which surface water flows) and the time until ponding occurs. The HIRO2 model performs well, but operates at larger scales than are useful to emplacement selection, generally being most accurate at scales of hundreds of meters. The HIRO2 model served as the basis for our model, but was modified to operate at higher levels of fidelity without significantly compromising performance.

Bayesian modeling techniques have been used in the field of hydrology for decades (Vicens, Rodriguez-Iturbe, Schaake et al., 1975), but the majority of this work has different goals than TIDE. Bayesian modelling approaches generally take existing models that use direct measurements as inputs (e.g., rainfall) and predict specific hydrologic response values (e.g., runoff rate, groundwater level). Bayesian techniques are then used to calibrate the models parameters to improve their accuracy (Beven & Binley, 1992; Thiemann, Trosset, Gupta et al., 2001; Vrugt, Ter Braak, Clark et al., 2008).

TIDE differs from past Bayesian hydrologic models in two fundamental ways. First, our model attempts to predict the impact of rainfall-runoff responses, not their precise values. Generally speaking, US Army logistics planners are not concerned about predicting the exact amount of surface water that may accumulate, but are instead primarily concerned about the impact on the mission. For example, the difference between 1.2 meters of standing water or 2.4 meters is irrelevant if either makes the mission impossible to complete.

Second, the TIDE model must perform with reasonable accuracy in regions of the world that have little, if any, hydrologic data observations (e.g., hourly flow rates for a stream) that can be used to train or calibrate a model. Instrumenting and measuring rainfall-runoff responses in these areas may be too costly, logistically infeasible, or dangerous. As a consequence the TIDE model must rely on generally available data (e.g., elevation, land cover, weather).

2.1 PROBABALISTIC RELATIONAL MODELS

To represent our terrain and hydrologic models in a probabilistic form that allows us to determine the suitability of an area of interest, we designed a probabilistic relational model (PRM) (Koller & Pfeffer, 1998; Pfeffer, Koller, Milch et al., 1999; Friedman, Getoor, Koller et al., 1999). PRMs describe the world in terms of classes of objects, instances of those classes and relationships between them. Serving as a powerful extension of Bayesian Networks (BNs), PRMs use object-oriented semantics that capture attribute, structural, and class uncertainty to overcome computational and storage complexity challenges faced by BNs.

The design of PRMs has proven to be useful in representing a wide range of complex domains that involve uncertainty and require flexibility and reusability. In regard to complexity, PRMs capture the logical and relational structure of a domain. For example, PRMs specify how one attribute influences the value of another attribute. In our PRM, the value of the attribute, RankDrainageCapacity, is dependent on the values of attributes, LandCoverType and SoilType, from two other classes. Therefore, the model uses the values of RankDrainageCapacity's dependent attributes, LandCoverType and SoilType, to infer the value of RankDrainageCapacity.

To handle uncertainty, PRMs use probability distributions encoded in the model to determine values of unknown variables. The value of LandCoverType and SoilType for a location are retrieved from data sources outside the model and then posted to the model. Therefore, there is little uncertainty in regard to these two attributes. Conversely, the value of RankDrainageCapacity is inferred inside the model using probability distributions. To overcome the uncertainty involved with this attribute, encoded in the model is a map of possible combinations of land cover and soil types to appropriate probability distributions. Relying on the team's hydrologic expertise, we created initial distributions for each possible pair of land cover and soil types, as well as each land cover and soil type provided the other attribute was unknown. Similarly, using domain knowledge, we supplied distributions for each individual slope ranking, flow ranking, and drainage ranking assuming that the other two attributes were unknown. Given the increased number of combinations for RankRunoffPotential, to obtain distributions in the case that two or three attributes were known, we multiplied the probabilities of the known attributes for each of the five possible RankRunoffPotential values. The distributions for Suitability were much simpler to encode, as only five probability distributions that required no further calculations were necessary. While these initial distributions pass face validation, future work is needed to adjust the distributions to meet higher accuracy needs.

To support flexibility and reusability, PRMs allow the reuse of the same class probability models for all instances of a class. New probabilistic models do not have to be constructed for each new situation. Instances of classes can be configured in any way desired for a given situation. The relationships that hold between these instances are captured by the PRM. For example, our PRM contains a class SiteModel that has one attribute, Suitability. The value of Suitability depends on the instance of the Runoff class' attribute, RankRunoffPotential. To reason over this model, one must create instances of both the SiteModel and Runoff classes.

The flexibility and reusability of PRMs grant us the ability to reason over millions of locations. For each location, the relevant set of known facts about specific attributes – the land cover type, soil type, rank of slope and rank of flow –must be provided to the instances of classes. As we transition to discuss our PRM in greater detail, it will become more evident that these four key features of PRMs – complexity, uncertainty, flexibility, and reusability – are crucial to obtaining successful results. Bayesian Networks could also apply to this problem, as the relational structure is fixed for every instance. Nevertheless, the object-oriented representation of PRMs were quite helpful in designing the model.

2.2 PRM EDITOR

We developed a PRM Editor that provides an intuitive graphical user interface (GUI) that allows users to create complex PRMs by defining classes of objects, adding attributes to those class definitions, creating instances of the classes, and specifying the relationships between them.

Upon launching the PRM Editor, the user can navigate between three views: the global view, class view, and instance view. While these three views are initially blank, the panels become populated with information and graphical representations of the model. The global view allows a user to view the PRM as a whole in a folder format. Its top-level folder, named after the PRM, can be expanded to display three other folders, enums, classes, and instances. The enums and instances folders can further be expanded to show all enumerations and instances of classes in the model. Within the classes folder are additional folders for each class that can be expanded to view the attributes in that class.

Unlike the global view, the class view displays a graphical representation of the PRM. Each class is represented by a box labeled with the class name. If applicable, arrows are automatically drawn between boxes indicating super and subclasses (parent-child relationships). The instance view also displays boxes that, rather than represent classes, represent the instances of classes in the model.

To begin utilizing these three views, the user has the option to either load an existing model into the GUI or create a new PRM. After loading or initializing the model, the user can begin building the model by adding classes. When creating a class, the user must specify the name and parent class of the new class. In the case of our model, we created six classes, none of which had parents, so this field remained blank.

Adding an attribute requires more detail than adding a class. A user must specify the attribute name, type, and resolution. The user has the option to assign single or multi as the attribute's type, as well as choose from a list of possible types. Possible types contained in this list include integer, real number, Boolean, type, nothing, as well as all of the classes and enumerations created by the user. If the attribute is of type enumeration, the user must have previously defined the appropriate enumeration. For example, in our model, the SiteModel class' attribute, Suitability, is of type enumeration. The possible values for Suitability are VeryPoor, Poor, Medium, Well and VeryWell. Therefore, we created an enumeration called RankPoor, to represent an attribute with these five possible values. Before defining an attribute's resolution, the user must create instances of other classes. By creating instances of classes, attributes in other classes can depend on the attributes of these instances. Figure 1 shows the global and class relationship views after the six classes and their six respective instances have been created.

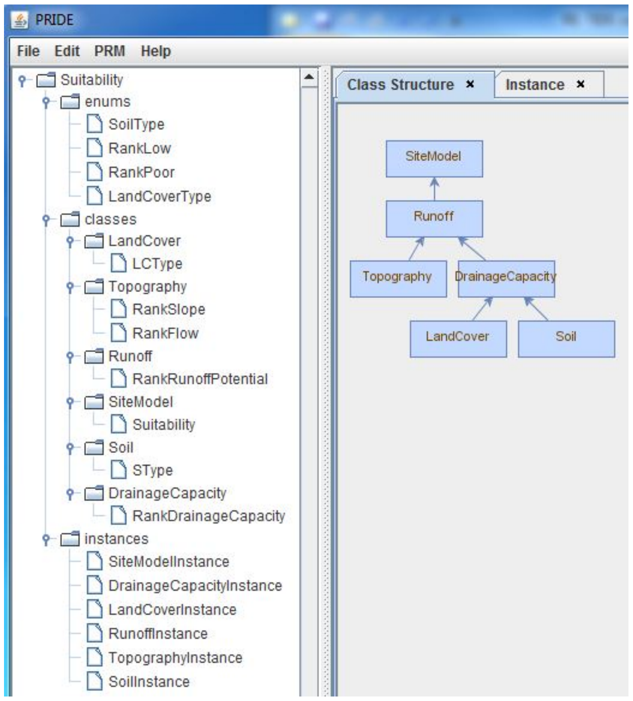


Figure 1: Global View (left), Class Relationship View (right)

To complete the implementation of the previously discussed attribute, Suitability, its resolution must be defined. The resolution of an attribute can be assigned as nothing, assignment, or dependency. Upon creating the attribute, the default choice is nothing. If the user updates the resolution to assignment, the user must enter the exact value of the attribute or its reference. For example, if the attribute were an integer, the user could indicate that value was 10. Alternatively, the reference could be set to an attribute of another class that was also an integer. The appropriate resolution for Suitability is dependency. Therefore, the user must specify the influencer, the attribute that Suitability depends on, as well as the conditions and their respective distributions. The conditions are the possible values of the influencer. Each possible value of the influencer is paired with a CPD indicating the likelihood of each possible value of the attribute. Suitability depends on the instance of the Runoff class', RunoffInstance, attribute RankRunoffPotential. RankRunoffPotential has five possible values – VeryLow, Low, Medium, High, and VeryHigh. Therefore, Suitability will have five conditions and five distributions that indicate the probability of each of Suitability's five possible values occurring given the value of RankRunoffPotential.

Having defined four enumerations, six classes, seven attributes, and six instances in our model using the PRM Editor, the model was saved to as a .prm file that could be used by the TIDE system.

2.3 HYRDOLOGIC MODEL

A PRM consists of a set of class probability models. The final version of our PRM (Figure 2) contains six classes – SiteModel, Runoff, Topography, DrainageCapacity, LandCover, and Soil. Each class has a set of attributes. Attributes are either simple or complex. Simple attributes are random variables that represent direct properties of an object, such as the type of land cover or type of soil, whereas complex attributes represent relationships to other objects. The attributes in our model are all simple.

Logical relationships can be described between classes. The lines in Figure 2 represent these relationships. Assuming we have an instance of every class, an instance of LandCover is related to an instance of the DrainageCapacity class by the LandCoverType attribute.

Each simple attribute is associated with a set of parents and a CPD. The parents are determined by the attributes that the attribute depends on. Attributes can depend on either other simple attributes of the same object or of related objects. An example of an attribute of an object depending on an attribute of a related object is the dependence of the RankDrainageCapacity on LandCoverType.

Attributes of related objects are specified via attribute slot chains, such as the slot chain LandCoverInstance LandCoverType. This slot chain begins with the object representing the land cover of a location, and accesses the simple attribute indicating the type of land cover at this location. The model specifies that the RankDrainageCapacity attribute of the DrainageCapacity class has this slot chain as a parent. To reiterate, this indicates that the RankDrainageCapacity depends probabilistically on the LandCoverType. The other slot chain parent of RankDrainageCapacity is SoilInstance.SoilType. It is important to emphasize that these parent relationships are general, meaning that the land cover or soil type may vary from scenario to scenario, but the probabilistic relationships hold for all scenarios (e.g., when a new area is investigated).

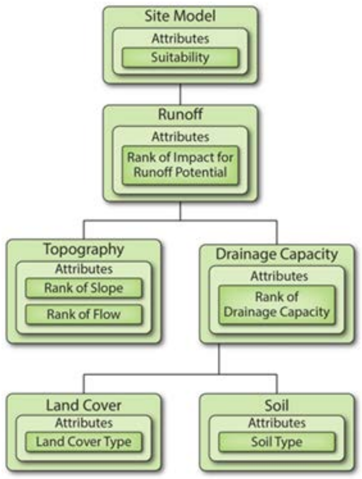


Figure 2: Hydrologic PRM

In our tree-structured PRM, the attributes in the classes directly below another class are parents to the attributes in the class above them. Therefore, the attributes in the three leaf classes, LandCover, Soil and Topography, do not have parents. The values of these attributes are derived outside the model and posted as evidence to the model. Conversely, the values of the attributes in the remaining classes, SiteModel, Runoff, and DrainageCapacity, are inferred from the data available within the model.

Recall that the other information associated with a simple attribute is a CPD that specifies a distribution over values of an attribute given the values of its parents. In the case of Suitability, its parent is Runoff.RankRunoffPotential. Table 1 shows the code for the implementation of the SiteModel class, complete with its attribute, Suitability, specification of its parent, RankRunoffPotential, and CPD for every possible value of RankRunoffPotential. The bolded line specifies, in plain terms, that if the RankRunoffPotential is VeryLow then there is 10% likelihood Suitability is VeryPoor, 15% likelihood Suitability is Poor, 50% likelihood Suitability is Medium, 15% likelihood Suitability is Well and 10% likelihood Suitability is VeryWell.

Similar to how parent relationships are defined, the assigned CPD is general; it holds no matter what the specific related objects are.

Table 1: SiteModel Class Implementation

```
class SiteModel =  {
     Suitability:       single       RankPoor       depends       on
[RunoffInstance.RankRunoffPotential]
     case [VeryLow] =>
           (0.1 -> VeryPoor, 0.15 -> Poor, 0.5 -> Med, 0.15 -> Well,
0.1 -> VeryWell)
     case [Low] =>
           (0.7 -> VeryPoor, 0.1 -> Poor, 0.1 -> Med, 0.05 -> Well,
0.05 -> VeryWell)
     case [Med] =>
           (0.05 -> VeryPoor, 0.1 -> Poor, 0.7 -> Med, 0.1 -> Well,
0.05 -> VeryWell)
     case [High] =>
           (0.05 -> VeryPoor, 0.05 -> Poor, 0.1 -> Med, 0.7 -> Well,
0.1 -> VeryWell)
     case [VeryHigh] =>
           (0.05 -> VeryPoor, 0.05 -> Poor, 0.1 -> Med, 0.1 -> Well,
0.7 -> VeryWell)
     case [_] =>
           (0.2 -> VeryPoor, 0.2 -> Poor, 0.2 -> Med, 0.2 -> Well, 0.2
-> VeryWell)
             }
```

With a clear understanding of how relationships and CPDs are specified in the model, we can discuss how inference ultimately determines if a location is suitable. The basic order of how the model performs inference is: once the values of an attribute's parents are known, the value of that attribute can be inferred. Therefore, the process begins by posting evidence to the leaf classes. First, the LandCoverType, SoilType, RankSlope, and RankFlow evidence is posted to the model. Next, the model can infer the value of RankDrainageCapacity from the land cover and soil data. For example, if the LandCoverType is Shrub and the SoilType is Vertisols, the probability distribution encoded in the model for RankDrainageCapacity given this evidence is: case [Shrub,Vertisols] => (0.7 -> VeryPoor, 0.3 -> Poor, 0.0 -> Med, 0.0 -> Well, 0.0 -> VeryWell). Again, this distribution can be interpreted as: If LandCoverType is Shrub and SoilType is Vertisols, there is 70% likelihood the RankDrainageCapacity is VeryPoor and 30% likelihood the rank of drainage capacity is Poor. Once the CPD of RankDrainageCapacity is determined, the distribution can be used in conjunction with the Topography evidence to infer the value of the RankRunoffPotential. This process propagates up the model, as RankRunoffPotential influences the value of Suitability.

To determine a site's suitability, the model uses all available data. While accuracy increases with amount of available data, our model is capable of reasoning with incomplete or no data. In the case that data is unavailable or unknown for the four inputs – LandCoverType, SoilType, RankSlope, and RankFlow – the probability is evenly distributed over all possible values.

Our PRM Editor utilizes the open source Figaro probabilistic programming language (PPL) (www.cra.com/figaro) to perform inference. PPLs provide a powerful and flexible way to represent probabilistic models using the power of programming languages. In addition, PPLs offer general-purpose reasoning algorithms for inference and machine learning. Our implementation utilizes the Metropolis-Hastings reasoning algorithm, capped with a runtime of 5,000 milliseconds per inference.

2.4 INTEGRATION

Our PRM used data from the following data sources.

2.4.1 SRTMVF2

The Shuttle Radar Topography Mission (SRTM) was a joint project between NASA and the National Geospatial-Intelligence Agency (NGA) to create high-resolution land surface data for much of the world (roughly 80% of the Earth's land surface is covered). The SRTM Void-Filled 2 (SRTMVF2) data set is at 1-arc-second (approximately 30-meter) resolution data, with many gaps in data void-filled using interpolation techniques (Dowding, Kuuskivi, Li et al., 2004). The SRTMVF2 dataset serves as our primary elevation data source, as our hydrologic model is heavily dependent on accurate, high-resolution elevation data. However, we have identified that there are gaps within the SRTMVF2 elevation data. In areas where no SRTMVF2 data can be found, we can fall back to lower resolution DTED data, including the SRTMVF1 and SRTMVF0 data sets. Elevation is used to determine inputs to our model, slope and water flow. Slope and flow implicitly capture the spatial relationships of each DTED point with its neighbors, allowing the PRM to reason about each point's data independently.

Slope

Slope is determined using the elevation dataset. For the initial effort, we used a simple algorithm that iterates across each elevation point. For each point, the relative change, dE, in elevation is calculated for each adjacent point (excluding diagonally adjacent points.) The dE value with the greatest magnitude is selected, and the distance between points (1 arc-second in the case of the SRTMVF2 dataset) is used to calculate the angle of the terrain's surface, Θ. This value can range from 0 degrees (i.e., perfectly flat) to 90 degrees (which would be a perfectly vertical surface.) While there are more elaborate methods for determining slope that provide more accurate results, this technique can process millions of points in a matter of minutes, and yields sufficient accuracy for the needs of the terrain assessment model.

Once the slope angles have been calculated using the algorithm described above, they are translated from a continuum of [0, 90) to five discrete values, which are used as inputs for the terrain model. Table 2 shows how angle ranges are mapped to model inputs.

Table 2: Mapping terrain slope angles to model inputs

Angle Range	Rank of Slope
$0 \leq \theta \leq 10$	Very Low
$10 < \theta \leq 20$	Low
$20 < \theta \leq 30$	Medium
$30 < \theta \leq 60$	High
$\theta > 60$	Very High

Flow

The elevation dataset is used to predict flow channels – that is, paths that surface water is likely to take in the event of rainfall. A greater amount of flow indicates a risk of surface water accumulation. To calculate flow, we relied on the TopoToolbox (Schwanghart & Kuhn, 2010). The toolbox includes techniques for predicting flow estimation. The flow values predicted can vary wildly. In the case of our AOI, estimated flow varied between 0 and over 3,300. To normalize the dataset, we first transformed the flows to a logarithmic scale (changing the range from 1 to ~9.7) and then normalized the results to [0, 1].

Table 3: Mapping Flow to Model Inputs

Flow Range	Rank of Flow
flow is exactly 0	Very Low
$0 < flow \leq 0.1$	Low
$0.1 < flow \leq 0.2$	Medium
$0.2 < flow \leq 0.5$	High
$flow > 0.5$	Very High

As shown in Table 3, these values are then translated into five discrete inputs for the terrain assessment model (same as the slope). As with the slope values, the process of mapping flows to discrete ranking values is independent from the flow calculations. This means that calculating the flows (a process that took roughly two hours for the Demonstration Scenario's AOI) need only be run once per AOI, even if we adjust model values or how flow values are mapped to model inputs.

2.4.2 GeoCover

Earth Satellite Corporation (EarthSat) developed the GeoGover data set, a global landcover database. The GeoCover dataset consists of 13 land cover classes and is available for much of the world (Cunningham, Melican, Wemmelmann et al., 2002). Classes of land cover include grasslands, agriculture areas (i.e., farmland), wetlands, and water/ice. This data will serve as additional inputs to our terrain models so we can more accurately assess rainfall-runoff response. The GeoCover dataset will also enable TIDE to identify bodies of water.

2.4.3 Harmonized World Soil Database

The Harmonized World Soil Database (HWSD) was produced by the European Union's European Commission Joint Research Centre (more specifically, the Land Management Unit of the Institute for Environment and Sustainability.) The HWSD is a 30 arc-second (approximately 90-meter) resolution that contains detailed information about the top soil and subsoil properties. It was created by merging data from four different soil databases (Nachtergaele, Van Velthuizen, Verelst et al., 2008).

This data allows the model to more accurately predict how terrain will respond to surface water (for example, how quickly water will be absorbed into the soil.) This dataset's low resolution means that some terrain boundaries (such as coasts) and geographical features (such as bodies of water) are of low accuracy compared to the other data sets.

2.5 MISSION DECISION RULES

The system must provide a set of logistic system-specific terrain assessment rules for a variety of systems and purposes (e.g., Tactical Water Distribution System, Assault Hose Line System). Terrain suitability may vary from system to system—for example, a suspended hose may be unaffected by some types of standing water while a ground-level hose could be at risk for contamination. Rule sets for individual systems will need to account for these differences, allowing planners to choose the appropriate system given the characteristics of a prospective emplacement site. Additionally, logistics planners must be able to easily modify and expand these rules as new systems are introduced, and as mission requirements change. (For example, different rules would be used for route planning than well placement.)

In our initial effort, we have implemented some basic rules that filter terrain suitability for a hypothetical fuel line. The fuel line has two requirements: (1) it must be installed on flat land (so the pumps can function properly); and (2) the fuel lines cannot be placed in standing water (to prevent contamination), which includes bodies of water (such as lakes) and areas that are prone to flooding.

To determine suitability for the fuel lines, we take slope, land coverage, and hydrologic suitability as inputs. We then apply a set of rules as described in Table 4. The rules transform the hydrologic suitability into mission suitability. These rules favor flat land over sloped land.

Table 4: Mission Rules

Condition	**Effect**
If the point is a body of water	Mission suitability is Very Poor
If slope is ranked as "Low" or "Very Low" and hydrologic suitability is "Medium"	Mission suitability is High
If slope is ranked as "Low" or "Very Low" and hydrologic suitability is *not* "Medium"	Mission suitability is equal to hydrologic suitability
If slope is ranked as "Medium" and hydrologic suitability is "High" or "Very High"	Mission suitability is Medium
If slope is ranked as "Medium" and hydrologic suitability is *not* "High" or "Very High"	Mission suitability is equal to hydrologic suitability
If slope is "High" or "Very High"	Mission suitability is Very Poor

Currently, rules are distinct from the PRM model, so that custom rules can be written for different operational needs while using the same PRM model. For example, while the PRM output is constant, the mission requirements for a long fuel pipeline may be very different than the mission requirements for a convoy. The fuel line would have a very low tolerance for changes in elevation (as the pumps cannot handle the increased workload) and would be susceptible to contamination from standing water. The convoy, while still limited by severe terrain or flooding, would be much more resilient to water and slopes.

A standard rules engine and associated rules language, such as that provided by JBoss, would allow mission experts to author rules for the TIDE system without requiring them to understand the PRM or hydrology.

2.6 VISUALIZATION

Outputs of the PRM and the rules engine, as well as the data sources themselves, were rendered within NASA WorldWind. WorldWind can accept a variety of GIS data formats and is easily customizable. Using the open-source the Geospatial Data Abstraction Library (GDAL), we wrote custom modules to render the HWSD and GeoCover data sets, while the SRTMVF2 data was loaded in using built-in WorldWind methods. Model output and rules output were rendered as textures which were then projected onto the WorldWind globe at the appropriate coordinates, but the data could easily be written to a variety of GIS formats.

The hydrologic suitability is presented as belief values in five categories: {Very Poor, Poor, Medium, High, Very High}. Related military impact assessment (e.g., weather impact assessment) is done at three intervals (e.g., low risk, medium risk, and high risk.) We expanded our model to use five intervals instead of three to present additional granularity in the model's output. Further work is needed to determine the best number of intervals and their thresholds.

For the initial effort, the category with the highest belief value is selected as the 'correct' suitability value. These categories are then color-mapped for visualization: {Red, Orange, Yellow, Green, Blue}. The same categories and colors are used for the rules output.

3. DISCUSSION

For our area of interest and the 1 arc-second SRTMFV2 set, there are 25,934,402 points to process. Executing the entire PRM for each point would be unnecessarily complex – instead, we store each unique combination of {soil type, land cover, rank slope, rank flow} and store the associated beliefs. This means we can simply look up the correct PRM output for each unique combination of inputs, which need only be run through the PRM once. As a result, we are able to process all 25.9 million points in only two hours. (Further updates to the Figaro library should increase runtime performance as well.) In a full-scale TIDE system, the PRM values for all combinations could be calculated once and only once, and then stored in a database for quick reference. This database would only need to be updated when the PRM is updated.

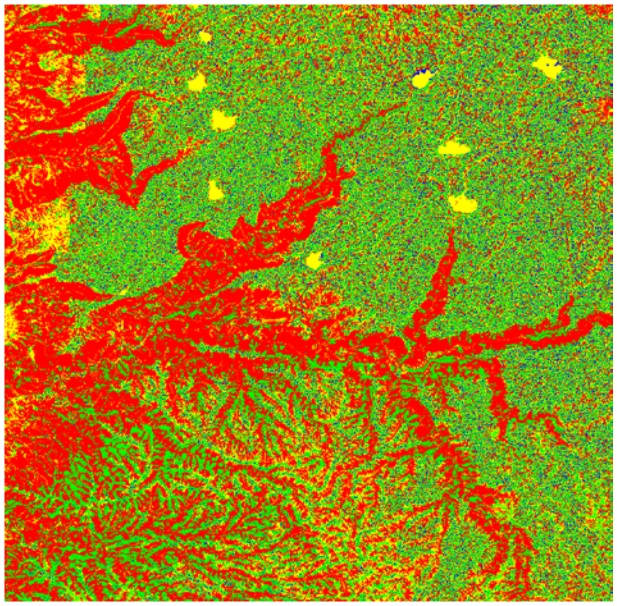


Figure 3: PRM Model Output

Figure 3 shows the output of the model. (Figures 3 and 4 are best viewed in color.) The output of the model is very grainy as each point in the elevation set can have a distinct rank. Of note are the red regions running across the central region of the image – these are riverbeds and their surrounding valleys, which were detected despite those bodies of water not being explicitly present within our GeoCover or HWSD data sets.

Figure 4 shows the output of our rules engine (and a region slightly larger than the figure above). While these rules are very simple, they demonstrate how rules can transform the high-density output of the models (Figure 3, above). The model output scores each point in the elevation grid (approximately a 30 by 30 meter square when using the SRTMVF2 data set), producing a very dense output. Rules can be used to simplify the models' output into easier-to-interpret regions. With these simple rules, we were able to execute rules across the entire region in five minutes. Figure 4 shows the same riverbeds as in Figure 3, but the view is expanded to show a large lake to the west, which has been appropriately flagged as having very poor mission suitability. Unlike the riverbeds (which were predicted by the PRM), this body of water is present within both the GeoCover and HWSD data sets (at differing levels of precision).

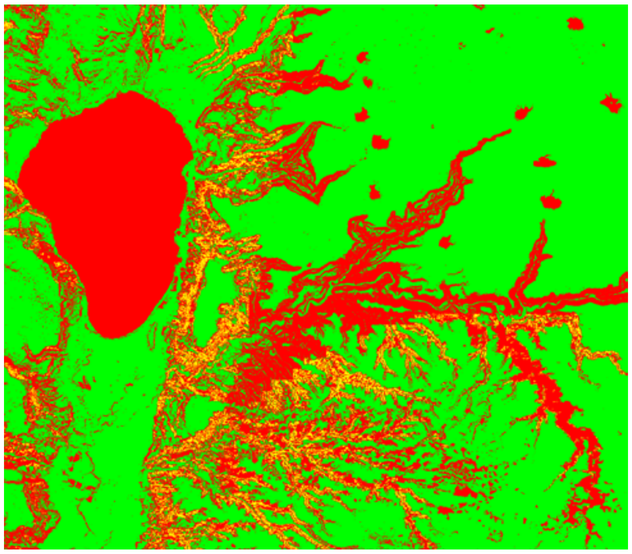


Figure 4: Rules Output

4. FUTURE WORK

4.1 TERRAIN AND HYDROLOGIC MODELS

Future improvements to the model begin by incorporating more data. The more information captured by the model, the more accurate the inferences will be. The next data source to integrate is precipitation data. Depending on the duration of the mission, weather or climate data would be used. For example, missions spanning from zero to four months would heavily rely on weather information, missions spanning from four to eight months would integrate both weather and climate data, and missions lasting longer than eight months would incorporate climate data. We will also work to quantitatively evaluate the performance and applicability of our models.

Our approach to testing and verifying the accuracy of our models is two-fold. First, we will compare the outputs of our models to those of existing, alternative hydrologic models. These models are often based on solving complex equations that govern the physics of surface and subsurface water (Abbott et al., 1986; Panday & Huyakorn, 2004) or assign statistical values to terrain based on observation (Yoram, 2003). These models are not practical for US Army planning because they require complete data sets, are extremely time-consuming to compute, and do not scale to the levels of detail and scope required by US Army logistics planners. However, their outputs have been validated when tested on carefully monitored and measured regions of terrain, typically within the US. By running the TIDE models on the same regions and comparing its output to that of the established models, we can confirm that the TIDE models are functioning correctly.

Second, we will gather existing data sources of rainfall-runoff responses. Several regions within the United States have had their rainfall-runoff responses measured at various degrees of fidelity. For example, the Leaf River basin in Mississippi has over forty years of time series data that includes precipitation and runoff (Yapo, Gupta, Sorooshian et al., 1996). Additional data sources could be built from flood records and high water level records. These data sets will serve to validate the PRM models used by TIDE. They may also serve as training data to calibrate the model to more accurately predict the severity of rainfall run-off responses (e.g., flooding).

4.2 DATA FUSION MODEL

Our basic solution for handling cases of limited or missing data assumes that each value is equally likely if no evidence is posted to the model. Under this assumption, the accuracy of our inferences declines with limited or no data. The inferences are only as strong as the data known and evidence provided.

Future improvements for how to reason with incomplete or no data involve adjusting the prior distributions. Although the prior distributions in our current model assume that all values of an attribute are equally likely if no data is available, one would argue this is not representative of the real world. We plan to explore the possibilities of more representative prior distributions. For example, the prior distribution for land cover type could reflect that fact that over 70% of the earth's surface is covered in water, making it the most likely of the seven values.

This being said, the most dramatic mitigation of consequences due to incomplete data or unknown values will result from future improvements to the model itself rather than the dependencies. As we integrate more data sources into the model, the number of attributes and dependencies will increase, resulting in more accurate inferences. Existing data can also be used to infer missing data. For example, using higher-resolution data (such as elevation data or land cover data) we can easily determine that the HSWD fails to cover the coastlines. We can then predict the missing values using spatial relationships. Ambiguous areas could be assigned multiple values with different confidence values. Figure 5 shows how the two HSWD regions could be used to infer the values for the missing regions.

Point A, to the north, would be assigned a high probability of having luvisols as the dominate soil type. Point B would be assigned near equal probabilities of being either luvisols or vertisols. Point C, to the south, would be assigned a high probability of vertisols as the dominate soil type. The inference used for point B could be assigned to any region near the boundaries of low-resolution data sets – for example, point D could also be assigned a probability of being either vertisols or luvisols; even though the data set classifies it as vertisols, the resolution is low enough that the point could be a misclassification. The assigned probabilities, along with the soil types themselves, would serve as inputs to the PRM models.. For example, the soil type input to our PRMs for Point D could be "{Vertisols-50%, Luvisols-50%} instead of simply {Vertisols}.

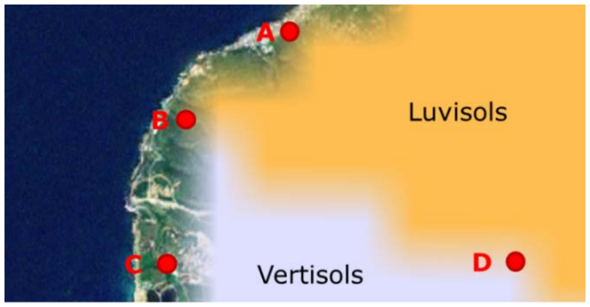


Figure 5: Reasoning about incomplete data

5. CONCLUSIONS

Flooding, and other terrain rainfall-runoff responses, pose significant risk and cost to US Army operations. Assessing the magnitude of flood risk and the impact it will have on a mission requires both time and expertise that may not always be available. An automated system for predicting the likelihood and impact of flooding and surface water accumulation would be of great benefit to logistics planners and the US Army at large.

During our initial effort, we demonstrated the feasibility of Terrain Impact Decision Extensions to predict rainfall-runoff response. We have identified key data sources required for predicting flooding and have developed an initial set of models that are capable of identifying regions that are at high risk of flooding. These models are capable of processing millions of data points per hour, allowing them to process thousands of square kilometers. We feel these models and their performance indicate our approach is sound, and future work will refine and validate the models' performance.

Acknowledgements

This work was performed under US Army Research Lab contract number W911QX-13-C-0111. The authors would like to thank Mr. Peter Grazaitis for his significant technical support and eager engagement on this project. This work was funded in its entirety by ARL. We would also like to thank Ms. Yvonne Fuller and Ms. Jill Oliver for their assistance in preparing this paper.

References:

Abbott, M. B., Bathurst, J. C., Cunge, J. A., O'Connell, P. E., and Rasmussen, J. (1986). An introduction to the European Hydrological System—Systeme Hydrologique Europeen,"SHE", 1: History and philosophy of a physically-based, distributed modelling system. *Journal of hydrology*, 87, 45-59.

Beven, K.& Binley, A. (1992). The future of distributed models: model calibration and uncertainty prediction. *Hydrological processes*, 6, 279-298.

Cunningham, D., Melican, J., Wemmelmann, E., and Jones, T. (2002). GeoCover LC-A moderate resolution global land cover database. In *ESRI International User Conference*.

Dowding, S., Kuuskivi, T., and Li, X. (2004). Void fill of SRTM elevation data–principles, processes and performance. In *Images to Decisions: Remote Sensing Foundations for GIS Applications, ASPRS, Fall Conf., Sep*, 12-16.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). Learning probabilistic relational models. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*.

Koller, D.& Pfeffer, A. (1998). Probabilistic frame-based systems. In *Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 580-587.

Meng, H., Green, T. R., Salas, J. D., and Ahuja, L. R. (2008). Development and testing of a terrain-based hydrologic model for spatial Hortonian Infiltration and Runoff/On. *Environmental Modelling & Software*, 23, 794-812.

Nachtergaele, F., Van Velthuizen, H., Verelst, L., Batjes, N., Dijkshoorn, K., Van Engelen, V., Fischer, G., Jones, A., Montanarella, L., and Petri, M. (2008). Harmonized world soil database. *Food and Agriculture Organization of the United Nations*.

Panday, S.& Huyakorn, P. S. (2004). A fully coupled physically-based spatially-distributed model for evaluating surface/subsurface flow. *Advances in water Resources*, 27, 361-382.

Pfeffer, A., Koller, D., Milch, B., and Takusagawa, K. T. (1999). SPOOK: A system for probabilistic object-oriented knowledge expression. In *14th Annual Conference on Uncertainty in AI (UAI)*.

Schwanghart, W.& Kuhn, N. J. (2010). TopoToolbox: A set of Matlab functions for topographic analysis. *Environmental Modelling & Software*, 25, 770-781.

Singh, V. P. (1988). Hydrologic systems. Volume I: Rainfall-runoff modeling. *Prentice Hall, Englewood Cliffs New Jersey. 1988. 480*.

Thiemann, M., Trosset, M., Gupta, H., and Sorooshian, S. (2001). Bayesian recursive parameter estimation for hydrologic models. *Water Resources Research*, 37, 2521-2535.

Vicens, G. J., Rodriguez-Iturbe, I., and Schaake, J. C. (1975). A Bayesian framework for the use of regional information in hydrology. *Water Resources Research*, 11, 405-414.

Vrugt, J. A., Ter Braak, C. J., Clark, M. P., Hyman, J. M., and Robinson, B. A. (2008). Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with Markov chain Monte Carlo simulation. *Water Resources Research*, 44.

Yapo, P. O., Gupta, H. V., and Sorooshian, S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. *Journal of Hydrology*, 181, 23-48.

Yoram, R. (2003). *Applied stochastic hydrogeology.*: Oxford University Press.

Semantics for Reducing Complexity and Improving Accuracy in Model Creation Using Bayesian Network Decision Tools

Oscar Kipersztok
Boeing Research & Technology
P.O.Box 3707, MC: 4C-77
Seattle, WA 98124
oscar.kipersztok@boeing.com

Abstract

The work presented simplifies and makes accessible the process of using advanced probabilistic models to reason about complex scenarios without the need for advanced training. More specifically, it greatly simplifies the effort involved in building Bayesian Networks for making probabilistic predictions in complex domains. These methods typically require trained users with a sophisticated understanding of how to build and use these networks to predict future events. It entails the creation of simplified semantics that keeps the complexity of the methodology transparent to users. We provide more precise semantics to the definition of concept variables in the domain model, as well as using those semantics to assign more precise and robust meaning to predicted outcomes. This work is presented in the context of a tool and methodology, called *DecAid*, where complex cognitive models are created by defining domain-specific concepts using free language and defining relations and causal weights between them. In response to a user query the *DecAid*, unconstrained, directed graph is converted into a Bayesian network to enable predictions of events and trends.

1 INTRODUCTION

DecAid is a hypothesis-driven decision support tool that facilitates complex strategic decisions with features that allow for easy, fast, knowledge capture and modeling in complex domains. It identifies the key variables relevant to a specific query. While the cognitive, unconstrained, model is built, the defined concepts are used to create a probabilistic model to forecast events and trends. Similarly, the free-language used to define and label the concepts is used to generate a document search classifier to retrieve evidence for validation of hypotheses raised by the predictive model. *DecAid*'s goal is to predict likelihood, impact and timing of events and trends (Kipersztok, 2004).

DecAid is aimed at strategic decision making where the risk of making the wrong decision can be very costly and where there is need for argumentative rigor and careful documentation of ideas, associations and assumptions leading to the final decision. The modeling methodology was created to enable domain experts to create Bayesian networks (BN) without having to familiarize with the theory of graphical probabilistic networks or the practice of how to build them. Such users may not also require the involvement of a knowledge engineer. At the levels where high impact decisions are made, requiring high-level of abstraction and dealing with large number of variables and interdependencies, it is less likely that decision makers will use advanced decision analytic tools requiring learning specialized methodology to define and represent complex domain knowledge. The overall goals and requirements identified for the development of the *DecAid* tool were described in (Kipersztok, 2007).

In a world of rapid change it is incresingly challenging to stay abreast of occurring events and trends, making it more difficult to process information without the use of advanced technology tools designed to manage complexity and large volumes of information. Furthermore, strategic decision makers recognize the need for argumentative explanations to strategic decisions that capture the hypothetical reasoning and the evidential context behind each decision. For these reasons the need arises to rely on advanced methods to gather, organize, process and analyze data and knowledge.

Bayesian networks practitioners recognize the need to make the technology more accessible to end users due to the challenges presented during the model creation process. Some of the most significant challenges that *DecAid* aims to address are: 1) the complexity in eliciting expert knowledge, 2) defining a, potentially, large number of parameters and relations in a particular domain, 3) adhering to conditional independence constraint in the definition of causal variables, and 4) requiring to avoid feedback reasoning during model creation that may result in graphs with cycles.

The first challenge has been addressed by various software packages (e.g., Netica, GeNIe, Hugin, etc.) that enable users to build BN with user-friendly interfaces equipped with knowledge elicitation tools. Learning algorithms have also provided the means for automated construction of BN structures and their parameters from data. To address the second challenge, canonical structures have been defined that reduce the number of parameters needed to construct conditional probability tables (CPT). (Farry et al, 2008) review several canonical models, including Influence Networks (Rose and Smith, 1996), Noisy-OR, Noisy-MAX, Qualitative Probabilistic Networks (QPN) and Causal Influence Models (CIM). They, in particular, emphasize usability of CIM models where the causal influence of each parent is captured by a single number and the combined influence of all parents is the mean of the individual parent values. (Pfautz et al, 2007) address the first three challenges and describe additional ones in findings from in-depth analyses of their experience in facilitation of model construction from numerous projects.

The purpose of this work is to describe formal semantics that enable *DecAid* to be directly accessible to domain experts to create BN models without having to concern themselves with these challenges. These semantics are aimed at easing the constraints imposed by the aforementioned challenges by enabling users to define concepts and their relations in free-association mode. Concepts are defined and labeled using free language and a single numerical weight is assigned to each parent-child relation. This effort results in the creation of the *DecAid* (unconstrained) network (DN), a directed graph, which allows cycles. The step of creating a BN from the DN starts with a query definition, and it involves the identification of the query-specific sub graph and removal of its cycles by, optimally, minimizing the information loss. The result is BN directed acyclic graph specific to the query.

2 FROM DECAID NETWORKS TO BAYESIAN NETWORKS

DecAid is a system for simple but powerful probabilistic modeling of arbitrary scenarios. It enables domain expert to create *DecAid* networks by defining *concepts* with free language and causal *relations* between them. For each pair of relations, the user assigns a *weight of causal belief*. There are two types of concepts: a) *Event* concepts that represent quantities that can occur or not-occur; and b) *Trend* concepts that represent quantities that *increase*, remain *unchanged*, or *decrease*. Various levels of granularity can be selected to define the *trend* concept states.

In this section we describe the formal definitions that enable the creation of a DN and its subsequent conversion into a BN.

2.1 Definition of a *DecAid* Network (DN)

Similar to a Bayesian network, each *DecAid* variable (DV) represents a concept, which is some aspect of the domain modeled. More specifically, a DV defines a probability distribution over its possible values and it is discrete—i.e., finite-valued and typically taking 2, 3, 5, or 7 values. For example, we might have a DV named 'Barometric Pressure' that has 3 values: 'decreasing', 'unchanged', and 'increasing'. The set of values is taken to have some natural ordering so that we can speak of high values versus low values. If the variable is binary, we would say that values such as false / off / does-not-occur would be "low" compared to true / on / occurs.

More formally, a *DecAid* model **M** includes a set **V** of DVs and, taken together, the variables in **V** jointly describe a distribution over the entire scenario modeled by **M**. Along with the set **V**, the model **M** includes a directed graph structure **G** connecting the variables of **V**. Each variable in **V** is a node of **G** and each arc denotes a direct probabilistic influence of the parent's value on the distribution over the child's values. The directed graph **G** is *unconstrained*—all connections are allowed and cycles are permitted. Each arc is labeled with a single real number between −1 and 1 called the *weight*. Intuitively, the closer $|w|$ is to 1, the stronger the influence of the parent over the child and the closer $|w|$ is to 0, the weaker the influence. If the weight is positive, a high parent value makes high child values more likely and a low parent value make low child values more likely. A negative weight flips the influence so that a high parent value makes low child values more likely and a low parent value makes high child values more likely (other things being equal). Note that a moderate parent value will make moderate child values more likely.

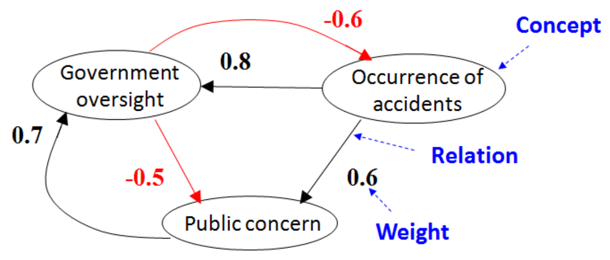


Figure 2.1: A *DecAid* Unconstrained Model

Figure 2.1 shows an example of an unconstrained DN representing three concept variables in the Aviation Safety domain and five parent-child relations with their corresponding weights.

Once, the unconstrained model is built, *DecAid* is capable of transforming the DN into a BN in order to make predictions in response to queries.

2.2 Transforming a *DecAid* Network (DN) into a Bayesian Network structure

A user can make a query to the DN by defining a set of *observation variables* and a *target variable*. In response to the query, *DecAid* is capable of transforming the unconstrained (directed graph) model to a Bayesian network by carrying out the following sequence of steps:

1) Identifying all cycles in the unconstrained model. We use an algorithm by (Johnson, 1975) that finds the elementary cycles in the directed graph by improving over the original algorithm by (Tarjan, 1973);

2) Eliminating the cycles in the unconstrained model by removing the weak edges. This is done, optimally, in order to minimize the information loss in the unconstrained model. This step constitutes a tradeoff between increased expressive power for domain-expert users and modest information loss resulting from removal of edges that least contribute to the information flow.

3) Identifying the sub graph relevant to the query by pruning the non relevant variables from the resulting Bayesian network (Geiger et al, 1990). This step constitutes an important feature of *DecAid* in that it can list all the relevant parameters to the user that are relevant to a specific user query.

The last step in the creation of a query specific Bayesian network is the creation of the conditional probability tables (CPT). The semantics to achieve that are described in section 3.

For practitioners involved in high-level, strategic, decision making the use of Bayesian network building tools can be counterintuitive and may require significant training time, unavailable to such intended users. Making, however, the BN technology accessible through tools like *DecAid* not only will improve the accuracy of decision making but will also provide the means to document and track the chain of causal reasoning behind each decision.

3 SEMANTICS TO CREATE CODITIONAL PROBABILITY TABLES

What follows is a description of the method used to express the random variable (RV) encoded by a DV. That is, we show how to calculate a conditional probability table (CPT) for each variable in the *DecAid* model given its parent set and the size of each variable.

3.1 Concepts Defined as Random Variables

Let X be an n-valued DV from a *DecAid* model **D**. We say that the sample space **S** for X is the real interval [0,1). That is, we can suppose that X describes an experiment whose outcome is a real number r such that $0 \leq r < 1$. The values of the random variable X break the sample space into n disjoint events—namely, half-open intervals of equal length. The set of events is thus:

$\{ r \in [k/n , (k+1)/n) \ : \text{for } 0 \leq k < n \}$

Example (3.1.1)

If X has 2 states, the events corresponding to the states of X are:

$\{ r \in [0.0, 0.5) , r \in [0.5,1.0) \}$.

Example (3.1.2)

If X has 5 states, the events corresponding to the states of X are:

$\{ r \in [0.0, 0.2), r \in [0.2, 0.4), r \in [0.4, 0.6), r \in [0.6, 0.8)$, and $r \in [0.8, 1.0) \}$.

3.2 Conditional Probability Tables

The heart of the probabilistic semantics is the definition of local conditional probability distributions for *DecAid* variables. We consider the various cases below: a) where the variable has no parents, b) where it has one parent of weight 1, c) where it has one parent of arbitrary weight, and finally, d) where it has any number of parents.

Case 3.2.1 -Variables without parents

If X has no parents in **D**, then it is simply given a uniform distribution:

$$P(X = x_k) = 1/n \quad \text{for } 0 \leq k < n .$$

That is, the event $X = x_k$ corresponds to $r \in [k/n, (k+1)/n)$. The probability equals the proportion of the total length of **S** contributed by $X=x_k$. Since the total length of **S** is 1.0, it is simply equal to the length of the interval, which is $(k+1 - k)/n = 1/n$.

Example (3.2.1.1)

If X has 2 states, $P(X = x_k) = 0.5$ for $0 \leq k \leq 1$.

Example (3.2.1.2)

If X has 5 states, $P(X = x_k) = 0.2$ for $0 \leq k \leq 4$.

Case 3.2.2 – Variables with one parent and |w| = 1

We first describe the case where we have a single parent Y and where the link from Y to its child X has weight 1. We need to show how to calculate the conditional probability $P(X = x_k \mid Y = y_j)$. This is given by the formula:

$P(X = x_k \mid Y = y_j , w = 1) = P(X = x_k \ \& \ Y = y_j) / P(Y = y_j)$.

That is, the conditional probability of the event $X = x_k$ given that $Y = y_j$ is equal to the intersection of the intervals corresponding to these events divided by the length of the interval corresponding to $Y = y_j$.

Example (3.2.2.1)

Suppose $Y \rightarrow X$ and Y has 5 states and X has 2 states,

$P(X = x_0 \mid Y = y_2) =$ | Intersection of [0, 0.5) & [0.4, 0.6) | / | 0.6 – 0.4 |= 0.5

The full CPT would be:

$P(x \mid y)$	x_0	x_1
y_0	1	0
y_1	1	0
y_2	0.5	0.5
y_3	0	1
y_4	0	1

Example (3.2.2.2)

Suppose $Z \rightarrow X$ and Z has 3 states and X has 5 states,

$P(X = x_0 \mid Z = z_0) =$ | Intersection of [0, 0.2) & [0, 0.33) | / | 0.33 – 0 | = 0.6

The full CPT is:

$P(x \mid z)$	x_0	x_1	x_2	x_3	x_4
z_0	0.6	0.4	0	0	0
z_1	0	0.2	0.6	0.2	0
z_2	0	0	0	0.4	0.6

Example (3.2.2.3)

Suppose $Y \rightarrow X$ and Y has 2 states and X has 5 states,

$P(X = x_0 \mid Y = y_0) =$ | Intersection of [0, 0.2) & [0, 0.5) | / | 0.5 – 0 | = 0.2 / 0.5 = 0.4

The full CPT is:

$P(x \mid y)$	x_0	x_1	x_2	x_3	x_4
y_0	0.4	0.4	0.2	0	0
y_1	0	0	0.2	0.4	0.4

Case 3.2.3 – Variables with one parent and |w| < 1

We next look at the case where the weight is different than 1. It is useful to refer to the distribution defined in Case 2a as the *full-weight* distribution—i.e., where w=1.

Let $P_{\text{full}}(X \mid y_j)$ be the distribution over the values of X given $Y = y_j$ under the assumption that the arc from Y to X has weight $w = 1$. Let $U(X)$ be the uniform distribution over the values of X. Then, if the weight is $0 \leq w < 1$, we have

$P(X \mid y_j , 0 \leq w < 1) = w \cdot P_{\text{full}}(X \mid y_j) + (1 - w) \cdot U(X)$

That is, the final distribution is a weighted combination of the distribution calculated in Case 3.2.1 and the uniform distribution—which is the default distribution if there were no parent. Note that the weight acts as the probability that we get the full-weight distribution instead of a uniform distribution.

Example (3.2.3.1)

Following the previous example (II.2.3), suppose $Y \rightarrow X$ and Y has 2 states and X has 5 states. But now suppose that the weight of the arc is $w = 0.6$, then we have

$$\begin{aligned} P(X = x_0 \mid Y = y_0) &= w \cdot P_{\text{full}}(X \mid y_0) + (1 - w) \cdot U(X) \\ &= 0.6 \cdot 0.4 + (1.0 - 0.6) \cdot (1/5) \\ &= 0.24 + 0.4 \cdot 0.2 = 0.3 + .08 = 0.32 \end{aligned}$$

The full CPT is:

$P(x \mid y)$	x_0	x_1	x_2	x_3	x_4
y_0	0.32	0.32	0.2	0.08	0.08
y_1	0.08	0.08	0.2	0.32	0.32

If the weight is negative, the direction of the parent's influence is reversed. If Y is an m-valued variable, we can calculate the resulting distribution using a similar calculation above but for the "opposed" value of the parent. By "opposed" we mean the value at the other side of the range—i.e., highest is opposed to lowest, second-highest is opposed to second-lowest, etc. More specifically, if the weight $w < 0$, we have

$P(X \mid y_j, -1 \le w < 0) = w \cdot P_{\text{full}}(X \mid y_{m-j-1}) + (1 - w) \cdot U(X)$

Example (3.2.3.2)

Following the previous example (II.2.3), suppose $Y \rightarrow X$ and Y has 2 states and X has 5 states. But now suppose that the weight of the arc is $w = -0.6$, then we have

$P(X = x_2 \mid Y = y_1) = 0.6 \cdot 0.2 + (1.0 - 0.6) \cdot (1/5) = 0.12 + 0.4 \cdot 0.2 = 0.12 + .08 = 0.2$

The full CPT is:

$P(x \mid y)$	x_0	x_1	x_2	x_3	x_4
y_0	0.08	0.08	0.2	0.32	0.32
y_1	0.32	0.32	0.2	0.08	0.08

Case 3.2.4 – Variables with multiple parents

The remaining situation is when we have a variable with multiple parents. In this situation, we assume that the influence of each parent is independent of the influence of other parents. So, if X has parents $Y^1, Y^2, \ldots, Y^N$, we set

$P(X \mid Y^1, Y^2, \ldots, Y^N) = c \cdot P(X \mid Y^1) \cdot P(X \mid Y^2) \cdot \ldots \cdot P(X \mid Y^N)$

where c is normalization constant to make the distribution sum to 1.

Example (3.2.4.1)

Suppose X has 5 states and two parents: Y with 2 states and weight 0.5 and Z with 3 states and weight –0.5. As we saw above from examples **(3.2.2.1)** and **(3.2.2.2)**, if we ignore the weights of the arcs and the fact that there are multiple parents, we have for parent Z:

$P_{\text{full}}(X \mid z_0) = [\,0.6, 0.4, 0.0, 0.0, 0.0\,]$

And for parent Y:

$P_{\text{full}}(X \mid y_0) = [\,0.4, 0.4, 0.2, 0.0, 0.0\,]$

Next, adding in the effect of the weights on the distributions, we get:

$P(X \mid z_0, w = -.5) = [\,0.1, 0.1, 0.1, 0.3, 0.4\,]$

and

$P(X \mid y_0, w = .5) = [\,0.3, 0.3, 0.2, 0.1, 0.1\,]$

Now, combining both parents we get

$P(X \mid y_0, z_0) = c \cdot P(X \mid y_0) \cdot P(X \mid z_0)$

$= c \cdot [\,0.3, 0.3, 0.2, 0.1, 0.1\,] \cdot [\,0.1, 0.1, 0.1, 0.3, 0.4\,]$

$= c \cdot [\,0.03, 0.03, 0.02, 0.03, 0.04\,]$

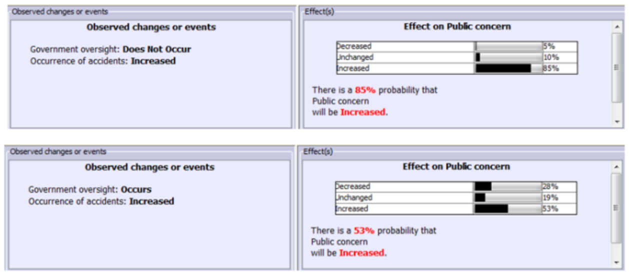
$= [0.2, 0.2, 0.13, 0.2, 0.27]$

The full CPT is:

$P(x \mid y, z)$	x_0	x_1	x_2	x_3	x_4
y_0, z_0	0.2	0.2	0.13	0.2	0.27
y_0, z_1	0.15	0.3	0.4	0.1	0.05
y_0, z_2	0.48	0.36	0.08	0.04	0.04
y_1, z_0	0.04	0.04	0.08	0.36	0.48
y_1, z_1	0.05	0.1	0.4	0.3	0.15
y_1, z_2	0.27	0.2	0.13	0.2	0.2

4 DISCUSSION

DecAid is used for strategic decision making. Here are a few examples of such decisions: a) when to launch a new product into a specific market, b) how close is a rouge country to achieving nuclear weapon capability, or c) whether to invest in a particular emerging technology. These are decisions that involve several variables and their inter relations. The system enables decision makers to define concepts of the problem in a simple, intuitive, manner using free language. As the user defines the concepts and relations, the system is creating an unconstrained model. Once, the model is built, *DecAid* is capable of making predictions in response to queries by converting the unconstrained model into a Bayesian network.


Observed changes or events

Observed changes or events

Government oversight: **Does Not Occur**
Occurrence of accidents: **Increased**

Effect(s)

Effect on Public concern

Decreased	5%
Unchanged	10%
Increased	85%

There is a **85%** probability that Public concern will be **Increased**.

Observed changes or events

Observed changes or events

Government oversight: **Occurs**
Occurrence of accidents: **Increased**

Effect(s)

Effect on Public concern

Decreased	28%
Unchanged	19%
Increased	53%

There is a **53%** probability that Public concern will be **Increased**.

Figure 4.2: Predictions made by *DecAid* Model

Figure 4.2 shows two such predictions derived from the model in Figure 2.1. The first prediction forecasts a 0.85 probability that "Public concern" will increase given that "Occurrence of accidents" has increased and "Government oversight" does not occur. The second prediction lowers the forecast that "Public concern" will increase to a 0.53 probability, if "Government oversight" occurs.

At the final stage of making decisions, summarization and argumentation becomes critical steps. It is the aim of *DecAid* to facilitate the capture of knowledge and information for that last stage, as well, by combining the predictive analytic capability obtained from the cognitive models with the ability to retrieve evidential data and information to validated predictive hypotheses, which is outside the scope of this paper.

The complete probabilistic semantics of a *DecAid* model include how the local probability models are combined (not discussed here). The cornerstone of the semantics, however, is the definition given here for the complete CPT of a local variable from the simple numeric weights associated with its parents as provided by the end-user creating the model.

DecAid variables represent a tradeoff between simplicity of model definition and expressive power. Aside from adding temporal modeling (Nodelman, et. al. 2002, 2003) to *DecAid*, there are additional areas where the balance between simplicity and expressivity could be further enhanced. In one such area there is, currently, complete symmetry between the positive effect of a parent taking on a high value and the negative effect of a parent taking on a low value. Sometimes this symmetry is warranted but sometimes it is not. For example, consider a child variable "Strength of a Fire" ("fire") with a parent "Oxygen Level Present" ("oxygen"). Increasing oxygen will tend to increase the fire and decreasing oxygen will tend to lessen the fire. But now consider an alternative parent "Use of fire-extinguisher". If the fire-extinguisher is used, that will tend to lessen the fire. But lack of fire-extinguisher use does not, in itself, increase the fire. So we may, in general, want to allow an asymmetry between the impact of a high-value parent and a low-value parent where the high-value has the regular effect but the low-value has no special impact on the child.

Furthermore, the assumption that the effects of multiple parents are independent of each other is strong. Obviously, there are many cases where this assumption is unwarranted. The problem would be to find a simple, understandable way for end-users to convey extra information about covariance and to find an algorithm that could examine the link structure in other parts of the *DecAid* model and extract some useful information about the dependencies among the parents.

5. SUMMARY

The semantics definition is given in this paper for the complete CPT of a local variable from simple numeric weights associated with its parents as provided by the end-user creating a *DecAid* model. Explicitly, 1) The values of a DV are represented as equal-length subintervals of the unit interval and making explicit that they have a natural ordering so they can be seen as coming in opposed pairs (except for a possible middle-most value). 2) A single parent full-weight conditional probability is defined as the size of the intersection of parent and child intervals divided by the size of parent interval. 3) The magnitude of the weight is used as the probability that you get the full-weight conditional distribution instead of a uniform distribution. 4) The sign of the weight is used to reverse the direction of influence. And 5) the probabilistic influence of multiple parents on a child are assumed to be independent of one another.

The semantics described in this paper enable the creation of a Bayesian networks from an unconstrained, directed graph model created by a user within a simpler, more intuitive, framework implemented in a tool called *DecAid*, without requiring specialized training in how to build Bayesian networks.

Acknowledgement

This work would have not been possible without the contributions and valuable, long-term, collaboration with Uri Nodelman for which the author and The Boeing Company are greatly thankful.

References

Farry M, Pfautz J, Cox Z, Bisantz A, Stone R, and Roth E (2008), "An Experimental Procedure for Evaluating User-Centered methods for Rapid Bayesian Network Construction", *Proceedings of the 6th Bayesian Modeling Applications Workshop at the 24th Annual Conference on Uncertainty in AI: UAI 2008, Helsinki, Finland*.

Geiger, Verma, and Pearl, (1990), "Identifying Independence in Bayesian Networks", *Networks 20:507-534*.

Johnson, B. (1975), "Finding all the elementary circuits in a directed graph", *SIAM Journal of Computing, 4(1)*.

Kipersztok, O (2007). "Using Human Factors, Reasoning and Text Processing for Hypothesis Validation." *Third International Workshop on Knowledge and Reasoning for Answering Questions. International Joint Conference in Artificial Intelligence, Hydrabad, India*.

Kipersztok O (2004). "Combining Cognitive Causal Models with Reasoning and Text Processing Methods for Decision Support." *Second Bayesian Modeling Applications Workshop at the Uncertainty in AI Conference, Banff Canada*.

Nodelman, U., Shelton, C. R., and Koller, D. (2002). "Continuous Time Bayesian Networks." *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, pp. 378-387, 2002*.

Nodelman, U., Shelton, C.R., and Koller, D. (2003). "Learning Continuous Time Bayesian Networks." *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, pp. 451-458, 2003.*

Pfautz J, Cox Z, Catto G, Koelle D, Campolongo J, Roth E (2007),"User-Centered Methods for Rapid Creation and Validation of Bayesian Networks". *In Proceedings of 5th Bayesian Applications Workshop at Uncertainty in Artificial Intelligence (UAI 07).*

Rosen J and Smith W (1996), "Influence Net Modeling with Causal Strengths: An Evolutionary Approach," *In the Proceedings of the 1996 Command and Control Research and Technology Symposium, Monterey CA.*

Tarjan, R. E. (1972), "Depth-first search and linear graph algorithms", *SIAM Journal on Computing 1 (2): 146–160.*

Tarjan, R.E. (1973), "Enumeration of the elementary circuits of a directed graph", *SIAM Journal of Computing 2 :211-216.*

.

Bayesian Network Parameter Learning using EM with Parameter Sharing

Erik Reed
Electrical and Computer Engineering
Carnegie Mellon University

Ole J. Mengshoel
Electrical and Computer Engineering
Carnegie Mellon University

Abstract

This paper explores the effects of parameter sharing on Bayesian network (BN) parameter learning when there is incomplete data. Using the Expectation Maximization (EM) algorithm, we investigate how varying degrees of parameter sharing, varying number of hidden nodes, and different dataset sizes impact EM performance. The specific metrics of EM performance examined are: likelihood, error, and the number of iterations required for convergence. These metrics are important in a number of applications, and we emphasize learning of BNs for diagnosis of electrical power systems. One main point, which we investigate both analytically and empirically, is how parameter sharing impacts the error associated with EM's parameter estimates.

1 INTRODUCTION

Bayesian network (BN) conditional probability tables (CPTs) can be learned when the BN structure is known, for either complete or incomplete data. Different algorithms have been explored in the case of incomplete data, including: Expectation Maximization [8, 14, 15, 28], Markov Chain Monte Carlo methods such as Gibbs sampling [17], and gradient descent methods [9]. Expectation Maximization (EM) seeks to maximize the likelihood, or the Maximum a Posteriori (MAP) estimate, for the BN CPTs.

We focus in this paper on EM [8, 14, 15, 28], an iterative algorithm that converges to a maximum likelihood estimate (MLE). While EM is powerful and popular, there are several challenges that motivate our research. First, when computing MLEs, EM is easily trapped in local optima and is typically very sensitive to the placement of initial CPT values. Methods of making EM less prone to getting trapped in local optimal have been investigated [11, 18, 34, 38]. Second, EM is often computationally demanding, especially when the BN is complex and there is much data [2, 3, 29, 35]. Third, parameters that EM converges to can be far from the true probability distribution, yet still have a high likelihood. This is a limitation of EM based on MLE.

In this paper we investigate, for known BN structures, how varying degree of parameter sharing [17, 25, 26], varying number of hidden nodes, and different dataset sizes impact EM performance. Specifically, we are:

- running many random initializations (or random restarts) of EM, a technique known to effectively counter-act premature convergence [10, 22];
- recording for each EM run the following metrics: (i) log-likelihood ($\ell\ell$) of estimated BN parameters, (ii) error (the Euclidean distance between true and estimated BN parameters), and (iii) number of EM iterations until convergence; and
- testing BNs with great potential for parameter sharing, with a focus on electrical power system BNs (reflecting electrical power system components known to exhibit similar behavior).

Even when EM converges to a high-likelihood MLE, the error can be large and vary depending on initial conditions. This is a fundamental limitation of EM using MLE; even a BN with high likelihood may be far from the true distribution and thus have a large error. Error as a metric for the EM algorithm for BN parameter learning has not been discussed extensively in the existing literature. The analysis and experiments in this paper provide new insights in this area.

Our main application is electrical power systems, and in particular NASA's Advanced Diagnostics and Prognostics Testbed (ADAPT) [27]. ADAPT has already been represented as BNs, which have proven themselves as very well-suited to electrical power system health management [12,19–21,30–33]. Through compilation of BNs to arithmetic circuits [4,5], a broad range of discrete and continuous faults can be detected and diagnosed in a computationally efficient and predictable manner, resulting in award-winning performance in international diagnostic competitions [30].[1] From a machine learning and EM perspective, as considered in this paper, it is hypothesized that the learning of ADAPT BNs may benefit from parameter sharing. This is because there are several repeated BN nodes and fragments in these BNs. In addition to parameter sharing, we study in this paper the impact on EM of varying the number of hidden nodes, reflecting different sensing capabilities.

Why are BNs and arithmetic circuits useful for electrical power system diagnostics? First, power systems exhibit multi-variate uncertainty, for example regarding component and sensor health (are they working or failing?) as well as noisy sensor readings. Second, there is substantial local structure, as reflected in an EPS schematic, that can be taken advantage of when constructing a BN automatically or semi-automatically [19,20,30]. Consequently, BN treewidth is small enough for exact computation using junction trees, variable elimination, or arithmetic circuits to be feasible [19,30]. Third, we compile BNs into arithmetic circuits [4,5], which are fast and predictable in addition to being exact. These are all important benefits in cyber-physical systems including electrical power systems.

The rest of this paper is structured as follows. In Section 2, we introduce BNs, parameter learning for incomplete data using EM, and related research. Section 3 presents our main application area, electrical power systems. In Section 4, we define the sharing concept, discuss sharing in EM for BN parameter learning, and provide analytical results. In Section 5 we present experimental results for parameter sharing in BNs when using EM, emphasizing electrical power system fault diagnosis using BNs. Finally, we conclude and outline future research opportunities in Section 6.

2 BACKGROUND

This section presents preliminaries including notation (see also Table 1).

2.1 BAYESIAN NETWORKS

Consider a BN $\beta = (\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\theta})$, where $\boldsymbol{X}$ are discrete nodes, $\boldsymbol{W}$ are edges, and $\boldsymbol{\theta}$ are CPT parameters. Let $\boldsymbol{E} \subseteq \boldsymbol{X}$ be evidence nodes, and $\boldsymbol{e}$ the evidence. A

Notation	**Explanation**
$\boldsymbol{X}$	BN nodes
$\boldsymbol{W}$	BN edges
$\boldsymbol{\theta}$	BN CPTs
$\hat{\boldsymbol{\theta}}$	estimated CPTs
$\boldsymbol{\theta}^*$	true CPTs
$\boldsymbol{O}$	observable nodes
$\boldsymbol{H}$	hidden nodes
$\boldsymbol{S}$	(actually) shared nodes
$\boldsymbol{P}$	(potentially) shared nodes
$\boldsymbol{U}$	unshared nodes
$\boldsymbol{Y}$	set partition of $\boldsymbol{X}$
$T_{\boldsymbol{P}}$	number of wrong CPTs
$\boldsymbol{E}$	evidence nodes
$\boldsymbol{R}$	non-evidence nodes
$t_{\min}$	min # of EM iterations
$t_{\max}$	max # of EM iterations
t'	iteration # at EM convergence
ϵ	tolerance for EM
$err(\hat{\boldsymbol{\theta}})$	error of $\hat{\boldsymbol{\theta}}$ relative to $\boldsymbol{\theta}^*$
$n_{\boldsymbol{A}} = \lvert\boldsymbol{A}\rvert$	cardinality of the set $\boldsymbol{A}$
ℓ	likelihood
$\ell\ell$	log-likelihood
$\beta = (\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\theta})$	Bayesian network (BN)
$E(Z)$	expectation of r.v. Z
$V(Z)$	variance of r.v. Z
r	Pearson's corr. coeff.
$\theta \in [0,1]$	CPT parameter
$\hat{\theta} \in [0,1]$	estimated CPT parameter
$\theta^* \in [0,1]$	true CPT parameter
δ	error bound for θ

Table 1: Notation used in this paper.

BN factors a joint distribution $\Pr(\boldsymbol{X})$, enabling different probabilistic queries to be answered by efficient algorithms; they assume that nodes $\boldsymbol{E}$ are clamped to values $\boldsymbol{e}$. One query of interest is to compute a most probable explanation (MPE) over the remaining nodes $\boldsymbol{R} = \boldsymbol{X} \setminus \boldsymbol{E}$, or MPE($\boldsymbol{e}$). Computation of marginals (or beliefs) amounts to inferring the posterior probabilities over one or more query nodes $\boldsymbol{Q} \subseteq \boldsymbol{R}$, specifically BEL($Q, \boldsymbol{e}$), where $Q \in \boldsymbol{Q}$.

In this paper, we focus on situations where $\boldsymbol{\theta}$ needs to be estimated but the BN structure ($\boldsymbol{X}$ and $\boldsymbol{W}$) is known. Data is complete or incomplete; in other words there may be hidden nodes $\boldsymbol{H}$ where $\boldsymbol{H} = \boldsymbol{X} \setminus \boldsymbol{O}$ and $\boldsymbol{O}$ are observed. A dataset is defined as $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ with m samples (observations), where $\mathbf{x}_i$ is a vector of instantiations of nodes $\boldsymbol{X}$ in the complete data case. When the data is complete, the BN parameters $\boldsymbol{\theta}$ are often estimated to maximize the data likelihood (MLE). In this paper, for a given dataset, a variable $X \in \boldsymbol{X}$ is either observable ($X \in \boldsymbol{O}$) or hidden ($X \in \boldsymbol{H}$); it is not hidden for just a strict subset of the samples.[2] Let $\boldsymbol{H} > 0$. For each hidden node

[1]Further information can be found here: `https://sites.google.com/site/dxcompetition/`.

[2]In other words, a variable that is completely hidden in the training data is a latent variable. Consequently, its

$H \in \boldsymbol{H}$ there is then a "?" or "N/A" in each sample. Learning from incomplete data also relies on a likelihood function, similar to the complete data case. However, for incomplete data several properties of the complete data likelihood function–such as unimodality, a closed-form representation, and decomposition into a product form–are lost. As a consequence, the computational issues associated with BN parameter learning are more complex, as we now discuss.

2.2 TRADITIONAL EM

For the problem of optimizing such multi-dimensional, highly non-linear, and multimodal functions, several algorithms have been developed. They include EM, our focus in this paper. EM performs a type of hill-climbing in which an estimate in the form of an expected likelihood function ℓ is used in place of the true likelihood ℓ.

Specifically, we examine the EM approach to learn BN parameters $\boldsymbol{\theta}$ from incomplete data sets.[3] The *traditional EM* algorithm, without sharing, initializes parameters to $\boldsymbol{\theta}^{(0)}$. Then, EM alternates between an E-step and an M-step. In the t-th E-step, using parameters $\boldsymbol{\theta}^{(t)}$ and observables from the dataset, EM generates the likelihood $\ell^{(t)}$ taking into account the hidden nodes $\boldsymbol{H}$. In the M-step, EM modifies the parameters to $\boldsymbol{\theta}^{(t+1)}$ to maximize the data likelihood. While $|\ell^{(t)} - \ell^{(t-1)}| \geq \epsilon$, where ϵ is a tolerance, EM repeats from the E-step.

EM monotonically increases the likelihood function ℓ or the log-likelihood function $\ell\ell$, thus EM converges to a point $\hat{\boldsymbol{\theta}}$ or a set of points (a region). Since $\ell\ell$ is bounded, EM is guaranteed to converge. Typically, EM converges to a local maximum [37] at some iteration t', bounded as follows: $t_{\max} \geq t' \geq t_{\min}$. Due to the use of ϵ above, it is for practical implementations of EM with restart more precise to discuss regions of convergence, even when there is point-convergence in theory.

One topic that has been discussed is the initialization phase of the EM algorithm [8]. A second research topic is stochastic variants of EM, typically known as Stochastic EM [7,11]. Generally, Stochastic EM is concerned with improving the computational efficiency of EM's E-step. Several other methods for increasing the efficacy of the EM algorithm for BNs exist. These include parameter constraints [1, 6, 25], parameter inequalities [26], exploiting domain knowledge [17, 24], and parameter sharing [13, 17, 25, 26].

true distribution is not identifiable.

[3]Using EM, learning from complete data is a special case of learning from incomplete data.

When data is incomplete, the BN parameter estimation problem is in general non-identifiable. There may be several parameter estimates $\hat{\boldsymbol{\theta}}_1$, ..., $\hat{\boldsymbol{\theta}}_m$ that have the same likelihood, given the dataset [36]. Thus, we need to be careful when applying standard asymptotic theory from statistics (which assumes identifiability) and when interpreting a learned model. Section 4.2 introduces an error measure that provides some insight regarding identifiability, since it measures distance from the true distribution $\boldsymbol{\theta}^*$.

3 ELECTRICAL POWER SYSTEMS

Electrical Power Systems (EPSs) are critical in today's society, for instance they are essential for the safe operation of aircraft and spacecraft. The terrestrial power grid's transition into a smart grid is also very important, and the emergence of electrical power in hybrid and all-electric cars is a striking trend in the automotive industry.

ADAPT (Advanced Diagnostics and Prognostics Testbed) is an EPS testbed developed at NASA [27]. Publicly available data from ADAPT is being used to develop, evaluate, and mature diagnosis and prognosis algorithms. The EPS functions of ADAPT are as follows. For power generation, it currently uses utility power with battery chargers (there are also plans to investigate solar power generation). For power storage, ADAPT contains three sets of 24 VDC 100 Amp-hr sealed lead acid batteries. Power distribution is aided by electromechanical relays, and there are two load banks with AC and DC outputs. For control and monitoring there are two National Instruments compact FieldPoint backplanes. Finally, there are sensors of several types, including for: voltage, current, temperature, light, and relay positions.

ADAPT has been used in different configurations and represented in several fault detection and diagnosis BNs [12, 19–21, 30–33], some of which are investigated in this paper (see Table 2). Each ADAPT BN node typically has two to five discrete states. BN nodes represent, for instance: sensors (measuring, for example, voltage or temperature); components (for example batteries, loads, or relays); or system health of components and sensors (broadly, they can be in "healthy" or "faulty" states).

3.1 SHARED VERSUS UNSHARED

In BN instances where parameters are not shared, the CPT for a node in the BN is treated as separate from the other CPTs. The assumption is not always reasonable, however. ADAPT BNs may benefit from shared parameters, because there are typically several

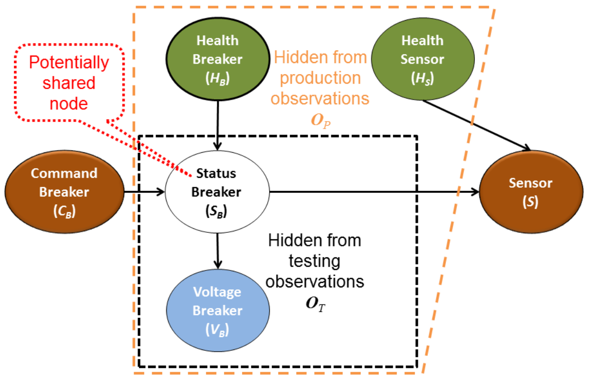


Figure 1: One of three similar sub-networks in the mini-ADAPT BN. Under parameter sharing, the S_B node is shared between the three sub-networks.

repeated nodes or fragments in these BNs [21, 30]. It is reasonable to assume that "identical" power system sensors and components will behave in similar ways. More broadly, sub-networks of identical components should function in a similar way to other "identical" sub-networks in a power system, and such knowledge can be the basis for parameter sharing.

For parameter sharing as investigated in this paper, the CPTs of some nodes are assumed to be approximately equal to the CPTs of different nodes elsewhere in the BN.[4] Data for one set of nodes can be used elsewhere in the BN if the corresponding nodes are shared during EM learning. This is a case of parameter sharing involving the global structure of the BN, where different CPTs are shared, as opposed to parameter sharing within a single CPT [13].

In some ADAPT BNs, one sub-network is essentially duplicated three times, reflecting triple redundancy in ADAPT's power storage and distribution network [27]. Such a six-node sub-network from an ADAPT BN is shown in Figure 1. This sub-network was, in this paper, manually selected for further study of sharing due to this duplication. The sub-network is part of the mini-ADAPT BN used in experiments in Section 5.3. In the mini-ADAPT sharing condition, only the node S_B was shared between all three BN fragments. Generally, we define shared nodes $\boldsymbol{S}$ and unshared nodes $\boldsymbol{U}$, with $\boldsymbol{U} = \boldsymbol{X} \setminus \boldsymbol{S}$.

[4]It is unrealistic to assume that several engineered physical objects, even when it is desired that they are exactly the same, in fact turn out to have exactly the same behavior. A similar argument has been made for object-oriented BNs [14], describing it as "violating the OO assumption." We thus say that CPTs are approximately equal rather than equal. Under sharing, however, we are making the simplifying assumption that shared CPTs are equal.

3.2 OBSERVABLE VERSUS HIDDEN

Consider a complex engineered system, such as an electrical power system. After construction, but before it is put into production, it is typically tested extensively. The sensors used during *testing* lead to one set of observation nodes in the BN, $\boldsymbol{O_T}$. The sensors used during *production* lead to another set of observations in the BN, $\boldsymbol{O_P}$. For reasons including cost, fewer sensors are typically used during production than during testing, thus we assume $\boldsymbol{O_P} \subseteq \boldsymbol{O_T}$.

As an example, in Figure 1 we denote $\boldsymbol{O_P} = \{C_B, S\}$ as *production* observation nodes, and $\boldsymbol{O_T} = \{C_B, S, H_B, H_S\}$ as *testing* observation nodes.

In all our experiments, shared nodes are also hidden, or $\boldsymbol{S} \subseteq \boldsymbol{H}$. Typically, there are hidden nodes that are not necessarily shared, or $\boldsymbol{S} \subset \boldsymbol{H}$. This is, for example, the case for the mini-ADAPT BN as reflected in the sub-network in Figure 1.

4 EM WITH SHARING

4.1 SHARING IN BAYESIAN NETWORKS

Consider a BN $\beta = (\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\theta})$. A sharing set partition for nodes $\boldsymbol{X}$ is a set partition $\boldsymbol{Y}$ of $\boldsymbol{X}$ with subsets $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_k$, with $\boldsymbol{Y}_i \subseteq \boldsymbol{X}$ and $k \geq 1$. For each $\boldsymbol{Y}_i$ with $k \geq i \geq 1$ the nodes $X \in \boldsymbol{Y}_i$ share a CPT during EM learning as discussed in Section 4.2. We assume that the nodes in $\boldsymbol{Y}_i$ have exactly the same number of states. The same applies to their respective parents in β, leading to each $Y \in \boldsymbol{Y}_i$ having the same number of parent instantiations and exactly the same CPTs.

Traditional non-sharing is a special case of sharing in the following way. We assign each BN node to a separate set partition such that for $\boldsymbol{X} = \{X_1, ..., X_n\}$ we have $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n$ with $\boldsymbol{Y}_i = \{X_i\}$.

One key research goal is to better understand the behavior of EM as sharing nodes $\boldsymbol{S}$ and observable nodes $\boldsymbol{O}$ vary. We examine three cases: complete data $\boldsymbol{O}_C$ (no hidden nodes); a sensor-rich testing setting with observations $\boldsymbol{O}_T$; and a sensor-poor production setting with observations $\boldsymbol{O}_P$. Understanding the impact of varying observations $\boldsymbol{O}$ is important due to cost and effort associated with observation or sensing.

4.2 SHARING EM

Similar to traditional EM (see Section 2.2), the sharing EM algorithm also takes as input a dataset and estimates a vector of BN parameters $\hat{\boldsymbol{\theta}}$ by iteratively improving $\ell\ell$ until convergence. The main difference of sharing EM compared to traditional EM is that we are now setting some nodes as shared $\boldsymbol{S}$, according to

$\boldsymbol{Y}$. To arrive at the sharing EM algorithm from the traditional EM algorithm, we modify the likelihood function to introduce parameter sharing and combine parameters (see also [13, 17]). When running sharing EM, nodes $\boldsymbol{X}$ are treated as separate for the E-step. There is a slightly modified M-step, using $\boldsymbol{Y}$, to aggregate the shared CPTs and parameters.[5] This is the use of *aggregate sufficient statistics*, which considers sufficient statistics from more than one BN node [13].

Let $\hat{\theta}_{i,j}$ be the j'th estimated probability parameter for BN node $X_i \in \boldsymbol{X}$. We define error of $\hat{\boldsymbol{\theta}}$ for BN $(\boldsymbol{X}, \boldsymbol{W}, \hat{\boldsymbol{\theta}})$ as the L^2 distance from the true probability distribution $\boldsymbol{\theta}^*$ from which data is sampled:

$$err(\hat{\boldsymbol{\theta}}) = \sum_i \sqrt{\sum_j \left(\theta^*_{i,j} - \hat{\theta}_{i,j}\right)^2}. \tag{1}$$

This error is the summation of the Euclidean distance between true and estimated CPT parameters, or the L^2 distance, providing an overall distance metric.

Why do we use Euclidean distance to measure error? One could, after all, argue that this distance metric is poor because it does not agree with likelihood. We use Euclidean distance because we are interested not only in the black box performance of the BN, but also its validity and understandability to a human expert.[6] This is important when an expert needs to evaluate, validate, or refine a BN model, for example a BN for an electrical power system.

4.3 EM'S BEHAVIOR UNDER SHARING

We now provide a simple analysis of certain aspects of traditional EM (TEM) and sharing EM (SEM). For simplicity, we only consider EM runs that converge and exclude runs that time out.[7]

For a node $X_i \in \boldsymbol{X}$, TEM will converge to one among potentially several convergence regions. Suppose that the CPT of node X_i has $\kappa(X_i)$ convergence regions. Then the actual number of convergence regions $\kappa(\beta_U)$ for a non-shared BN β_U with nodes $\boldsymbol{X} = \{X_1, ..., X_n\}$ is upper bounded by $\bar{\kappa}(\beta_U)$ as follows:

$$\kappa(\beta_U) \leq \bar{\kappa}(\beta_U) = \prod_{i=1}^{n} \kappa(X_i). \tag{2}$$

Due to the sharing, SEM intuitively has fewer convergence regions than TEM. This is due to SEM's slightly modified M-step that aggregates the shared CPTs and parameters. Consider a BN β_S with exactly the same nodes and edges as β_U, but with sharing, specifically with sharing set partitions $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_k$ and $k < n$. Without loss of generality, assume that $X_i \in \boldsymbol{Y}_i$. Then the actual number of convergence regions $\kappa(\beta_S)$ is upper bounded by $\bar{\kappa}(\beta_S)$ as follows:

$$\kappa(\beta_S) \leq \bar{\kappa}(\beta_S) = \prod_{i=1}^{k} \kappa(X_i), \tag{3}$$

assuming that the $\kappa(X_i)$ convergence regions used for X_i in (2) carry over to $\boldsymbol{Y}_i$.

A special case of (3) is when there exists exactly one $\boldsymbol{Y}' \in \boldsymbol{Y}$ such that $|\boldsymbol{Y}'| \geq 2$ while for any $\boldsymbol{Z} \in \boldsymbol{Y} \backslash \boldsymbol{Y}'$ we have $|\boldsymbol{Z}| = 1$. The experiments performed in Section 5 are all for this special case. Specifically, but without loss of generality, let $\boldsymbol{Y}_i = \{X_i\}$ for $1 \leq i < k$ and $\boldsymbol{Y}_k = \{X_k, ..., X_n\}$ with $n_{\boldsymbol{S}} = n - k + 1$ (*i.e.*, $\boldsymbol{S} = \boldsymbol{Y}_k$ has $n_{\boldsymbol{S}}$ sharing nodes). It is illustrative to consider the ratio:

$$\frac{\bar{\kappa}(\beta_U)}{\bar{\kappa}(\beta_S)} = \frac{\prod_{i=1}^{n} \kappa(X_i)}{\prod_{i=1}^{k} \kappa(X_i)} = \prod_{i=k+1}^{n} \kappa(X_i). \tag{4}$$

Here, we assume that X_i has $\kappa(X_i)$ convergence regions in both β_U and β_S and take into account that for shared nodes $\boldsymbol{Y}_k = \boldsymbol{S}$, CPTs are tied together.

The simple analysis above suggests a non-trivial impact of sharing, given the multiplicative effect of the $\kappa(X_i)$'s for $k+1 \leq i \leq n$ in (4). However, since upper bounds are the focus in this analysis, only a partial and conservative picture is painted. The experiments in Section 5—see for example Figure 3, Figure 5, and Figure 6—provide further details.

4.4 ANALYSIS OF ERROR

We now consider the number of erroneous CPTs as estimated by SEM when sharing is varied. Clearly, a CPT parameter is continuous and its EM estimate is extremely unlikely to be equal to the original parameter. Thus we consider here a discrete variable, based on forming an interval in the one-parameter case. Generally, let a discrete BN node $X \in \boldsymbol{X}$ have k states such that $x_i \in \{x_1, ..., x_k\}$. Consider $\theta_{x_i|\boldsymbol{z}} = \Pr(X = x_i \mid \boldsymbol{Z} = \boldsymbol{z})$ for a parent instantiation $\boldsymbol{z}$. We now have an original CPT parameter $\theta_i \in \{\theta_{x_1|\boldsymbol{z}}, ..., \theta_{x_{k-1}|\boldsymbol{z}}\}$ and its EM estimate $\hat{\theta}_i \in \{\hat{\theta}_{x_1|\boldsymbol{z}}, ..., \hat{\theta}_{x_{k-1}|\boldsymbol{z}}\}$. Let us jointly consider the original CPT parameter θ^*_i and its estimate $\hat{\theta}_i$. If $\hat{\theta}_i \in [\theta^*_i - \delta_i, \theta^*_i + \delta_i]$ we count $\hat{\theta}_i$ as correct, and say $\hat{\theta}_i = \theta^*_i$; else it is incorrect or wrong, and we

[5]For the relevant LibDAI source code, please see here: `https://github.com/erikreed/HadoopBNEM/blob/master/src/emalg.cpp#L167`. In words, it is a modification on the collection of sufficient statistics during the maximization step.

[6]We assume that (1) is better than likelihood in this regard, if the original BN was manually constructed. The BNs experimented with in Section 5 were manually constructed, for example.

[7]In practice, runs that time out are very rare with the parameter settings we use in experiments.

say $\hat{\theta}_i \neq \theta_i^*$. This analysis clearly carries over to multiple CPT parameters, parent instantiations, and BN nodes. This shows how we go from a continuous (estimated CPT parameters $\hat{\boldsymbol{\theta}}$) to a discrete value (number of wrong or incorrect CPT estimates), where the latter is used in this analysis.

Suppose that up to $n_{\boldsymbol{P}}$ nodes can be shared. Further suppose that $n_{\boldsymbol{S}}$ nodes are actually shared while $n_{\boldsymbol{U}}$ nodes are unshared, with $n_{\boldsymbol{U}} + n_{\boldsymbol{S}} = n_{\boldsymbol{P}}$. Let $T_{\boldsymbol{P}}$ be a random variable representing the total number of wrong or incorrect CPTs, $T_{\boldsymbol{S}}$ the total for shared nodes, and $T_{\boldsymbol{U}}$ the total for unshared nodes. Clearly, we have $T_{\boldsymbol{P}} = T_{\boldsymbol{S}} + T_{\boldsymbol{U}}$.

Let us first consider the expectation $E(T_{\boldsymbol{P}})$. Due to linearity, we have $E(T_{\boldsymbol{P}}) = E(T_{\boldsymbol{S}}) + E(T_{\boldsymbol{U}})$. In the non-shared case, assume for simplicity that errors are iid and follow a Bernoulli distribution, with probability p of error and $(1-p) = q$ of no error.[8] This gives $E(T_{\boldsymbol{U}}) = n_{\boldsymbol{U}}p$, using the fact that a sum of Bernoulli random variables follows a Binomial distribution.[9]

In the shared case, all shared nodes either have an incorrect CPT $\hat{\theta} \neq \theta^*$ or the correct CPT $\hat{\theta} = \theta^*$. Assuming again probabilities p of error[10] and $(1-p) = q$ of no error, and by using the definition of expectation of Binomials we obtain $E(T_{\boldsymbol{S}}) = n_{\boldsymbol{S}}p$.

Substituting into $E(T_{\boldsymbol{P}})$ we get

$$E(T_{\boldsymbol{P}}) = n_{\boldsymbol{U}}p + n_{\boldsymbol{S}}p = n_{\boldsymbol{P}}p. \qquad (5)$$

Let us next consider the variance $V(T_{\boldsymbol{P}})$. While variance in general is not linear, we assume linearity for simplicity, and obtain

$$V(T_{\boldsymbol{P}}) = V(T_{\boldsymbol{U}}) + V(T_{\boldsymbol{S}}). \qquad (6)$$

In the non-shared case we have again a Binomial distribution, with well-known variance

$$V(T_{\boldsymbol{U}}) = n_{\boldsymbol{U}}p(1-p). \qquad (7)$$

In the shared case we use the definition of variance, put $p_1 = (1-p)$, $p_2 = p$, and $\mu = n_{\boldsymbol{S}}p$, and obtain after some simple manipulations:

$$V(T_{\boldsymbol{S}}) = \sum_{i=1}^{2} p_i(X_i - \mu)^2 = n_{\boldsymbol{S}}^2 p(1-p), \qquad (8)$$

[8]This is a simplification, since our use of the Bernoulli assumes that each CPT is either "correct" or "incorrect." When learned from data, the estimated parameters are clearly almost never exactly correct, but close to or far from their respective original values.

[9]If X is Binomial with parameters n and p, it is well-known that the expected value is $E(X) = np$.

[10]The error probabilities of $T_{\boldsymbol{S}}$ and $T_{\boldsymbol{U}}$ are assumed to be the same as a simplifying assumption.

and by substituting (7) and (8) into (6) we get

$$V(T_{\boldsymbol{P}}) = p(1-p)((n_{\boldsymbol{P}} - n_{\boldsymbol{S}}) + n_{\boldsymbol{S}}^2). \qquad (9)$$

In words, (9) tells us that as the number $n_{\boldsymbol{S}}$ of shared nodes increases at the expense of the number of unshared nodes $n_{\boldsymbol{U}}$, variance due to non-shared nodes decreases linearly, but variance due to sharing increases quadratically. The net effect shown in (9) is that variance $V(T_{\boldsymbol{P}})$ of the error *increases* with the number of shared nodes, according to our analysis above. Expectation, on the other hand, remains *constant* (5) regardless of how many nodes are shared. These analytical results have empirical counterparts as discussed in Section 5, see for example the error sub-plot at the bottom of Figure 2.

5 EXPERIMENTS

We now report on EM experiments for several different BNs, using varying degrees of sharing. We also vary the number of hidden nodes and dataset size. We used $\epsilon = 1e^{-3}$ as an EM convergence criterion, meaning that EM stopped at iteration t' when the $\ell\ell$-score changed by a value $\leq \epsilon$ between iterations $t'-1$ and t' for $t_{\max} \geq t' \geq t_{\min}$. In these experiments, $t_{\max} = 100$ and $t_{\min} = 3$.

5.1 METHODS AND DATA

Bayesian networks. Table 2 presents BNs used in the experiments.[11] Except for the BN Pigs, these BNs all represent (parts of) the ADAPT electrical power system (see Section 3). The BN Pigs has the largest number of nodes that can be shared ($n_{\boldsymbol{P}} = 296$), comprising 67% of the entire BN. The largest BN used, in terms of node count, edges, and total CPT size, is ADAPT_T2.

Datasets. Data for EM learning of parameters for these BNs were generated using forward sampling.[12] Each sample in a dataset is a vector $\boldsymbol{x}$ (see Section 2.1). The larger BNs were tested with increasing numbers of samples ranging from 25 to 400, while mini-ADAPT was tested with 25 to 2000 samples.

Sharing. Each BN has a different number of parameters that can be shared, where a set of nodes $\boldsymbol{Y}_i \in$

[11]ADAPT BNs can be found here: `http://works.bepress.com/ole_mengshoel/`.

[12]Our experiments are limited in that we are only learning the parameters of BNs, using data generated from those BNs. Clearly, in most applications, data is not generated from a BN and the true distribution does not conform exactly to some BN structure. However, our analytical and experimental investigation of error would not have been possible without this simplifying assumption.

NAME	$\|\boldsymbol{X}\|$	$\|\boldsymbol{P}\|$	$\|\boldsymbol{W}\|$	CPT
ADAPT_T1	120	26	136	1504
ADAPT_T2	671	107	789	13281
ADAPT_P1	172	33	224	4182
ADAPT_P2	494	99	602	10973
mini-ADAPT	18	3	15	108
Pigs	441	296	592	8427

Table 2: Bayesian networks used in experiments. The $|\boldsymbol{P}|$ column presents the number of potentially shared nodes, with actually shared nodes $\boldsymbol{S} \subseteq \boldsymbol{P}$. The CPT column denotes the total number of parameters in the conditional probability tables.

$\boldsymbol{Y}$ with equal CPTs are deemed sharable. In most cases, there were multiple sets of nodes $\boldsymbol{Y}_1$, ..., $\boldsymbol{Y}_k$ with $|\boldsymbol{Y}_i| \geq 2$ for $k \geq i \geq 1$. When multiple sets were available, the largest set was selected for experimentation, as shown in Section 5.2's pseudo-code.

Metrics. After an EM trial converged to an estimate $\hat{\boldsymbol{\theta}}$, we collected the following three *metrics*:

1. number of iterations t' needed to converge to $\hat{\boldsymbol{\theta}}$,
2. log-likelihood $\ell\ell$ of $\hat{\boldsymbol{\theta}}$, and
3. error: distance between $\hat{\boldsymbol{\theta}}$ and the original $\boldsymbol{\theta}^*$ (see (1) for the definition).

To provide reliable statistics on mean and standard deviation, many randomly initialized EM trials were run.

Software. Among the available software implementations of the EM algorithm for BNs, we have based our work on LibDAI [23].[13] LibDAI uses factor graphs for its internal representation of BNs, and has several BN inference algorithms implemented. During EM, the exact junction tree inference algorithm [16] was used, since it has performed well previously [19].

5.2 VARYING NUMBER OF SHARED NODES

Here we investigate how varying the number of shared nodes impacts EM. A set of hidden nodes $\boldsymbol{H}$ was created for each BN by selecting

$$\boldsymbol{H} = \arg\max_{\boldsymbol{Y}_i \in \boldsymbol{Y}} |\boldsymbol{Y}_i|,$$

where $\boldsymbol{Y}$ is a sharing set partition for BN nodes $\boldsymbol{X}$ (see Section 4.1). In other words, each experimental BN had its largest set of shareable nodes hidden, giving $n_{\boldsymbol{H}} = 12$ nodes for ADAPT_T1, $n_{\boldsymbol{H}} = 66$ nodes for ADAPT_T2, $n_{\boldsymbol{H}} = 32$ nodes for ADAPT_P1, and $n_{\boldsymbol{H}} = 145$ nodes for Pigs.

[13]`www.libdai.org`

The following *gradual sharing method* is used to vary sharing. Given a fixed set of hidden nodes $\boldsymbol{H}$ and an initially empty set of shared nodes $\boldsymbol{S}$:

1. Randomly add $\Delta n_{\boldsymbol{S}} \geq 1$ hidden nodes that are not yet shared to the set of shared nodes $\boldsymbol{S}$. Since we only have a single sharing set, this means moving $\Delta n_{\boldsymbol{S}}$ nodes from the set $\boldsymbol{H} \setminus \boldsymbol{S}$ to the set $\boldsymbol{S}$.
2. Perform m sharing EM trials in this configuration, and record the three metrics for each trial.
3. Repeat until all hidden nodes are shared; that is, $\boldsymbol{S} = \boldsymbol{H}$.

Using the gradual sharing method above, BN nodes were picked as hidden and then gradually shared. When increasing the number of shared nodes, the new set of shared nodes was a superset of the previous set, and a certain number of EM trials was performed for each set.

5.2.1 One Network

For ADAPT_T2, $m = 200$ samples were generated and $n_{\boldsymbol{H}} = 66$ nodes were hidden. We used, in the gradual sharing method, $\Delta n_{\boldsymbol{S}} = 4$ from $n_{\boldsymbol{S}} = 2$ to $n_{\boldsymbol{S}} = 66$ (every hidden node was eventually set as shared).

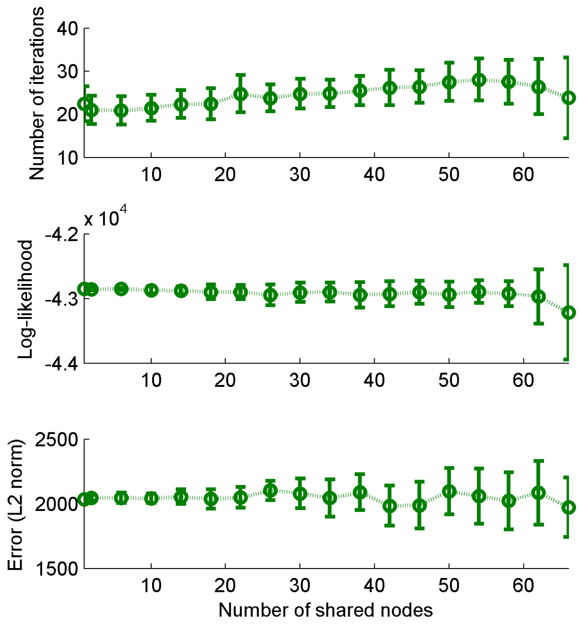


Figure 2: The number of iterations (top), log-likelihood or $\ell\ell$ (middle), and error (bottom) for a varying number of shared nodes $n_{\boldsymbol{S}}$ (along the x-axis) for the BN ADAPT_T2. Here, $n_{\boldsymbol{H}} = 66$ nodes are hidden. Shared nodes are a random subset of the hidden nodes, so $n_{\boldsymbol{S}} \leq n_{\boldsymbol{H}}$.

ADAPT_T1

	Iterations		Likelihood		Error	
Size	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$
25	-0.937	-0.135	0.997	0.882	-0.068	0.888
50	-0.983	-0.765	0.992	0.941	0.383	0.988
100	-0.825	0.702	0.494	0.941	0.786	0.985
200	0.544	0.926	-0.352	0.892	0.517	0.939
400	0.810	0.814	-0.206	0.963	0.863	0.908

ADAPT_T2

	Iterations		Likelihood		Error	
Size	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$
25	0.585	0.852	0.749	0.842	0.276	0.997
50	0.811	0.709	0.377	0.764	0.332	0.981
100	0.772	0.722	-0.465	0.855	-0.387	0.992
200	0.837	0.693	-0.668	0.788	-0.141	0.989
400	0.935	0.677	-0.680	0.784	-0.0951	0.987

ADAPT_P1

	Iterations		Likelihood		Error	
Size	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$
25	-0.769	-0.0818	-0.926	0.278	-0.107	0.774
50	-0.864	-0.0549	-0.939	0.218	-0.331	0.188
100	-0.741	0.436	-0.871	0.678	-0.458	0.457
200	-0.768	0.408	-0.879	0.677	-0.196	0.700
400	-0.665	0.560	-0.862	0.681	0.00444	0.653

PIGS

	Iterations		Likelihood		Error	
Size	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$	$r(\mu)$	$r(\sigma)$
25	0.954	0.763	0.970	0.614	0.965	0.887
50	0.966	0.956	0.137	0.908	0.740	0.869
100	-0.922	-0.0167	-0.994	-0.800	-0.893	0.343
200	-0.827	-0.394	-0.985	-0.746	-0.577	0.0157
400	-0.884	-0.680	-0.942	-0.699	-0.339	0.285

Table 3: Values for Pearson's r for the correlation between the number of shared nodes and statistics for these metrics: number of iterations, log-likelihood ($\ell\ell$), and error. $r(\mu)$ defines the correlation between number of shared nodes and the mean of the metric, while $r(\sigma)$ defines the correlation for the standard deviation.

Figure 2 summarizes the results from this experiment. Here, $n_{\boldsymbol{S}}$ is varied along the x-axis while the y-axis shows statistics for different metrics in each of the three sub-plots. For example, in the top plot of Figure 2, each marker is the mean number of iterations (μ), and the error bar is +/- one standard deviation (σ). The main trends[14] in this figure are: parameter sharing increased the mean number of iterations required for EM and slowly decreased the mean $\ell\ell$. Increasing the number of shared nodes resulted in a corresponding increase in standard deviation for the number of iterations, $\ell\ell$, and error of the BN ADAPT_T2. For standard deviation of error, this is in line with our analysis in Section 4.4.

5.2.2 Multiple Networks

We now investigate how varying the number of shared nodes impacts EM for several BNs, specifically the correlation between the number of parameters shared and the mean μ and standard deviation σ for our three metrics. To measure correlation, we use Pearson's sample correlation coefficient r:

$$r(X,Y) = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{(m-1)s_x s_y}, \tag{10}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of two random variables X and Y, and s_x and s_y are the sample standard deviations of X and Y respectively. Here, (10) measures the correlation between m samples from X and Y.[15]

The number of samples, m, refers to the number of (X,Y) sharing samples we have for this correlation analysis (and not the number of samples used to learn BN parameters). In all these experiments, we do a trial for a number of shared nodes, giving several (X,Y) pairs. Consequently, each number of shared nodes tested would be an X, and the metric measured would be a Y. For example, if we use $n_{\boldsymbol{S}} \in \{2,4,6,8,10\}$ shared nodes, then $m=5$.

We now tie $r(X,Y)$ in (10) to the $r(\mu)$ and $r(\sigma)$ used in Table 3. In Table 3, μ and σ show the mean and standard deviation, respectively, for a metric. Thus, $r(\mu)$ is the correlation of the number of shared nodes and the mean likelihood of a metric, while $r(\sigma)$ is the correlation of the number of shared nodes and the standard deviation of likelihood of a metric.

Figure 2 helps in understanding exactly what is being correlated, as μ and σ for all three metrics are shown for the BN ADAPT_T2. In the top plot, $r(\mu)$ is the correlation between the number of shared nodes (x-axis) and the mean number of iterations (y-axis). In other words, the mean $y_i = \mu_i$ is for a batch of 50 trials of EM. The mean $\bar{y}$ used in Pearson's r is, in this case, a mean of means, namely the mean over 50-EM-trials-means over different numbers of shared nodes.

Table 3 summarizes the experimental results for four BNs. In this table, a positive correlation implies that parameter sharing increased the corresponding metric statistic. For example, the highest correlation between number of shared nodes and mean likelihood is for ADAPT_T1 at 25 samples, where $r(\mu) = 0.997$. This suggests that increasing the number of shared nodes was highly correlated with an increase in the likelihood of EM. Negative coefficients show that increasing the number of shared nodes resulted in a decrease of the corresponding metric statistic.

A prominent trend in Table 3 is the consistently positive correlation between the number of shared nodes

[14] We say "main trends" because the curves for the metrics mean number of iterations and $\ell\ell$ are in fact reversing their respective trends and dropping close to the maximum of 66 shared nodes.

[15] In this case, X is the independent variable, specifically the number of shared nodes. We treat the metric Y as a function of X. When X is highly correlated with Y, this is expressed in r through extreme (positive or negative) correlation values.

NO SHARING

Observable	Error μ	Error σ	Likelihood μ	Likelihood σ	Iterations μ	Iterations σ
$\boldsymbol{O}_C$	16.325	(-)	-3.047e4	(-)	(-)	(-)
$\boldsymbol{O}_P$	48.38	3.74	-2.009e4	43.94	16.39	4.98
$\boldsymbol{O}_T$	33.25	7.45	-2.492e4	1190	8.65	1.99

SHARING

Observable	Error μ	Error σ	Likelihood μ	Likelihood σ	Iterations μ	Iterations σ
$\boldsymbol{O}_C$	16.323	(-)	-3.047e4	(-)	(-)	(-)
$\boldsymbol{O}_P$	48.56	3.92	-2.010e4	62.87	15.98	4.95
$\boldsymbol{O}_T$	34.12	14.3	-2.629e4	2630	6.59	2.48

Table 4: Comparison of No Sharing (top) versus Sharing (bottom) for different observable node sets $\boldsymbol{O}_C$, $\boldsymbol{O}_P$, and $\boldsymbol{O}_T$ during 600 EM trials for mini-ADAPT.

$n_{\boldsymbol{S}}$ and the standard deviation of error, $r(\sigma)$, for all 4 BNs. This is in line with the analytical result involving $n_{\boldsymbol{S}}$ in (9).

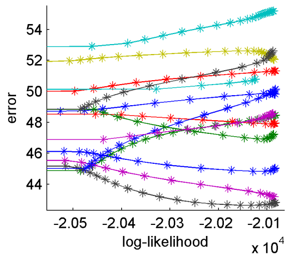
The number of samples was shown to have a significant impact on these correlations. The Pigs network showed a highly correlated increase in the mean number of iterations for 25 and 50 samples. However, for 100, 200, and 400 samples there was a decrease in the mean number of iterations. The opposite behavior is observed in ADAPT_T1, where fewer samples resulted in better performance for parameter sharing (reducing the mean number of iterations), while for 200 and 400 samples we found that parameter sharing increased the mean number of iterations. Further experimentation and analysis may improve the understanding of the interaction between sharing and the number of samples.

5.3 CONVERGENCE REGIONS

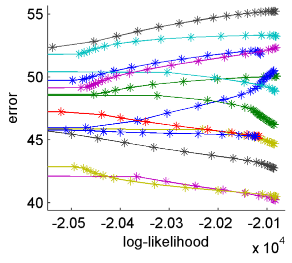
5.3.1 Small Bayesian Networks

First, we will show how sharing influences EM parameter interactions for the mini-ADAPT BN shown in Figure 1 and demonstrate how shared parameters jointly converge.

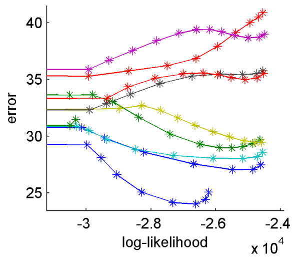
Earlier we introduced $\boldsymbol{O}_P$ as observable nodes in a production system and $\boldsymbol{O}_T$ as observable nodes in a testing system. Complementing $\boldsymbol{O}_P$, hidden nodes are $\boldsymbol{H}_P = \{H_B, H_S, V_B, S_B\}$. Complementing $\boldsymbol{O}_T$, hidden nodes are $\boldsymbol{H}_T = \{V_B, S_B\}$. When a node is hidden, the EM algorithm will converge to one among its potentially many convergence regions. For $\boldsymbol{O}_P$, EM had much less observed data to work with than for $\boldsymbol{O}_T$ (see Figure 1). For $\boldsymbol{O}_P$, the health breaker node H_B was, for instance, not observed or even connected to any nodes that were observed. In contrast, $\boldsymbol{O}_T$ was designed to allow better observation of the components' behaviors, and $H_B \in \boldsymbol{O}_T$. From mini-ADAPT, 500 samples were generated. Depending on the observable set used, either $n_{\boldsymbol{H}} = |\boldsymbol{H}_T| = 2$ or $n_{\boldsymbol{H}} = |\boldsymbol{H}_P| = 4$ nodes were hidden, and 600 random EM trials were executed with and without sharing.



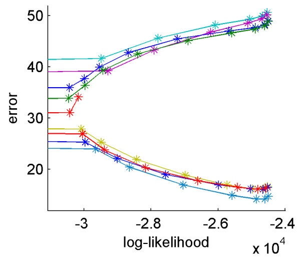
(a) $\boldsymbol{O}_P$ – No Sharing



(b) $\boldsymbol{O}_P$ – Sharing



(c) $\boldsymbol{O}_T$ – No Sharing



(d) $\boldsymbol{O}_T$ – Sharing

Figure 3: The progress of different random EM trials for the mini-ADAPT BN. Both the degree of sharing (No Sharing versus Sharing) and the number of observables nodes ($\boldsymbol{O}_P$ versus $\boldsymbol{O}_T$) are varied.

Table 4 shows results, in terms of means μ and standard deviations σ, for these EM trials. For $\boldsymbol{O}_P$, with $n_{\boldsymbol{H}} = 4$, the means μ of the metrics error, likelihood, and number of iterations showed minor differences when parameter sharing was introduced. The largest change due to sharing was an increase in σ of likelihood. For $\boldsymbol{O}_T$, where $n_{\boldsymbol{H}} = 2$, differences were greater. The μ of likelihood for sharing was lower with over a 2x increase in σ. The μ for error demonstrated only a minor change, but nearly a 2x increase in σ. This is consistent with our analysis in Section 4.4.

Figure 3a and Figure 3c show how log-likelihood or $\ell\ell$ (x-axis) and error (y-axis) changed during 15 EM trials for $\boldsymbol{O}_P$ and $\boldsymbol{O}_T$ respectively. These EM trials were selected randomly among the trials reported on in Table 4. Parameter sharing is introduced in Figure 3b and Figure 3d. For $\boldsymbol{O}_P$, the progress of the EM trials is similar for sharing (Figure 3b) and non-sharing (Figure 3a), although for a few trials in the sharing condition the error is more extreme (and mostly smaller!). This is also displayed in Table 4, where the difference in number of iterations, error, and likelihood was minor (relative to $\boldsymbol{O}_T$). On the other hand, there is a clear difference in the regions of convergence for $\boldsymbol{O}_T$ when parameter sharing is introduced, consistent with the analysis in Section 4.3. Figure 3d shows how the

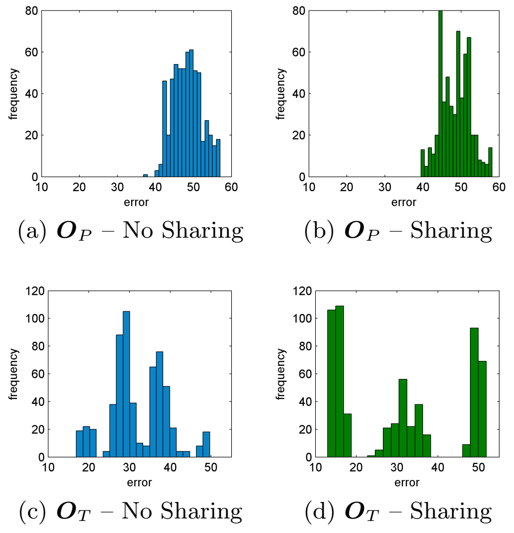


Figure 4: Four 20-bin histograms, for mini-ADAPT, of the error values of 600 randomly initialized EM trials at convergence. Both the degree of sharing (No Sharing versus Sharing) and the number of observables nodes ($\boldsymbol{O}_P$ versus $\boldsymbol{O}_T$) are varied.

EM trials typically followed a path heading to optima far above or far below the mean error, with two of the EM trials plotted converging in the middle region of the error.

Histograms for the 600 EM trials used in Table 4 are shown in Figure 4. The 20-bin histograms show error $err(\hat{\boldsymbol{\theta}})$ at convergence. The $\boldsymbol{O}_P$ and $\boldsymbol{O}_T$ sets are shown without parameter sharing in Figure 4a and Figure 4c, respectively. Parameter sharing is introduced in Figure 4b and Figure 4d. There is an increased σ of error due to parameter sharing for $\boldsymbol{O}_T$. When comparing Figure 4c (No Sharing) and Figure 4d (Sharing), we notice different regions of error for EM convergence due to sharing. Figure 4c appears to show four main error regions, with the middle two being greatest in frequency, while Figure 4d appears to show three regions of error, with the outer two being most frequent. The outer two regions in Figure 4d are further apart than their non-sharing counterparts, showing that parameter sharing yielded a larger range of error for $\boldsymbol{O}_T$, see Section 4.4.

5.3.2 Large Bayesian Networks

Next, large BNs are used to investigate the effects of parameter sharing, using a varying number of shared nodes. The larger ADAPT networks and Pigs were run with 50 EM trials[16] for each configuration of observable nodes, number of samples, and number of shared nodes. Some of the results are reported here.

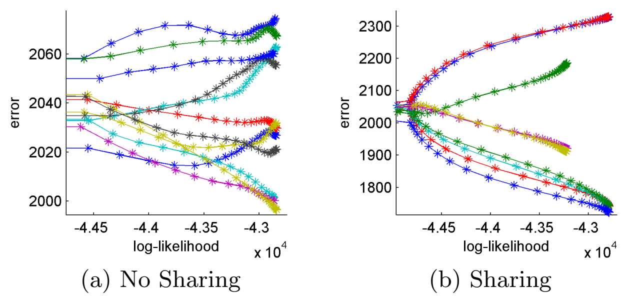


Figure 5: The progress of 15 random EM trials for ADAPT_T2. The No Sharing condition (a) shows more locally optimal convergence regions than Sharing (b), where there appears to be only four locally optimal convergence regions.

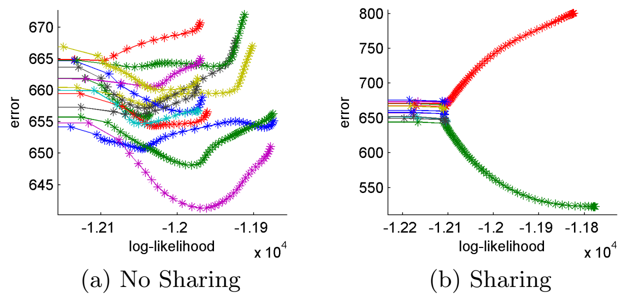


Figure 6: The progress of random EM trials for ADAPT_P1. While the number of EM trials is the same for both conditions, the No Sharing condition (a) clearly shows more local optima than Sharing (b).

Figure 5a shows results for ADAPT_T2 without parameter sharing during 15 EM trials using $n_{\boldsymbol{H}} = 66$ hidden nodes and 200 samples. Figure 5b shows a substantial change in error when the $n_{\boldsymbol{H}} = 66$ hidden nodes were shared. The range of the error for EM is much larger in Figure 5b, while the upper and lower error curves have a symmetric quality. Four regions for the converged error are visible in Figure 5b, with the inner two terminating at a lower $\ell\ell$ than the outer two. The lowest error region of Figure 5b is also lower than the lowest error of Figure 5a, while retaining a similar $\ell\ell$.

Figure 6 uses a smaller ADAPT BN, containing 172 nodes instead of 671 nodes (see Table 2). Here, $n_{\boldsymbol{H}} = 33$ nodes were hidden and 200 samples were used. In several respects, the results are similar to those obtained for ADAPT_T2 and mini-ADAPT. However,

[16]The decrease in number of EM trials performed relative to mini-ADAPT was due to the substantial increase in CPU time required (days to weeks).

Figure 6a shows that EM terminates on different likelihoods, which is not observed in Figure 5a. The error also appears to generally fluctuate more in Figure 6a, whereas the error changes the most during later iterations in Figure 5a. Figure 6b applies parameter sharing to the $n_{\boldsymbol{H}} = 33$ hidden nodes. A symmetric effect is visible between high and low error, reflecting the analysis in Section 4. Of the 15 trials shown in Figure 6b, two attained $\ell\ell > -1.2e^{-4}$, while the rest converged at $\ell\ell \approx -1.21e^{-4}$. Additionally, the $\ell\ell$s of these two trials were greater than any of the non-sharing $\ell\ell$s shown in Figure 6a.

6 CONCLUSION

Bayesian networks have proven themselves as very suitable for electrical power system diagnostics [19,20, 30–33]. By compiling Bayesian networks to arithmetic circuits [4,5], a broad range of discrete and continuous faults can be handled in a computationally efficient and predictable manner. This approach has resulted in award-winning performance on public data from ADAPT, an electrical power system at NASA [30].

The goal of this paper is to investigate the effect, on EM's behavior, of parameter sharing in Bayesian networks. We emphasize electrical power systems as an application, and in particular examine EM for ADAPT Bayesian networks. In these networks, there is considerable opportunity for parameter sharing.

Our results suggest complex interactions between varying degrees of parameter sharing, varying number of hidden nodes, and different dataset sizes when it comes to impact on EM performance, specifically likelihood, error, and the number of iterations required for convergence. One main point, which we investigated both analytically and empirically, is how parameter sharing impacts the error associated with EM's parameter estimates. In particular, we have found analytically that the error variance increases with the number of shared parameters. Experiments with several BNs, mostly for fault diagnosis of electrical power systems, are in line with the analysis. The good news here is that there is, in the sharing case, smaller error some of the time.

Further theoretical research to better understand parameter sharing is required. Since parameter sharing was demonstrated to perform poorly in certain cases, further investigations appear promising. Parameter sharing sometimes reduced the number of EM iterations required for parameter learning, while at other times the number of EM iterations increases. Improving the understanding of the joint impact of parameter sharing and the number of samples on the number of EM iterations would be useful, for example. Finally, it would be interesting to investigate the connection to object-oriented and relational BNs in future work.

References

[1] E.E. Altendorf, A.C. Restificar, and T.G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of UAI*, volume 5, 2005.

[2] A. Basak, I. Brinster, X. Ma, and O. J. Mengshoel. Accelerating Bayesian network parameter learning using Hadoop and MapReduce. In *Proc. of BigMine-12*, Beijing, China, August 2012.

[3] P.S. Bradley, U. Fayyad, and C. Reina. Scaling EM (Expectation-Maximization) clustering to large databases. *Microsoft Research Report, MSR-TR-98-35*, 1998.

[4] M. Chavira and A. Darwiche. Compiling Bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1306–1312, 2005.

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

[6] C.P. de Campos and Q. Ji. Improving Bayesian network parameter learning using constraints. In *Proc. 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, 2008.

[7] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, (27):94–128, 1999.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Of The Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[9] R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322, 2005.

[10] S.H. Jacobson and E. Yucesan. Global optimization performance measures for generalized hill climbing algorithms. *Journal of Global Optimization*, 29(2):173–190, 2004.

[11] W. Jank. The EM algorithm, its randomized implementation and global optimization: Some challenges and opportunities for operations research. In F. B. Alt, M. C. Fu, and B. L. Golden, editors, *Perspectives in Operations Research: Papers in Honor of Saul Gass 80th Birthday*. Springer, 2006.

[12] W. B. Knox and O. J. Mengshoel. Diagnosis and reconfiguration using Bayesian networks: An electrical power system case study. In *Proc. of the IJCAI-09 Workshop on Self-$\star$ and Autonomous Systems (SAS): Reasoning and Integration Challenges*, pages 67–74, 2009.

[13] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[14] H. Langseth and O. Bangsø. Parameter learning in object-oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1):221–243, 2001.

[15] S.L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201, 1995.

[16] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[17] W. Liao and Q. Ji. Learning Bayesian network parameters under incomplete data with domain knowledge. *Pattern Recognition*, 42(11):3046–3056, 2009.

[18] G.J. McLachlan and D. Peel. *Finite mixture models*. Wiley, 2000.

[19] 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.

[20] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, and S. Uckun. Diagnosing faults in electrical power systems of spacecraft and aircraft. In *Proceedings of the Twentieth Innovative Applications of Artificial Intelligence Conference (IAAI-08)*, pages 1699–1705, Chicago, IL, 2008.

[21] O. J. Mengshoel, S. Poll, and T. Kurtoglu. Developing large-scale Bayesian networks by composition: Fault diagnosis of electrical power systems in aircraft and spacecraft. In *Proc. of the IJCAI-09 Workshop on Self-$\star$ and Autonomous Systems (SAS): Reasoning and Integration Challenges*, pages 59–66, 2009.

[22] O.J. Mengshoel, D.C. Wilkins, and D. Roth. Initialization and restart in stochastic local search: Computing a most probable explanation in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(2):235–247, 2011.

[23] J.M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010.

[24] S. Natarajan, P. Tadepalli, E. Altendorf, T.G. Dietterich, A. Fern, and A.C. Restificar. Learning first-order probabilistic models with combining rules. In *ICML*, pages 609–616, 2005.

[25] R.S. Niculescu, T.M. Mitchell, and R.B. Rao. Bayesian network learning with parameter constraints. *The Journal of Machine Learning Research*, 7:1357–1383, 2006.

[26] R.S. Niculescu, T.M. Mitchell, and R.B. Rao. A theoretical framework for learning Bayesian networks with parameter inequality constraints. In *Proc. of the 20th International Joint Conference on Artifical Intelligence*, pages 155–160, 2007.

[27] 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, 2007.

[28] M. Ramoni and P. Sebastiani. Robust learning with missing data. *Machine Learning*, 45(2):147–170, 2001.

[29] E. Reed and O.J. Mengshoel. Scaling Bayesian network parameter learning with expectation maximization using MapReduce. *Proc. of Big Learning Workshop on Neural Information Processing Systems (NIPS-12)*, 2012.

[30] B. Ricks and O. J. Mengshoel. Diagnosis for uncertain, dynamic and hybrid domains using Bayesian networks and arithmetic circuits. *International Journal of Approximate Reasoning*, 55(5):1207–1234, 2014.

[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 *In Proc. of the Annual Conference of the Prognostics and Health Management 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] B. W. Ricks and O. J. Mengshoel. Diagnosing intermittent and persistent faults using static Bayesian networks. In *Proc. of the 21st International Workshop on Principles of Diagnosis (DX-10)*, Portland, OR, 2010.

[34] A. Saluja, P.K. Sundararajan, and O.J. Mengshoel. Age-Layered Expectation Maximization for parameter learning in Bayesian Networks. In *Proceedings of Artificial Intelligence and Statistics (AIStats)*, La Palma, Canary Islands, 2012.

[35] B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45(3):279–299, 2001.

[36] S. Watanabe. Algebraic analysis for nonidentifiable learning machines. *Neural Computation*, 13(4):899–933, 2001.

[37] C.F. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[38] Z. Zhang, B.T. Dai, and A.K.H. Tung. Estimating local optimums in EM algorithm over Gaussian mixture model. In *Proc. of the 25th international conference on Machine learning*, pages 1240–1247. ACM, 2008.

Using Bayesian Attack Detection Models to Drive Cyber Deception

James H. Jones, Jr.
Department of Electrical and Computer Engineering
George Mason University
Fairfax, VA 22030

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

Abstract

We present a method to devise, execute, and assess a cyber deception. The aim is to cause an adversary to believe they are under a cyber attack when in fact they are not. Cyber network defense relies on human and computational systems that can reason over multiple individual evidentiary items to detect the presence of meta events, i.e., cyber attacks. Many of these systems aggregate and reason over alerts from Network-based Intrusion Detection Systems (NIDS). Such systems use byte patterns as attack signatures to analyze network traffic and generate corresponding alerts. Current aggregation and reasoning tools use a variety of techniques to model meta-events, among them Bayesian Networks. However, the inputs to these models are based on network traffic which is inherently subject to manipulation. In this work, we demonstrate a capability to remotely and artificially trigger specific meta events in a potentially unknown model. We use an existing and known Bayesian Network based cyber attack detection system to guide construction of deceptive network packets. These network packets are not actual attacks or exploits, but rather contain selected features of attack traffic embedded in benign content. We provide these packets to a different cyber attack detection system to gauge their generalizability and effect. We combine the deception packets' characteristics, the second system's response, and external observables to propose a *deception model* to assess the effectiveness of the manufactured network traffic on our target. We demonstrate the development and execution of a specific deception, and we propose the corresponding deception model.

Key words: Cyber Deception, Cyber Attack, Bayesian Model, Deception Model, Intrusion Detection System

1. INTRODUCTION

Network-based Intrusion Detection Systems (NIDS) are essentially granular sensors. Their measurements consist of computer network traffic, sometimes at the packet level, which matches signatures of known cyber attack activity. For a typical network, the individual data points are numerous and require aggregation, fusion, and context to acquire meaning. This reasoning may be accomplished through the use of cyber attack detection models, where the NIDS data points represent evidence and specific cyber attacks or classes of attacks represent hypotheses. Modeling approaches, including Bayesian Networks, have been applied in the past, are an active research area, and are in use today in deployed systems.

The input to a NIDS sensor is network traffic, which is inherently uncertain and subject to manipulation. Prior research has exploited this fact to create large numbers of false NIDS alerts to overwhelm or disable the backend processing systems. In this work, we leverage knowledge of the backend cyber attack models to craft network traffic which manipulates the inputs and hence the outputs of both known and unknown models. With a small number of packets and no actual cyber attack, we are able to create the false impression of an active attack.

In this work, we describe a general approach to network-based offensive cyber deception, and we demonstrate an implementation of such a deception. We use an existing cyber attack detection model to guide the development of deception traffic, which is then processed by a second and distinct cyber attack detection model. Finally, we propose a deception model to assess the effectiveness of the deception on a target. Future work will expand and automate the generation of deceptive network packets and further develop the deception model.

2. RELATED WORK

Deception has been a staple of military doctrine for thousands of years, and a key element of intelligence agency activities since they took their modern form in World War II. From Sun Tzu 2,500 years ago (Tzu, 2013), to *Operation Mincemeat* in 1943 (Montagu and Joyce, 1954), to the fictional operation in the 2007 book *Body of Lies* (Ignatius, 2007), one side has endeavored to mislead the other through a variety of means and for a variety of purposes. The seminal work of Whaley and Bell (Whaley, 1982; Bell and Whaley, 1982; Bell and Whaley, 1991), formalized and in some ways defended deception as both necessary and possible to execute without "self contamination".

Deception operations have naturally begun to include the cyber domain, although the majority of this work has been on the defensive side. Fred Cohen suggested a role for deception in computer system defense in 1998 (Cohen, 1998) and simultaneously released his honeypot implementation called The Deception Toolkit (Cohen, 1998). Honeypots are systems meant to draw in attackers so they may be distracted and/or studied. The Deception Toolkit was one of the first configurable and dynamic honeypots, as opposed to prior honeypots which were simply static vulnerable systems with additional administrator control and visibility. Other honeypot implementations have followed, and they remain a staple of defensive cyber deception. Neil Rowe (2003)(2007), his colleague Dorothy Denning (Yuill, Denning, and Feer, 2006), and students (Tan, 2003) at the Naval Postgraduate School have been researching defensive cyber deception for several years. Extending their early work identifying key disruption points of an attack, they propose deception by resource denial, where some key element of an active attack vector is deceptively claimed to be unavailable. Such an approach stalls the attacker while the activity can be analyzed and risks mitigated. Other defensive cyber deception approaches include masking a target's operating system (Murphy, McDonald, and Mills, 2010), and actively moving targets within an IP address and TCP port space (Kewley, et al., 2001), later labeled "address shuffling". Crouse's (2012) comparison of the theoretical performance of honeypots and address shuffling remains one of the few rigorous comparisons of techniques. Until recently, most defensive cyber deception involved theoretical work or small proofs of concept. However, in 2011, Ragsdale both legitimized defensive cyber deception and raised the bar when he introduced DARPA's Scalable Cyber Deception program (Ragsdale, 2011). The program aims to automatically redirect potential intruders to tailorable decoy products and infrastructures in real time and at an enterprise scale.

By comparison, offensive cyber deception has been discussed only briefly in the literature, often as a secondary consideration. For example, honeypots are typically a defensive tool but may be used in an offensive sense to provide disinformation to an adversary. Similarly, deliberately triggering an adversary's network defenses to overwhelm or disable equipment, software, or operators was discussed openly in 2001 (Patton, Yurcik, and Doss 2001) but proposed as cover for other attacks rather than to effect a deception. A small number of offensive cyber deception implementations have been presented, such as the D^3 (Decoy Document Distributor) system to lure malicious insiders (Bowen, Hershkop, Keromytis, and Stolfo, 2009) and the ADD (Attention Deficit Disorder) tool to create artificial host-based artifacts in memory to support a deception (Williams and Torres, 2014). While offensive cyber warfare has entered the public awareness with the exposure of activity based on tools such as Stuxnet, Flame, and Shamoon, offensive cyber deception remains the subject of limited open research and discussion.

Our deception work focuses on aggregation and reasoning tools applied to Network Intrusion Detection Systems (NIDS). These reasoning tools emerged from the inundation of alerts when NIDS sensors were first deployed on enterprise networks. Such tools may simply correlate and aggregate alerts or may model cyber attack and attacker behavior to reason over large quantities of individual evidentiary items and provide assessments of attack presence for human operators to review. Such reasoning models are abundant in the literature and in operational environments, having become indispensable to cyber defenders and remaining an active research area. Initial work on correlating and aggregating NIDS alerts appeared in 2001 (Valdes and Skinner, 2001). A few years later, a body of research emerged which correlated NIDS events with vulnerability scans to remove irrelevant alerts, for example (Zhai, et al., 2004). More advanced reasoning models emerged a few years later, attempting to capture attack and attacker behavior using various techniques. For example, Zomlot, Sundaramurthy, Luo, Ou, and Rajagopalan (2011) applied Dempster-Shafer theory to prioritize alerts, and Bayesian approaches remain popular (Tylman, 2009; Hussein, Ali, and Kasiran, 2012; Ismail, Mohd and Marsono, 2014). Jones and Beisel (2014) developed a Bayesian approach to reasoning over custom NIDS alerts for novel attack detection. A prototype of this approach, dubbed Storm, was used as the base model in this work.

Our research builds on this rich body of prior work, merging the basic precepts of deception, manipulation of network traffic, and model-based reasoning into an offensive cyber deception capability. We use existing *detection* models to derive corresponding *deception* models, demonstrating the ability to deceive an adversary on their own turf and causing them to believe they are under attack when in fact they are not. This capability may be used offensively to create an asymmetry between attackers generating small numbers of deception packets and targets investigating multiple false leads, and defensively to improve the sensitivity, specificity, and deception recognition of existing cyber attack detection tools.

3. BACKGROUND

Signature-based Network Intrusion Detection Systems operate by matching network traffic to a library of patterns derived from past attacks and known techniques. Matches generate alerts, often one per packet, which are saved and sent to a human operator for review. The basic idea of intrusion detection is attributed to Anderson (1980). Todd Heberlein introduced the idea of network-based intrusion detection in 1990 (Heberlein, et al., 1990). Only when processing capabilities caught up to network bandwidth did the market take off in the late 1990s and early 2000s. Unfortunately, early enterprise deployments generated massive numbers of alerts, to the point that human operators could not possibly process them all. Two capabilities came out of this challenge: (1) correlation between NIDS data and other enterprise sources, such as vulnerability scanning data, e.g., see ArcSight[i], and (2) aggregators which model cyber attacks and use NIDS alerts as individual evidentiary items, only alerting a human when multiple aspects of an attack are detected, e.g., see Hofmann and Sick (2011). As noted in Section 2, many modeling approaches have been applied to the aggregation and context problem for more than a decade, and the area remains one of active research, e.g., Boukhtouta, et al (2013).

For the base model in this project, we used an existing cyber attack detection model previously developed by one of the authors. The implementation of this model, called Storm, uses network traffic observables and a Bayesian Network reasoning model to detect a system compromise resulting from known and novel cyber attacks. The system ingests raw network traffic via a live network connection or via traffic capture files in libpcap[ii] format. Individual packets and groups of packets are assessed against signatures associated with cyber attack stages, such as reconnaissance, exploitation, and backdoor access (see Figure 1). Prior to ingest by the model, saturation and time decay functions are applied to packets which match signatures so that model output reflects the quantity and timing of packets. Ingest and model updates occur in real time as packets are received and processed. Packet capture and signature matching is implemented in C++ using libpcap on a Linux (Ubuntu) system. Matching packet processing, model management, and the user interface are implemented in Java, and the Bayesian Network is implemented with Unbbayes[iii]. Packets are passed to the model via a TCP socket so that packet processing and reasoning may be performed on different systems, although we used a single server for our testing.

The Storm system reasons over indirect observables resulting from the necessary and essentially unavoidable steps necessary to effect a system compromise. This underlying cyber attack process is shown in Figure 1 below, where a typical attack progress downward from State 1 (S1) to State 7 (S7). Observables are created at each state and transition. The existing Storm implementation contains one or more observable signatures for each of the six state transitions (T1-T6 in the figure).

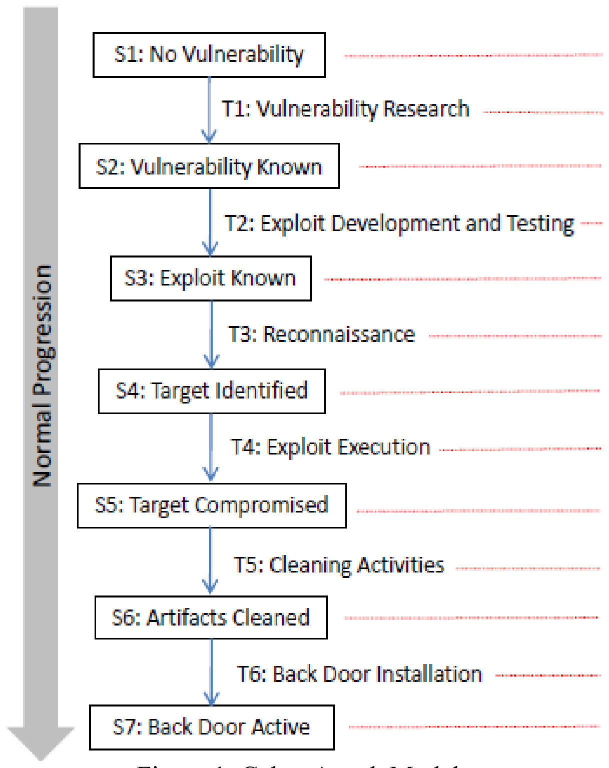


Figure 1: Cyber Attack Model

The reasoning model, shown in Figure 2, was derived from expert knowledge and consists of 20 signature evidence nodes (leaves labeled Tnn), three derived evidence nodes (labeled Mn), two protocol aggregation nodes (labeled Port80 and Port25), six transition aggregation nodes (labeled Tn), and a root node (labeled Compromise). Signature hits are processed and used to set values for the Tnn and Mn nodes. As implemented, one model is instantiated for each cyber attack target (unique target IP address). Model instances are updated whenever new evidence is received or a preconfigured amount of time has passed, and the root node values are returned as Probability of Compromise given Evidence, P(C|E), for each target.

The theory behind the model, further explained by Jones and Beisel (2014), is to recognize observables created when a cyber attack transitions to a new state. For example, when transitioning to the exploit stage, packets with NOP instructions (machine code for "do nothing" and used in buffer overflow type attacks) or shell code elements (part of many exploit payloads) are often seen. As such, the Storm signature rules for detecting observables are not specific to particular attacks, but rather represent effects common to actions associated with cyber attack stages in general. Taken individually, signature matches do not imply an attack. However, when aggregated and combined in context by the reasoning model, an accurate assessment of attack existence may be produced. The system is able to detect novel attacks, since signatures are based on generic cyber attack state transitions instead of specific attack signatures.

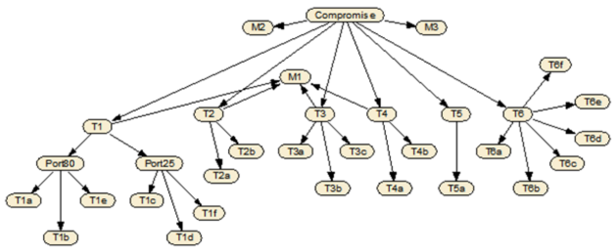


Figure 2: Bayes Net Cyber Attack Detection Model

For most environments, network traffic is insecure and untrusted. Most networks and systems carry and accept traffic that may be both unencrypted and unauthenticated. As such, nearly anyone can introduce traffic on an arbitrary network or at least modify traffic destined for an arbitrary network or system. While full arbitrary packet creation, modification, and introduction is not generally possible, the ability to at least minimally manipulate packets destined for an arbitrary network or system is inherent in the design and implementation of the Internet. Packet manipulation is straightforward using available tools like Scapy[iv], requiring only knowledge of basic object oriented concepts and an understanding of the relevant network protocols.

It is the combination of an ability to manipulate network traffic and models which use network traffic as evidentiary inputs which we exploit in our work.

4. METHODOLOGY

Our goal for this project is to establish the viability of manipulating an adversary's perception that they are the target of a cyber attack when in fact they are not. We begin by conducting a sensitivity analysis of a known cyber attack detection model to identify candidate influence points. We design, construct, and inject network packets to trigger evidence at a subset of these influence points. We construct a corresponding deception model to assess the likelihood that our deception is effective. This derivative deception model combines the impact of our deception packets with other factors, such as the ease with which a target may invalidate the deception packets, the prevalence of alternative explanations for detection system alarms, and external indicators of the target's response activities. The impact of our deception packets is estimated by their effect on an alternative cyber attack detection system, in this case Snort[v]. See Figure 3 for an overview of this process.

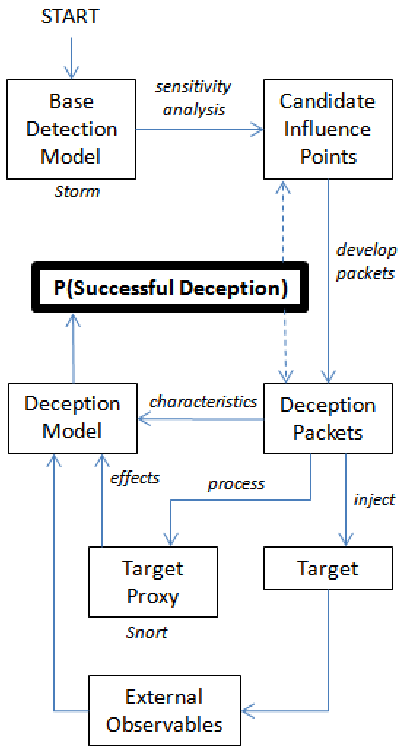


Figure 3: Process Overview

The deception model output, P(Successful Deception) is envisioned to be a dynamic value computed in real time as deception packets are delivered to a target and external observables are collected. As the deception operation unfolds and feedback from external observables is incorporated, additional existing deception packets may be injected, or influence points may be examined for additional deception packet development.

Our base detection model is a Bayesian Network (Figure 2) from a test implementation of the Storm cyber attack detection system. A single node sensitivity to findings analysis (from Netica) is summarized in Table 1 for the 20 evidence input nodes. We reviewed this output and the descriptions of each signature to select those which (a) would have high impact on Storm's probability of compromise based on the sensitivity analysis, (b) could be reasonably developed into a deception packet or packets, and (c) could be general enough to be detected by a target's cyber attack detection system, i.e., not Storm. In Table 1, the eight non-gray rows are those that were selected for deception packet development (signatures T5a, T6a, T6b, T6d, T4a, T4b, T1e, and T1f).

Table 1: Storm model sensitivity analysis

Signature	Mutual Info	Variance of Beliefs
T5a	0.03305	0.0011763
T6e	0.02719	0.0008344
T6a	0.02719	0.0008344
T6b	0.02719	0.0008344
T6d	0.02719	0.0008344
T6c	0.02719	0.0008344
T6f	0.02719	0.0008344
T4a	0.01701	0.0004315
T4b	0.01701	0.0004315
T3b	0.01658	0.0003618
T3c	0.00702	0.0001202
T3a	0.00259	0.0000372
T1a	0.00002	0.0000002
T1b	0.00002	0.0000002
T1e	0.00002	0.0000002
T1c	0.00002	0.0000002
T1d	0.00002	0.0000002
T1f	0.00002	0.0000002
T2a	0.00001	0.0000002
T2b	0.00001	0.0000002

Our test environment consisted of a Storm implementation running on Ubuntu 11.10 and a packet manipulation host running BackTrack5[vi]. We captured normal network traffic in a test environment and processed the traffic through the Storm system to confirm that no attacks were detected. Storm, like most NIDS implementations, has the ability to ingest live network traffic as well as network traffic capture files without loss of accuracy or fidelity. Traffic was captured using the open source Wireshark[vii] tool, saved as a pcap file, then ingested by Storm. We labeled this original packet capture file "clean" and used it as the basis for our subsequent packet manipulations.

We used Scapy on the BackTrack5 instance to craft deception packets. Scapy is an open source Python based packet crafting and editing tool. Packets may be loaded from a pcap file, then manipulated in an environment similar to a Python command shell and written out to a pcap file. Scapy supports creation and modification of any packet field down to the byte level and to include the raw creation and editing of packet data. To create our deception packets, we made minor modifications to packets from the clean set. By minimizing changes, we produced packets that maintained most of the clean session characteristics and so would not be blocked by a firewall or other packet screening device. Also, packets were modified only to the extent necessary to trigger the desired signature, so the modified packets do not contain any actual attacks. The modified packets and associated unmodified session packets, such as session establishment via the TCP 3-way handshake, were exported to a separate pcap file so they could be ingested by the Storm and Snort systems in a controlled manner.

The eight signatures selected for deception and the related deception packets are summarized in Table 2.

Table 2: Signatures and deception packets

Signature T1e **Description**: After 3-way handshake, DstPort=80, payload≠<ASCII> **Explanation**: Abnormal traffic to web server (usually expect GET or POST with ASCII data) **Deception packet**: Inserted non-ASCII (hex > 7F) at beginning of payload for existing HTTP session
Signature T1f **Description**: After 3-way handshake, DstPort=25, payload≠<ASCII> **Explanation**: Abnormal traffic to a mail server (normally we expect plaintext commands) **Deception packet**: Edited HTTP session to use server port 25; inserted non-ASCII (hex > 7F) at beginning of payload
Signature T4a **Description**: Client to server traffic containing 20+ repeated ASCII characters **Explanation**: Buffer overflows often use a long string of ASCII characters to overflow the input buffer **Deception packet**: Inserted 43 "d" (hex 64) characters at the beginning of existing HTTP session payload
Signature T4b **Description**: Client payload contains 20+ identical and consecutive NOP instruction byte patterns **Explanation**: A "NOP sled" is a common technique used in buffer overflow exploits; the sled consists of multiple NOP (No Operation) instructions to ensure that the real instructions fall in the desired range **Deception packet**: Inserted 24 hex 90 (known NOP code) characters at the beginning of existing HTTP session payload

Signature T5a
Description: Client to server traffic if port≠23 and first 100 bytes of payload contains "rm", "rmdir", "rd", "del", "erase"
Explanation: File or directory removal activity
Deception packet: Inserted "rm " (hex 726D20) characters at the beginning of existing HTTP session payload

Signature T6a
Description: First two bytes of client to server payload="MZ"
Explanation: COM, DLL, DRV, EXE, PIF, QTS, QTX, or SYS file transfer for use in a backdoor
Deception packet: Inserted "MZ" (hex 4D5A) and filename "exe" characters at the beginning of existing HTTP session payload

Signature T6b
Description: New Port opened on server; ignore first 500 packets after startup
Explanation: Traffic from a port not previously seen might indicate the opening of a new back door
Deception packet: Edited HTTP session to use server port 25 (ingested after first 500 packets)

Signature T6d
Description: Unencrypted traffic on encrypted port
Explanation: Traffic on encrypted sockets (HTTPS, SMTP with SSL, Secure Shell, etc.) should be encrypted once the session is established.
Deception packet: Inserted ASCII text in an established SSH session

Manual creation of the deception packets required a moderate amount of effort. When crafting deception packets, care must be taken to use carrier traffic which will be passed by a firewall or similar network security gateway while still triggering the desired signature. Flexibility in carrier traffic is signature dependent. For example, some signatures have offset or port dependencies like requiring the traffic to be, or not to be, on port 80 (HTTP), while others are more flexible. Future work will develop an automated deception packet creation capability.

We began by identifying a candidate session for packet modification. For example, we could start with an existing HTTP or HTTPS session and alter packet payload, or we might also alter TCP ports. Payload modification required adjustments to the TCP checksum, IP checksum, and IP length values as well. To create a deception packet set, we loaded the clean pcap file in scapy, made the desired packet modifications, and wrote the resulting packet set to a new pcap file. We then used Wireshark to confirm our modifications and to extract and save only the session of interest as a distinct pcap file. We confirmed our deception packets by processing them with Storm, and later Snort, in a controlled environment.

We tested single occurrences of each signature separately and in a subset of possible combinations. For each individual signature and for selected combinations, we also tested the effects of 10 and 20 signature instances. Finally, for selected signatures, we measured the effect of multiple occurrences for values 1, ..., 25. For all tests, we reset the Storm model, loaded the desired pcap file, and recorded the resulting P(C|E).

We then processed each of the deception pcap files (one per signature) with Snort, separately and in combinations and repetitions.

5. EXPERIMENTAL RESULTS

Each signature pcap file was processed by Storm in quantities of 1, 10, and 20 hits. Storm was reset after each run, that is, reset after a run of one T1e hit, then reset after a run of 10 T1e hits, then reset after a run of 20 T1e hits, etc. Results are recorded in Table 3.

Table 3: Single signature impact on P(C|E) with repetition

Signature		Qty=1	Qty=10	Qty=20
ID	Short Description	P(C|E)	P(C|E)	P(C|E)
T1e	HTTPload!=ASCII	0.00	0.01	0.03
T1f	SMTPload!=ASCII	0.00	0.01	0.03
T4a	Repeated ASCII	0.01	0.05	0.16
T4b	Repeated NOPs	0.01	0.05	0.15
T5a	Cleanup cmds	0.02	0.09	0.27
T6a	Executable load	0.02	0.08	0.22
T6b	New server port	0.02	0.08	0.22
T6d	Unencrypted SSL	0.02	0.08	0.22

The relationships between the quantity of signature hits and P(C|E) in each row indicate that setting the Bayesian Network findings is not a simple True/False assignment. To confirm this behavior, we recorded the effect on P(C|E) of 1, 2, ..., 25 signature hits for signatures T5a and T6a. These results are graphed in Figures 4a and 4b. The curves and apparent inflection points of the graphs indicate that the signature hits are subject to a ramping up requirement at low quantities and a saturation adjustment at high quantities. This is in fact implemented by a pre-processing step in the Storm system and is not actually a part of the Bayesian Network component of Storm.

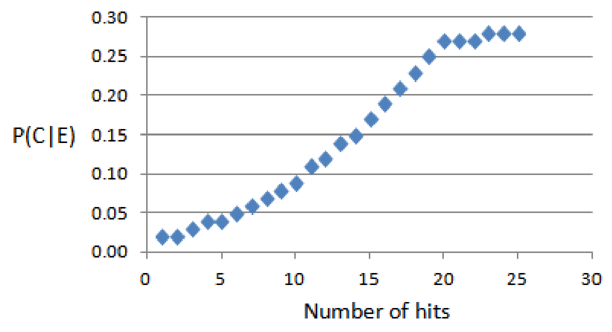


Figure 4a: T5a signature hit effect for n = 1..25

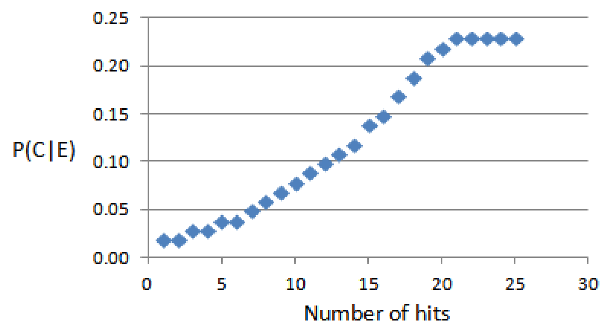


Figure 4b: T6a signature hit effect for n = 1..25

We constructed 11 combinations of signature hits at selected quantities and tested each combination. The results are shown in Tables 4a and 4b below (split for readability).

The results indicate that the desired effect was achieved in two ways: (1) single hits across a wide range of signatures (e.g., tests D, E, and F), and (2) repeated hits on selected signatures (e.g., tests H and K). Assuming a threshold of P(C|E) > 0.75 for alerting, meta-event alerts could be generated with as few as five packets spread across five signatures as in test F, or 40 packets spread across only two different signatures as in test K.

We processed the same eight pcap deception files with Snort. Six of the eight files triggered Snort alerts, as summarized below in Table 5.

Snort Priority 1 are the most severe, Priority 3 the least. The name, classification, and priority of each signature are assigned by the signature author. A base Snort install contains signatures contributed by the Snort developers and the open source community.

Our deception packets produced five Priority 1 alerts, one Priority 2 alert, and one Priority 3 alert. Two deception packets (pcap files for T5a and T6b) did not trigger any Snort alerts.

Table 4a: Effect of signature combinations on P(C|E)

Signature	**Test**					
ID	**A**	**B**	**C**	**D**	**E**	**F**
T1c					1	
T1e					1	
T4a		1	1	1	1	1
T4b		1	1	1	1	
T5a				1	1	1
T6a	1		1	1	1	1
T6b	1		1	1	1	1
T6d	1		1	1	1	1
P(C|E)	**0.10**	**0.02**	**0.46**	**0.86**	**0.89**	**0.79**

Table 4b: Effect of signature combinations on P(C|E)

Signature	**Test**				
ID	**G**	**H**	**I**	**J**	**K**
T1c					
T1e					
T4a	1	10	10		
T4b					
T5a	1	10	10	10	20
T6a	1	10		10	20
T6b					
T6d					
P(C|E)	**0.37**	**0.93**	**0.58**	**0.72**	**0.78**

In a default configuration, and without any subsequent aggregation or alert thresholds, Snort alerts are explicitly linear, meaning that combination and repetition testing produced the obvious results. For example, running any one pcap file N times produces N alerts. Similarly, running the files in combination produced the expected total of alerts, e.g., running the entire set 10 times produced 70 Snort alerts.

Table 5: Snort alerts from deception packets

File	Snort Alerts
T5a	None
T6a	Lotus Notes .exe script source download attempt [Classification: Web Application Attack] [Priority: 1]
T6b	None
T6d	Protocol mismatch [Priority: 3]
	EXPLOIT ssh CRC32 overflow /bin/sh [Classification: Executable code was detected] [Priority: 1]
T4a	MailSecurity Management Host Overflow Attempt [Classification: Attempted Admin Privilege Gain] [Priority: 1]
T4b	SHELLCODE x86 NOOP [Classification: Executable code was detected] [Priority: 1]
T1e	apache chunked enc mem corrupt exploit attempt [Classification: access to potentially vuln web app] [Priority: 2]
T1f	x86 windows MailMax overflow [Classification: Attempted Admin Privilege Gain] [Priority: 1]

6. DECEPTION MODEL

A sample deception model is shown in Figure 5. The model consists of three key parts: (A) the deception packets, (B) external observables indicating a successful deception, and (C) external observables indicating an unsuccessful deception.

Each deception packet, area A in the figure, is assessed for alternative explanations. For example, a byte string that we embed in a JPEG image file may generate a NIDS alert for an unrelated attack, but upon examination will be discounted as a chance occurrence and hence a false alarm. Strong alternative explanations suggest that the target might not interpret the packet as part of an attack and so would weaken the packet node's intended effect on the Successful Deception node. Similarly, each deception packet is assessed for how difficult it will be for a target to invalidate the packet. Again using the example of a byte string embedded in a JPEG image file, if the triggered NIDS alert is an exploit of image viewers, then the packet will be difficult to invalidate. A difficult-to-invalidate packet will have a strong positive influence on the Successful Deception node via the intended effect node.

Processing the original eight deception packets (pcap files) with Snort provides additional parameters for the model. The number, priority, and relevance of Snort alerts are used to build the Conditional Probability Table of the Deception Success node.

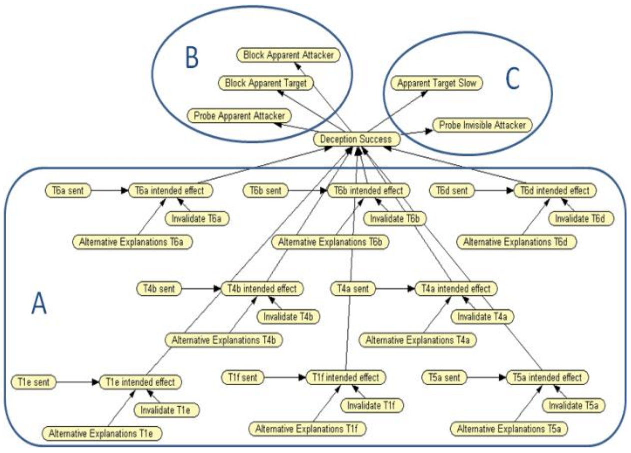


Figure 5: Bayes Net Cyber Deception Model

Area B in the graphic contains three nodes representing external observables which could indicate a successful deception. The "apparent target" and "apparent attacker" are the endpoints of the deception packets. As noted elsewhere, these systems may not send or receive any of the observed traffic, but they will be endpoints from a network monitor's point of view. If a target blocks the apparent target or attacker, or takes the apparent target off-line, then the deception is likely working. Similarly, if the target system operators probe the apparent attacker, then the deception is likely working.

Area C in the graphic contains two nodes for external observables which may indicate that the deception is not working. If the apparent target's response or processing time slows down, this may indicate that the target has added monitoring capabilities in order to trace the source of the deception, although this could also indicate monitoring in response to a perceived successful deception. The other node in area C is the worst case scenario. Although none of the deception packet contents are directly traceable to the actual perpetrators of the deception, probing of the perpetrators systems, especially from the target of the deception, might indicate that the deception has failed and the target suspects the true source of the deception.

The model of Figure 5 is a work in progress at the time of this writing. Preliminary values for the conditional probability tables have been developed but not yet tested or refined. Our work suggests that such a deception model may be developed for other domains where we have some control over the inputs to the base model. Our process in Figure 3 may be generalized by replacing "packets" with "evidence", since packets are simply our mechanism for affecting an evidentiary input node. Generally, a derived deception model consists of the nodes that we directly influence, their estimated effect on the target's model, and external observables.

7. CONCLUSIONS AND FUTURE WORK

We demonstrated the ability to construct network packets which will look similar to normal network traffic, pass through a typical Firewall, trigger specific attack element signatures, and have a controlled impact on a back-end cyber attack detection reasoning model. Further, we proposed a derived deception model to dynamically assess the effectiveness of the cyber deception activities, and we suggested how such a deception model might be constructed for other domains. In support of cyber defense, our work also supports the testing and development of more accurate reasoning models and research geared towards detecting deception.

An apparent limitation of our work is a requirement to know the signatures which trip alerts and are fed to the back end reasoning model. However, this is not necessarily true. While these signatures may be known, as in the case of systems leveraging open source tools like Snort, it is also true that a system designed to detect specific attacks or attacks of a certain class will use similar signatures. The common requirement to derive a discriminatory signature that is as short as possible results in different entities independently producing similar signatures. We have observed this effect in the NIDS domain, where commercial and open source tools have similar signature sets for many attacks. Similarly, we observe this effect in the antivirus and malware detection industry, where different vendors and open source providers frequently generate similar signatures independently. The implication is that we could develop probable signatures for specific attacks or behaviors, then develop deception packets to trip these signatures with a reasonable expectation of successfully affecting a target system using unknown signatures. We partially demonstrated this by processing our Storm-derived packets with Snort.

As noted above, we assert that the use of pcap files is equivalent for our purposes to live network traffic capture and processing. However, it is true that in most live network scenarios we will not be able to put both sides of a TCP session on the wire as we did in this work. Rather, we will have to establish a live session with a target computer and modify subsequent session packets in real time, or we will have to intercept and modify packets between a target and some other system. This is an implementation issue vs. a question of validity, as the results presented here hold regardless of how the deceptive packets are introduced.

Future work will focus on automated deception packet creation, development of delivery mechanisms, and the derived deception model. We created our packets manually based on a review of the target signature and several iterations of trial and error. Our next step is to create deception packets directly from signature descriptions. For example, given a Snort signature file, we could craft multiple deception packets in an automated fashion. A related effort will explore the automation of delivery mechanisms, for example establishing TCP sessions with an internal host and delivering deception packets and injection of deception material into an existing network traffic stream. Author Jones recently led a project to develop a hardware-based inline packet rewriting tool which could be used for such a purpose. Finally, we will continue the development and generalization of deriving deception models from detection models.

References

Anderson, J. P. (1980). Computer security threat monitoring and surveillance (Vol. 17). Technical report, James P. Anderson Company, Fort Washington, Pennsylvania.

Bell, J. B., & Whaley, B. (1982). Cheating: deception in war & magic, games & sports, sex & religion, business & con games, politics & espionage, art & science. St Martin's Press.

Bell, J. B., & Whaley, B. (1991). Cheating and deception. Transaction Publishers.

Boukhtouta, A., Lakhdari, N. E., Mokhov, S. A., & Debbabi, M. (2013). Towards fingerprinting malicious traffic. Procedia Computer Science, 19, 548-555.

Bowen, B. M., Hershkop, S., Keromytis, A. D., & Stolfo, S. J. (2009). Baiting inside attackers using decoy documents (pp. 51-70). Springer Berlin Heidelberg.

Cohen, F. (1998). A note on the role of deception in information protection. Computers & Security, 17(6), 483-506.

Cohen, F. (1998). The deception toolkit. Risks Digest, 19.

Crouse, M. B. (2012). Performance Analysis of Cyber Deception Using Probabilistic Models (Master's Thesis, Wake Forest University).

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. (1990, May). A network security monitor. In Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on (pp. 296-304). IEEE.

Hofmann, A., & Sick, B. (2011). Online intrusion alert aggregation with generative data stream modeling. Dependable and Secure Computing, IEEE Transactions on, 8(2), 282-294.

Hussein, S. M., Ali, F. H. M., & Kasiran, Z. (2012, May). Evaluation effectiveness of hybrid IDs using snort with naive Bayes to detect attacks. In Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on (pp. 256-260). IEEE.

Ignatius, D. (2007). Body of Lies. WW Norton & Company.

Ismail, I., Mohd Nor, S., & Marsono, M. N. (2014). Stateless Malware Packet Detection by Incorporating Naive Bayes with Known Malware Signatures. Applied Computational Intelligence and Soft Computing, 2014.

Jones, J. and Beisel, C. (2014) Extraction and Reasoning over Network Data to Detect Novel Cyber Attacks. National Cybersecurity Institute Journal. Volume 1, Number 1.

Kewley, D., Fink, R., Lowry, J., & Dean, M. (2001). Dynamic approaches to thwart adversary intelligence gathering. In DARPA Information Survivability Conference &amp; Exposition II, 2001. DISCEX'01. Proceedings (Vol. 1, pp. 176-185). IEEE.

Montagu, E., & Joyce, P. (1954). The man who never was. Lippincott.

Murphy, S. B., McDonald, J. T., & Mills, R. F. (2010). An Application of Deception in Cyberspace: Operating System Obfuscation1. In Proceedings of the 5th International Conference on Information Warfare and Security (ICIW 2010) (pp. 241-249).

Patton, S., Yurcik, W., & Doss, D. (2001). An Achilles' heel in signature-based IDS: Squealing false positives in SNORT. Proceedings of RAID 2001.

Ragsdale, D. (2011). Scalable Cyber Deception. Defense Advanced Research Projects Agency, Arlington, Virginia, Information Innovation Office.

Rowe, N. C. (2003, June). Counterplanning deceptions to foil cyber-attack plans. In Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society (pp. 203-210). IEEE.

Rowe, N. (2007, March). Planning cost-effective deceptive resource denial in defense to cyber-attacks. In Proceedings of the 2nd International Conference on Information Warfare & Security (p. 177). Academic Conferences Limited.

Tan, K. L. G. (2003). Confronting cyberterrorism with cyber deception (Doctoral dissertation, Monterey, California. Naval Postgraduate School).

Tylman, W. (2009) Detecting Computer Intrusions with Bayesian Networks. Intelligent Data Engineering and Automated Learning - IDEAL 2009. Lecture Notes in Computer Science Volume 5788, 2009, pp 82-91.

Tzu, S. (2013). The art of war. Orange Publishing.

Valdes, A., & Skinner, K. (2001, January). Probabilistic alert correlation. In Recent Advances in Intrusion Detection (pp. 54-68). Springer Berlin Heidelberg.

Whaley, B. (1982). Toward a general theory of deception. The Journal of Strategic Studies, 5(1), 178-192.

Williams, J., & Torres, A. (2014). ADD - Complicating Memory Forensics Through Memory Disarray. Presented at ShmooCon 2014 and archived at https://archive.org/details/ShmooCon2014_ADD_Complicating_Memory_Forensics_Through_Memory_Disarray. Retrieved June 8, 2014.

Yuill, J., Denning, D. E., & Feer, F. (2006). Using deception to hide things from hackers: Processes, principles, and techniques. North Carolina State University at Raleigh, Department of Computer Science.

Zhai, Y., Ning, P., Iyer, P., & Reeves, D. S. (2004, December). Reasoning about complementary intrusion evidence. In Computer Security Applications Conference, 2004. 20th Annual (pp. 39-48). IEEE.

Zomlot, L., Sundaramurthy, S. C., Luo, K., Ou, X., & Rajagopalan, S. R. (2011, October). Prioritizing intrusion analysis using Dempster-Shafer theory. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 59-70). ACM.

---

[i] http://www.hp.com/go/ArcSight
[ii] http://sourceforge.net/projects/libpcap/
[iii] http://sourceforge.net/projects/unbbayes/
[iv] http://www.secdev.org/projects/scapy/
[v] http://www.snort.org/
[vi] http://www.backtrack-linux.org/downloads/
[vii] http://www.wireshark.org/

Using Bayesian Networks to Identify and Prevent Split Purchases in Brazil

Rommel N. Carvalho,[*] Leonardo J. Sales, Henrique A. da Rocha, and Gilson L. Mendes
Department of Research and Strategic Information (DIE)
Brazil's Office of the Comptroller General (CGU)
SAS, Quadra 01, Bloco A, Edifício Darcy Ribeiro
Brasília, DF, Brazil

Abstract

To cope with society's demand for transparency and corruption prevention, the Brazilian Office of the Comptroller General (CGU) has carried out a number of actions, including: awareness campaigns aimed at the private sector; campaigns to educate the public; research initiatives; and regular inspections and audits of municipalities and states. Although CGU has collected information from various different sources – Revenue Agency, Federal Police, and others –, going through all the data in order to find suspicious transactions has proven to be really challenging. In this paper, we present a Data Mining study applied on real data – government purchases – for finding transactions that might become irregular before they are considered as such in order to act proactively. Moreover, we compare the performance of various Bayesian Network (BN) learning algorithms with different parameters in order to fine tune the learned models and improve their performance. The best result was obtained using the Tree Augmented Network (TAN) algorithm and oversampling the minority class in order to balance the data set. Using a 10-fold cross-validation, the model correctly classified all split purchases, it obtained a ROC area of .999, and its accuracy was 99.197%.

1 Introduction

The Brazilian Office of the Comptroller General (CGU) is the Brazilian central body of the internal control system of the federal executive branch. It has, among its responsibilities, the task of inspecting and auditing the Brazilian Government projects and programs with respect to their legality, results, efficacy and efficiency.

[*]Department of Computer Science (CIC), University of Brasília (UnB), Brasília, DF, Brazil

A primary responsibility of CGU is to prevent and detect government corruption. To carry out this mission, CGU must gather information from a variety of sources and combine it to evaluate whether further action, such as an investigation, is required. One of the most difficult challenges is the information explosion. Auditors must fuse vast quantities of information from a variety of sources in a way that highlights its relevance to decision makers and helps them focus their efforts on the most critical cases. This is no trivial duty. The Growing Acceleration Program (PAC) alone has a budget greater than 250 billion dollars with more than one thousand projects only in the state of Sao Paulo [1]. All of these have to be audited and inspected by CGU – and, in spite of having only three thousand employees. Therefore, CGU must optimize its processes in order to carry out its mission.

In Brazil, all contracts with the private sector must be in accordance with the Law N° 8,666/93, also known as the national Procurement Law. According to [15] procurement is the administrative procedure by which the Public Administration selects the most advantageous proposal for a contract in its interest. From the former definition, the conclusion is that the public interest must always be the objective of the procedure. In terms of purchasing with the use of public money, this means that not only must the winner of the procurement process be the best supplier in terms of the price of the good or service supplied, but also in terms of other objectives of the procurement process.

Corruption can happen in many ways in Public Procurements [16]. The public agent may favor a specific supplier that she happens to know. She may receive, from the bidder, a financial compensation for awarding a contract to that firm. Bidders may collude as to set the results of the procurement. The whole process is susceptible to many forms of corruption, from within and outside the public administration.

The government purchases large quantities of goods and services. It is also the sole purchaser for some goods, such

[1]`http://www.pac.gov.br/`

as hydraulic turbines for large damns. The government spends large quantities of money in the market and is a guaranteed payer. Hence, many firms have a strong interest in negotiating with the public administration. There is a temptation for many suppliers to cheat in the procurement process to find means of being awarded a lucrative government contract.

The Brazilian Procurement Law has as one of its main objectives the curbing of corruption in public purchasing and contracting. Concern over corruption is evident in Brazil's daily press. There are frequent accusations of public administrators who did not abide by the procurement rules, and are accused of favoring a certain supplier or worse, receiving a payment in the process.

When writing the law, legislators included many articles that established penalties for firms or/and public legislators caught in corruption activities. There are two types of penalties stated in Law N° 8,666/93 dealing with this subject. They are administrative actions and penal actions.

Since enforcing the law is difficult [16], legislators will have to find another manner to prevent corruption in public procurement. The question is one of preventing corruption practices, against one of punishing the ones that have already happened.

Therefore, the main objective of this project is to apply Data Mining methods to create models, more specifically to learn Bayesian Network models, that will aid the experts in identifying procurement frauds and, if possible, identify suspicious transactions as soon as possible in order to prevent the fraud from happening.

The structure of this paper is as follows: Section 2 presents the CRoss Industry Standard Process for Data Mining (CRISP-DM) methodology used in this work. Section 3 presents the data used and its underlying semantics. Section 4 presets the models learned as well as their evaluation. Section 5 presents ideas on how the learned model for identifying split purchases will be deployed. Finally, Section 6 presents the conclusion and future work.

2 Methodology

The CRoss Industry Standard Process for Data Mining (CRISP-DM) project developed an industry and tool-neutral data mining process model. Starting from the embryonic knowledge discovery processes used in early data mining projects and responding directly to user requirements, this project defined and validated a data mining process that is applicable in diverse industry sectors. This methodology makes large data mining projects faster, cheaper, more reliable, and more manageable. Even small-scale data mining investigations benefit from using CRISP-DM.

The process model provided by the consortium CRISP-DM can be summarized through the life cycle of the data mining process presented in Figure 1, reproduced from [5]. The outer circle symbolizes the cyclical nature of the process of data mining. Data mining does not end when a solution is achieved. In fact, the lessons learned during a process can be useful in subsequent processes.

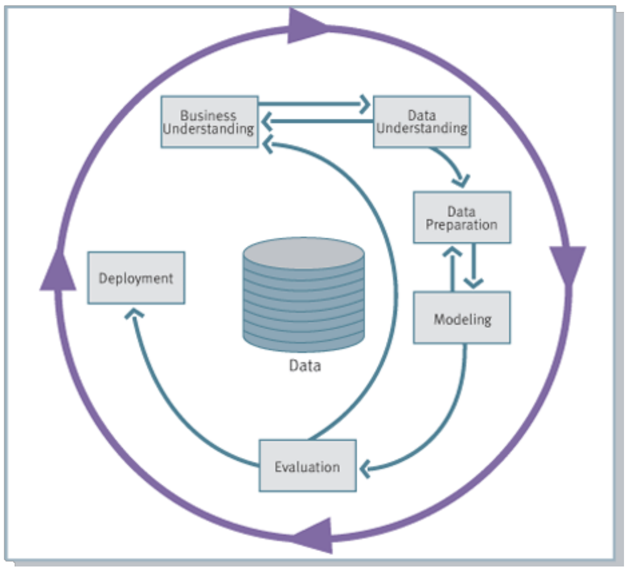


Figure 1: Phases of the CRISP-DM Process Model

The life cycle of a data mining project consists of six phases. The sequence of the phases is not strict. Moving back and forth between different phases is often required. The arrows indicate the most important and frequent dependencies between phases. The phases are [19]:

Business Understanding: This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition, and a preliminary project plan designed to achieve the objectives.

Data Understanding: The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information. There is a close link between Business Understanding and Data Understanding. The formulation of the data mining problem and the project plan require at least some understanding of the available data.

Data Preparation: The data preparation phase covers all activities to construct the final data set (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection, data cleaning, construction of new attributes, and transformation of data for modeling tools.

Modeling: In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques require specific data formats.

Evaluation: At this stage in the project you have built one or more models that appear to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

Deployment: Creation of the model is generally not the end of the project. Usually, the knowledge gained will need to be organized and presented in a way that the customer can use it. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process. In many cases it will be the user, not the data analyst, who will carry out the deployment steps. In any case, it is important to understand up front what actions will need to be carried out in order to actually make use of the created models.

The Business Understanding phase was already covered in the introduction section. The Data Understanding phase will not be covered in detail since the data and its structure is confidential. The following sections will cover the Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment phases.

3 Data Understanding and Preparation

The data set used in this work is related to procurements and contracts of IT services in the Brazilian Federal Government from 2005 to 2010. This data set is actually merged data from different databases (DB) available in different agencies in Brazil.

CGU has been working closely with subject matter experts (SMEs) in order to identify certain fraud topologies, such as:

1. Identify if owners from different companies are actually partners. In the public procurement we expect competition, but if the only two enterprises participating in the procurement have owners that are partners, then it is obvious that there is no competition at all, so this should be prevented/identified.

2. In Brazil if the contract is small enough (8 thousand reais, or roughly around 4 thousand dollars), then there is no need to actually go through the whole procurement process. However, sometimes, they break down a big contract in a lot of small ones. This is called split purchase[2] and it is not allowable by law and should be prevented/identified. *E.g.*, identify a lot of contracts that sums up to more than the threshold with the same objective (buying computers, for instance) in a week or a month.

There are actually more than 20 typologies like these two described by the SMEs that we would like to identify/predict using the data set we have. However, this project focuses on the second one described above.

First the data set was loaded in Excel, then we removed some characters (comma, double quotes, and single quotes), because they were being a problem when loading the file in Weka (see [13] for details about Weka). Then we replaced values like NA, -9, -8, and 0 by "?" to represent a missing value in Weka.

The initial data set had 42 attributes and 70,365 transactions. From Excel we were able to bring that number down to 26. Most of the attributes removed were identification of descriptions that were also available as a different attribute. We decided to keep the description because it is easier to understand what it means. For instance, we kept the name of the state instead of its ID.

The next step was to save the data as a CSV file and load it in Weka. Then we changed the year to nominal and removed rows with missing values in the final price attribute, since we will need a value later on to identify the class.

Before running any algorithm we started looking at the data trying to understand it better. The first thing that we noticed was the range of the values. There were values from cents to hundreds of trillions of dollars. Besides that, there were cases where the proposed unit price was 4 thousand

[2]For more information about split purchases and case examples see `http://guide.iacrc.org/potential-scheme-split-purchases/`.

whereas the final actual price was a dollar. Those are typical cases of data that should not be trusted or considered incorrect.

The number of cases with a really high value is small so they should be carefully analyzed before being thrown away, just to make sure they are not outliers, but actually inconsistent data. For that, we got in contact with the SMEs at CGU and asked them which ones should be ignored. With their consent we removed those transactions that were considered noise. The transactions that could be outliers are being analyzed by the experts in more detail.

Since we are interested in transactions that sum to 8,000 in a given period of time (see typology 2 above), we computed using R[3] the transactions that involved the same institutions on the same month and year that added up to more than 8,000, then we flagged them as irregular transactions[4] (*i.e.*, split purchases).

4 Modeling and Evaluation

The main objective of this work is to try to classify, without looking at the sums, which transactions should be considered suspicious. The reason for that is to avoid waiting for the problem to occur (having several transactions that add to more than 8,000 reais) and to identify the problem as soon as possible.

Before being able to classify we had to discretize the value of the transactions, since we decide to work with learning algorithms that do not support continuous values. We used equal frequency to generate 11 different bins. The number of bins could be different. The only thing we kept in mind was that we did not want it to be binary, nor to have too many bins to the point that we would have just a few transactions in each bin.

Since Bayesian Networks (BNs) have been successfully used in classification problems (*e.g.*, see [4,7,10–12,17,18, 20]), we decided to experiment with different BN learning algorithms in order to classify the transactions as regular or not, in order to identify split purchases as soon as possible, without having to wait for them to actually be confirmed as such (before having several purchases summing up to more than 8 thousand reais). We ran all classifiers with 10-fold cross-validation.

At first we compared the results of Naïve Bayes versus Bayesian Network models. Since the data is unbalanced with 50,911 regular cases and 2,626 irregular cases, we decided to resample the data.

Table 1 presents the performance of the Naïve Bayes and Bayesian Network classifiers using 10-fold cross-validation. We learned models without resampling the data set, oversampling the minority class, undersampling the majority class, and both oversampling the minority class and undersampling the majority class. The metrics used to compare the models are regular false positive (FP) rate, irregular FP rate, accuracy, and Receiver Operating Characteristic (ROC) area.

The standard algorithm used in Weka for BN is K2. K2 uses a hill climbing algorithm restricted by an order on the variables. For more information on K2 see [8, 9]. The parameters used for learning the BN model were all the default values in Weka expect for the maximum number of parents allowed, which we changed from 1 to 2. The `estimator` parameter used, which is independent of which algorithm we use to learn the model, was the `SimpleEstimator`. `SimpleEstimator` is used for estimating the conditional probability tables of a BN once the structure has been learned. It estimates probabilities directly from data. The only configurable parameter for this estimator is `alpha`. Alpha is used for estimating the probability tables and can be interpreted as the initial count on each value. The default value used for `alpha` is .5. As explained in [3], the options for the K2 algorithm in Weka are:

`initAsNaiveBayes` – When set to true (default), the initial network used for structure learning is a Naïve Bayes Network, that is, a network with an arrow from the classifier node to each other node. When set to false, an empty network is used as initial network structure.

`markovBlanketClassifier` – When set to true (default is false), after a network structure is learned a Markov Blanket correction is applied to the network structure. This ensures that all nodes in the network are part of the Markov blanket of the classifier node.

`maxNrOfParents` – Set the maximum number of parents a node in the Bayes net can

[3]R is a free software environment for statistical computing and graphics. For more information see `http://www.r-project.org/`.

[4]Usually we also consider the type of service/product contracted/bought. However, since we limited the scope of our analysis to only computer purchases, all 10 different types of service/product are too broad and really similar (they all basically say that these are computer related purchases without any significant difference between each other). Therefore, we can safely ignore this field when defining split purchases, since they do not add any more useful information. To avoid comparing purchases of different nature, we use the contractor field, since a company that provides software development does not usually sell computer peripherals. Unfortunately, we do not have any other structured field that we could use to improve this proxy classification. Of course we might have cases that purchases identified as split purchases are, in fact, false positives. However, our current alert system would also identify those purchases as split purchases. What we are trying to do is to identify those ”split purchases” as soon as possible, instead of waiting for the whole month cycle.

have. When initialized as Naïve Bayes, setting this parameter to 1 results in a Naïve Bayes classifier. When set to 2, a Tree Augmented Bayes Network (TAN) is learned, and when set to greater than 2, a Bayes Net Augmented Bayes Network (BAN) is learned. By setting it to a value much larger than the number of nodes in the network (the default of 100000 pretty much guarantees this), no restriction on the number of parents is enforced.

`randomOrder` – When set to true, the order of the nodes in the network is random. Default random order is false and the order of the nodes in the data set is used. In any case, when the network was initialized as Naïve Bayes Network, the class variable is first in the ordering though.

`scoreType` – The score type determines the measure used to judge the quality of a network structure. It can be one of Bayes (default), BDeu, Minimum Description Length (MDL), Akaike Information Criterion (AIC), and Entropy.

When oversampling, we used the sample size of 200% to avoid removing known cases of the majority class, which is basically the oversampling of the minority class. The goal was to increase the irregular cases to match the number of regular ones.

We can see that we got much better results with Bayesian Network, when oversampling the minority class. Moreover, the false positive rate for the regular cases, which is our main concern since it represents the number of irregular transactions that were classified as regular, has dropped significantly after oversampling (from .363 to .140 for Naïve Bayes and from .452 to .005 for Bayesian Network). As a result, the FP rate for the irregular cases increased while the accuracy decreased for NB and slightly increased for BN. Nevertheless, since we are more interested in correctly classifying the irregular cases, the result with oversampling is better.

When undersampling, we decided to use the sample size of $(2,626/50,911) * 2 \approx 10\%$. Finally, we tried doing both oversampling and undersampling by keeping the sample size at 100%.

As it can be seen from Table 1, oversampling the minority class gave us the best results both in accuracy and in ROC Area. Besides that, we did have a significant decrease in regular FP rate, which is our main concern. The accuracy without resampling is higher than most of the other results with resampling due to the fact that we had an increase in irregular FP rate and since this is the majority class the number of correctly classified cases is much larger. However, as explained before, the main objective is to obtain low regular FP rate and not necessarily high accuracy, which could only mean that we are getting just the majority class right.

As BN gave the best result, we decided to try different search algorithms. The algorithms used were K2 (again using the default parameters), Hill Climber[5], Tabu Search[6], and Tree Augmented Network[7] (TAN).

The options for the Hill Climber algorithm in Weka are the same as K2, except that it does not have the parameter `randomOrder`. Besides, we have the `useArcReversal`, which when set to true (default is false), the arc reversal operation is used in the search. As in K2, we have used all default parameters, except for the maximum number of parents and the score metric (we used MDL instead of Bayes).

The Tabu Search algorithm in Weka has the same options as Hill Climber. Besides that, it also has `runs`, which sets the number of steps to be performed (default is 10), and `tabuList`, which sets the length of the tabu list (default is 5). As in both K2 and Hill Climber, we have used all default parameters, except for the maximum number of parents and the score metric (we used MDL instead of Bayes).

The TAN algorithm in Weka only has the `markovBlanketClassifier` and `scoreType` options with the same values as K2. We have used the default parameters. Note that the TAN algorithm does not allow the configuration in the maximum number of parents, since it is always 2.

Table 2 shows a summary of the BN classifiers performance using different search algorithms with oversampled data. It is worth noting that we were only able to use the default Bayes metric with the K2 and TAN algorithms. We had to use MDL metric in order to avoid out of memory error with the Hill Climbing and Tabu Search algorithms. This difference might be the reason why K2 and TAN outperforms both Hill Climbing and Tabu Search, since we have tried both K2 and TAN using MDL metric and we did obtain worse results. As it can be seen, TAN search algorithm had the best performance with a ROC area of .999.

Since oversampling resulted in very good results, we decided to also try Synthetic Minority Oversampling TEchnique (SMOTE) [6] to oversample the minority class. We used 1800% to get roughly the same number of transac-

[5]Hill Climber uses a hill climbing algorithm adding, deleting and reversing arcs. The search is not restricted by an order on the variables (unlike K2). For more details see `http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/net/search/global/HillClimber.html`.

[6]Tabu Search algorithm uses tabu search for finding a well scoring BN structure. For more details see [2].

[7]TAN algorithm determines the maximum weight spanning tree and returns a NB network augmented with a tree. For more details see [10].

Table 1: Evaluation of Naïve Bayes (NB) and Bayesian Network (BN) classifiers without resampling, with oversampling, with undersampling, and with both over and undersampling

Metric	NB	BN	NB - Over	BN - Over	NB - Under	BN - Under	NB - Both	BN - Both
Regular FP Rate	.363	.452	.140	**.005**	.169	.068	.146	.012
Irregular FP Rate	.028	**.003**	.048	.036	.082	.054	.054	.045
Accuracy	95.6%	97.5%	90.6%	**97.9%**	87.4%	90.6%	90.0%	97.1%
ROC Area	.931	.973	.975	**.997**	.945	.968	.971	.996

Table 2: BN classifiers performance with oversampling and different number of parents

Metric	BN - K2		BN - HC		BN - Tabu		BN - TAN
Parents	2	3	2	3	2	3	-
Regular FP Rate	.005	.018	.098	.066	.093	.093	**0**
Irregular FP Rate	.036	.038	.082	.067	.039	.039	**.02**
Accuracy	97.94%	97.23%	91.01%	93.35%	93.40%	93.40%	**98.98%**
ROC Area	.997	.996	.976	.985	.988	.988	**.999**

tions in both classes. After we oversampled the data, we discretized it as we did previously (11 bins with equal frequency). However, even though the SMOTE oversampling gave better results for most metrics, the standard oversampling method outperformed the SMOTE method on the regular FP rate, which is the one that we are most interested in. Therefore, we do not present the details of the results here.

Since TAN algorithm presented the best results, we decided to try different values for the alpha parameter in order to improve its performance. Table 3 presents performance for the alpha values .7, .5, .3, .1, and .05. On the one hand, when we increased the default value of .5 to .7, the performance slightly worsened. On the other hand, when we decreased the default value we kept getting slightly better results up to .05, when the performance started worsening again.

Table 4 presents the confusion matrix for the best model we obtained, which was learned using TAN with the value .1 for the `alpha` parameter. As it can be seen, every single split purchase (irregular) was correctly classified. Although the regular FP rate in Table 3 suggests that alpha values of .5, .3, and .05 also classified all irregular instances correctly, the model with .5 alpha missclassified 7 instances (the regular FP rate is shown as 0 due to rounding, since the value is actually .00013). Besides that, even though the irregular FP rate is the same for the models with .1 and .05 alpha, the model with the .05 alpha missclassified 5 instances more than the model with .1 alpha.

In order to try to better understand the relationship between the variables, we generated some association rules using Apriori algorithm [1, 14]. In the first run we used .1 as lower bound min support, 1 as upper bound min support, lift as the metric type, 1.1 as min metric, and 50 as number of rules. For more information on each of these parameters see `http://weka.sourceforge.net/doc.dev/weka/associations/Apriori.html`.

The results were related to variables that were not of interest and even though they had a high confidence (1) and high lift (1.82) they were not describing anything that we did not already know. So we decided to remove those variables from the data set and run the algorithm again to see the new rules it would generate.

We analyzed all 50 new rules generated and they still did not provide any insightful information. Besides that, these new rules were not as "strong" as the previous ones in the sense that the best confidence was .62 and the best lift was 1.11.

The reason for not presenting the rules is two-fold. The first is because, as previously explained, they did not provide any insightful information. The second is because they are confidential and we cannot discuss the contents of the rules. All that can be said is that they were not useful.

Finally, we analyzed the BN that had the best evaluation results (BN – TAN with .1 alpha – standard oversampled data set) to try to better understand the relations between the variables in order to comprehend why it is getting such good results. This could provide insightful information that we were not able to extract from association rules.

By analyzing the structure of the BN we were able to identify some interesting relations associated to the government

Table 3: TAN performance with different values for the alpha parameter

Metric	TAN - .7 alpha	TAN - .5 alpha	TAN - .3 alpha	TAN - .1 alpha	TAN - .05 alpha
Regular FP Rate	.001	**0**	**0**	**0**	**0**
Irregular FP Rate	.023	.020	.018	**.016**	**.016**
Accuracy	98.834%	98.984%	99.108%	**99.197%**	99.192%
ROC Area	**.999**	**.999**	**.999**	**.999**	**.999**

Table 4: Confusion matrix of the best model, which was learned using TAN with .1 alpha

	Predicted Regular	Predicted Irregular
Is Regular	52,670	860
Is Irregular	0	53,544

office responsible for the procurement and also to the winner of the procurements. However, due to the confidentiality nature of the problem we are not allowed to discuss these relations further. Nevertheless, we can comment on the fact that to better understand these relations a more detailed study is necessary by looking at the conditional probability tables on these relations, which was not done in this work due to time constraints.

5 Deployment

As it was explained before, the main goal is to identify suspicious transactions as soon as possible to avoid the occurrence of split purchases.

Split purchases in the data set we worked on happen when a procurement that should be done just once with a value higher than 8,000 is broken down in smaller ones to avoid the normal procurement procedure, allowing the office to hire an enterprise directly.

So, the idea is that we want to detect one or more of these procurements, but that still does not sum up to more than 8,000, as soon as they become suspicious (using the classifier) and act proactively. This will allow a faster response and prevent irregularities in the first place. This means that instead of waiting a whole month to find an irregular transaction, we will be finding suspicious ones (not necessarily irregular ones) and warn/educate/teach the people involved that they should be careful, since breaking down big procurements in small ones is not allowed.

Hopefully, this will make them think twice before continuing with these kinds of transactions, if they are really thinking about doing irregular transactions on purpose. And at the same time this will educate those that are not aware they cannot do these kinds of transactions by law.

Another thing to keep in mind when deploying this classifier is that as we warn/teach the different offices in Brazil about this type of fraud, it is expected that they will decrease the number of irregular transactions associated to this type of fraud. Therefore, it is important to keep track of these numbers in order to evaluate if this expected behavior is actually happening and at what rate.

Besides that, as people learn and change their behavior, it is also expected that our classifier will also need some modifications to cope with these changes. Therefore, it should be expected that every month or so a new classifier should be trained and evaluated to learn the new behavior and analyze its performance.

6 Conclusion

This project shows how Data Mining, BN learning more specifically, can be used to identify and prevent split purchases in Brazil using real data provided by the Brazil's Office of the Comptroller General (CGU).

At first, the idea was to allow the identification of a few different types of irregularities. However, as we started working with the data and creating the different models, it became clear that this is a great effort and after discussing with stakeholders at CGU, we decided to prioritize depth instead of breadth. In other words, it was decided that it was best to choose one type of fraud and do a thoroughly job to develop a model that is good enough to actually deploy it and analyze the results before looking at other types of frauds.

Fortunately, the results paid off in the sense that we were able to generate a model that correctly classified all split purchases and a really high ROC area (.999). The next step is to deploy this classifier in CGUs intelligence system and analyze its impact.

As future work, it is necessary to better understand why the model is capable of getting such good results. By analyzing the structure of the BNs generated, it is possible to understand relations between variables that were previously unknown. A brief analysis showed that some interesting relations were found, but a more detailed analysis is needed.

Even though we have used resampling for dealing with the rare event of split purchases and cross-validation to avoid overfitting, we might have not completely ruled out the possibility that this model might be slightly overfitted. Therefore, before deploying the model, we will gather new data from recent purchases in order to test the performance of our model with new unseen data. Due to time constraints, we have not been able to gather the new data and perform this test yet.

Besides that, there are a lot of different types of frauds that were not analyzed. So, a starting point for future work is to do the same kind of analysis done in this project with different types of fraud.

Acknowledgements

The authors would like to thank Prof. Daniel Barbará from George Mason University for providing useful insights for the development of this work. Finally, the authors would like to thank CGU for providing the resources necessary to work in this research, as well as for allowing its publication.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *20th International Conference on Very Large Data Bases*, pages 478–499. Morgan Kaufmann, Los Altos, CA, 1994.

[2] R. R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, University of Utrecht, Utrecht, Netherlands, 1995.

[3] Remco R. Bouckaert. Bayesian network classiers in weka for version 3-5-7. Technical report, May 2008.

[4] S. Ceccon, D.F. Garway-Heath, D.P. Crabb, and A. Tucker. Exploring early glaucoma and the visual field test: Classification and clustering using bayesian networks. *IEEE Journal of Biomedical and Health Informatics*, 18(3):1008–1014, May 2014.

[5] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. CRISP-DM 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.

[6] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321357, 2002.

[7] Jie Cheng and Russell Greiner. Comparing bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, page 101108, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] Gregory F. Cooper and Edward Herskovits. A bayesian method for constructing bayesian belief networks from databases. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'91, page 8694, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[9] GregoryF. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[10] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, November 1997.

[11] Nir Friedman and Moises Goldszmidt. Building classifiers using bayesian networks. In *Proceedings of the national conference on artificial intelligence*, page 12771284, 1996.

[12] Moises Goldszmidt, James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith. Bayesian network classifiers. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[13] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):1018, November 2009.

[14] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86. AAAI Press, 1998.

[15] Hely Meirelles. *Licitacao e Contrato Administrativo*. Editora Revista dos Tribunais, Sao Paulo, Brazil, 7a. ed., atualizada pelo decreto-lei 2,300/86. edition, 1987.

[16] Andre Mueller. A critical study of the brazilian procurement law. Technical report, IBI - The Institute of Brazilian Business & Public Management Issues, Washington, 1998.

[17] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, page 98105, 1998.

[18] Wei Shi, Yao Wu Pei, Liang Sun, Jian Guo Wang, and Shao Qing Ren. The defect identification of LED chips based on bayesian classifier. *Applied Mechanics and Materials*, 333-335:1564–1568, July 2013.

[19] Rdiger Wirth. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, page 2939, 2000.

[20] Ye Ye, Fuchiang (Rich) Tsui, Michael Wagner, Jeremy U. Espino, and Qi Li. Influenza detection from emergency department reports using natural language processing and bayesian network classifiers. *Journal of the American Medical Informatics Association*, pages amiajnl–2013–001934, January 2014.

Hierarchical Bayesian Survival Analysis and Projective Covariate Selection in Cardiovascular Event Risk Prediction

Tomi Peltola
tomi.peltola@aalto.fi
Aalto University,
Finland

Aki S. Havulinna
aki.havulinna@thl.fi
National Institute for
Health and Welfare,
Finland

Veikko Salomaa
veikko.salomaa@thl.fi
National Institute for
Health and Welfare,
Finland

Aki Vehtari
aki.vehtari@aalto.fi
Aalto University,
Finland

Abstract

Identifying biomarkers with predictive value for disease risk stratification is an important task in epidemiology. This paper describes an application of Bayesian linear survival regression to model cardiovascular event risk in diabetic individuals with measurements available on 55 candidate biomarkers. We extend the survival model to include data from a larger set of non-diabetic individuals in an effort to increase the predictive performance for the diabetic subpopulation. We compare the Gaussian, Laplace and horseshoe shrinkage priors, and find that the last has the best predictive performance and shrinks strong predictors less than the others. We implement the projection predictive covariate selection approach of Dupuis and Robert (2003) to further search for small sets of predictive biomarkers that could provide cost-efficient prediction without significant loss in performance. In passing, we present a derivation of the projective covariate selection in Bayesian decision theoretic framework.

1 INTRODUCTION

Improving disease risk prediction is a major task in epidemiological research. Non-communicable diseases, many of which develop and progress slowly, are a major cause of morbidity worldwide. Accurate risk prediction could be used to screen individuals for targeted intervention. Advances in measurement technologies allow researchers cost-efficient quantification of large numbers of potentially relevant biomarkers, for example, in blood samples. However, often only a few of such candidate biomarkers could be expected to give practically relevant gain in risk stratification or could be realistically used in routine health care setting. The statistical challenge is then to identify an informative subset of the biomarkers and estimate its predictive performance.

Here, we describe an application of linear, hierarchical Bayesian survival regression to model cardiovascular event risk in diabetic individuals. The available data consists of 7932 Finnish individuals in the FINRISK 1997 cohort [1], of whom 401 had diabetes at the beginning of the study. The covariates consist of a set of 55 candidate biomarkers measured from blood samples and 12 established risk factors (e.g., baseline age, sex, body-mass index, lipoprotein cholesterol measures, blood pressure and smoking). The length of the follow-up period was 15 years. We focus on three key elements in the model construction: 1) using shrinkage priors to model the assumption of possibly limited relevance of many biomarkers, 2) utilizing the large set of non-diabetic individuals in the modelling, and 3) the selection of a subset of the biomarkers with predictive value. While the statistical approach is not limited to this particular application, we use the setting to make the description of the methods concrete.

Shrinkage or sparsity-promoting priors for regression coefficients are used to shrink the effects of (apparently) irrelevant covariates to zero, while retaining the effects of relevant covariates. Their use has increased with the availability of datasets with large numbers of features, for example, from high-throughput measurement technologies, which often capture a snapshot of a whole system (e.g., metabolome, genome) instead of targeted features. The interest has spawned considerable research effort into such priors and multiple alternatives have been proposed (see, e.g., refs [2–6]). In this work, we chose to compare three priors: the Laplace [3], the horseshoe [5] and, as a baseline, a Gaussian prior. The Laplace prior corresponds to the popular lasso penalty [7] in non-Bayesian regularized regression. The horseshoe prior has been shown to have desirable features in Bayesian analysis [5, 8]. We briefly review these priors in Section 2.2.

Of the 401 diabetic individuals in the study, 155 experienced a cardiovascular event within the follow-up period. This leaves a limited set of informative samples to perform the model fitting, covariate selection and predictive performance evaluation with. Although the risk of cardiovascular events is larger in diabetic individuals than the general population [9], we would expect that the risk factors are shared at least to some extent. Based on this assumption, we incorporate the non-diabetic individuals ($n = 7531$, 1031 events) into the analysis by constructing a hierarchical joint model, where the submodels for diabetic and non-diabetic individuals can be correlated (akin to transfer or multi-task learning [10]). The joint model does not place hard constraints on the similarity of the submodels, but allows the models to differ between non-diabetic and diabetic individuals and also between men and women. Details are given in Section 2.3.

While lasso regression in the non-Bayesian context can perform hard covariate selection by estimating exact zeroes for regression coefficients, the Bayesian shrinkage priors do not lead to sparse posterior distributions as there will remain uncertainty after observing a finite dataset. However, we are interested in finding a minimal subset of predictively relevant biomarkers as discussed above. To this end, we examine the use of projection predictive covariate selection[1], where the full model, encompassing all the candidate biomarkers and the uncertainties related to their effects, is taken as a *yardstick* for the smaller models. Specifically, the models with subsets of covariates are found by maximizing the similarity of their predictions to this reference as proposed by Dupuis and Robert [12]. Notably, this approach does not require specifying priors for the submodels and one can instead focus on building a good reference model. Dupuis and Robert [12] suggest choosing the size of the covariate subset based on an acceptable loss of explanatory power compared to the reference model. We examine using cross-validation based estimates of predictive performance as an alternative.

The structure of this article is as follows. In Section 2, we describe the survival model, shrinkage priors, and the hierarchical extension to include data of non-diabetic individuals. The projection predictive covariate selection is described in Section 3. The results from the application of the methods for cardiovascular-event-free survival modelling in diabetic individuals are presented in Section 4. Finally, Section 5 discusses the modelling approach.

[1]A comprehensive review of predictive Bayesian model selection approaches is given by Vehtari and Ojanen [11]. Our terminology follows theirs.

2 MODEL

We first consider modelling the cardiovascular-event-free survival in the subset of diabetic individuals only. The model is then extended to include the data of non-diabetic individuals, while allowing the covariate effects and the baseline hazard to differ in these groups and between men and women.

2.1 OBSERVATION MODEL

Let the observation t_i be the event time T_i or the censoring time C_i since the beginning of the study for ith individual and v_i be the corresponding event/censoring indicator (1 for observed events, 0 for censored). All censored cases are right censored (i.e., $T_i > C_i$ where only C_i is observed; censoring occurs in the data mostly because of event-free survival to the end of the follow-up). Further, let $\boldsymbol{x}_i$ be a column vector of the observed covariate values for the ith subject. We assume a parametric survival model, where the observations follow the Weibull model[2]

$$p(t_i|\boldsymbol{x}_i, v_i, \boldsymbol{\beta}, \alpha) = \alpha^{v_i} t_i^{v_i(\alpha-1)} \exp(v_i \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i - t_i^{\alpha} \exp(\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i))$$

with the shape α and the scale defined through the linear combination $\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i$ of the covariates [14]. The Weibull model is a proportional hazard model with the hazard function $h(T_i) = \alpha T_i^{\alpha-1} \exp(\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i)$.

We include a constant term 1 in the covariates $\boldsymbol{x}_i$ and denote the corresponding regression coefficient β_0. The intercept and the shape are given the diffuse priors:

$$\begin{aligned} \beta_0 &\sim \mathrm{N}(0, 10^2), \\ \log \alpha &\sim \mathrm{N}(0, 10^2). \end{aligned}$$

The covariates are divided into a set of established risk (or protective) factors and a set of new candidate biomarkers, which are of more uncertain relevance. The coefficients of the established predictors, β_j for $j = 1, \ldots, m_{bg}$, are given the prior [15]:

$$\begin{aligned} \beta_j &\sim \mathrm{N}(0, \sigma_s^2 \sigma_j^2), \text{ for } j = 1, \ldots, m_{bg}, \\ \sigma_j^2 &\sim \text{Inv--}\chi^2(1), \text{ for } j = 1, \ldots, m_{bg}, \\ \sigma_s &\sim \text{Half--N}(0, 10^2). \end{aligned}$$

Priors for the coefficients of the candidate biomarkers are considered below.

[2]The notation for probability distributions follows the parametrizations given in ref. [13], except for the Weibull model, which is explicitly written out. *Half*-distributions refer to the restriction to the real positive axis.

2.2 PRIORS FOR BIOMARKER COEFFICIENTS

Based on our prior assumption that only some of the biomarkers are expected to be practically relevant for prediction, we consider the use of shrinkage priors for the biomarker coefficients. As discussed in the introduction, there has been a lot of recent research into these type of priors and there are multiple proposals. We restrict our consideration to three alternatives: the horseshoe prior [5], the Laplace prior [3], and, as a baseline approach, a Gaussian prior. Each of these can be expressed as normal scale mixtures

$$\beta_j \sim \mathrm{N}(0, \tau_s^2 \tau_j^2), \text{ for } j = m_{bg}+1, \ldots, m_{bg}+m_{bm},$$

where τ_s is a global scale parameter (shared across j) and τ_j are local parameters. Ideally, the prior shrinks the coefficients of irrelevant biomarkers to zero, but allows large coefficients for relevant biomarkers. In a sparse situation, with many irrelevant biomarkers and few relevant, this could be effected by making τ_s small, but allowing some τ_j to take on large values to escape the shrinkage [16].

The priors for τ_js, for $j = m_{bg}+1, \ldots, m_{bg}+m_{bm}$, for the three alternatives are

$$\begin{aligned} \tau_j &\sim \text{Half–Cauchy}(0,1) && \text{for horseshoe,} \\ \tau_j^2 &\sim \text{Exponential}(0.7) && \text{for Laplace,} \\ \tau_j &= 1 && \text{for the Gaussian.} \end{aligned}$$

A comparison of the Laplace and horseshoe prior is given in ref. [5]: it is noted that the Laplace prior may overshrink large coefficients in a sparse situation, while the horseshoe prior is more robust (see also ref. [16]). Furthermore, van der Pas et al. [8] derive theoretical results indicating that the posterior distribution under the horseshoe prior may be more informative[3] than under the Laplace prior in a sparse normal means problem. The Gaussian prior does not try to separate between relevant and irrelevant covariates as it depends only on the shared scale parameter τ_s.

The same prior is given for the global scale parameter in each case:

$$\tau_s \sim \text{Half–Cauchy}(0,1),$$

which has its (bounded) mode at zero, but is only weakly informative as it also places a substantial amount of prior mass far from zero (see refs [15–17] for discussion on priors for global variance parameters).

[3]That is, the posterior mean estimator attains a minimax risk, possibly up to a multiplicative constant, in a sparse setting and the posterior contracts at a similar rate (with conditions on τ_s).

2.3 HIERARCHICAL EXTENSION

Next, we consider extending the approach to jointly model the event-free survival of non-diabetic men (NM), non-diabetic women (NW), diabetic men (DM), and diabetic women (DW). Our aim is to increase the predictive performance of the model specifically in the subset of diabetic individuals, but gain power by including the larger set of observations for non-diabetic individuals in the model. To this end, we tie together the submodels of the four groups using the following assumptions:

1. The relevance of a biomarker will be similar for all the submodels.
2. The effect size of a biomarker (or other covariate) and its direction are similar between men and women, and between diabetic and non-diabetic individuals.
3. The baseline hazard functions have similar shapes for men and women, and diabetic and non-diabetic individuals.

Let $\boldsymbol{\beta}_j = [\beta_{j,NM}\ \beta_{j,NW}\ \beta_{j,DM}\ \beta_{j,DW}]^{\mathrm{T}}$ be the coefficients for the jth biomarker in the four submodels. We set

$$\boldsymbol{\beta}_j \sim \mathrm{N}(\mathbf{0}, r_j^2 \lambda \boldsymbol{\Lambda}^{-1}),$$

where $r_j^2 \lambda \boldsymbol{\Lambda}^{-1}$ is the prior covariance matrix. Here, $r_j = \tau_j \tau_s$ and follows one of the prior specifications given in the previous section. This encodes the first assumption above: a single r_j parameter defines the relevance of the jth biomarker in all the four submodels.

To encode the second assumption, we specify the structure of the prior precision matrix as

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1+c_N+s_M & -c_N & -s_M & 0 \\ -c_N & 1+c_N+s_W & 0 & -s_W \\ -s_M & 0 & 1+c_D+s_M & -c_D \\ 0 & -s_W & -c_D & 1+c_D+s_W \end{bmatrix}.$$

The corresponding graphical structure is illustrated in Figure 1. As will be made more explicit below, the c_N and c_D control the similarity of the submodels of non-diabetic men and women, and between the submodels of diabetic men and women, respectively. s_M and s_W control the similarity between the submodels of non-diabetic and diabetic men, and non-diabetic and diabetic women, respectively. We further simplify the model by taking $c_N = c_D = c$ and $s_M = s_W = s$ and constrain $c > 0$ and $s > 0$. The precision matrix has similarity to the one used by Liu et al. [18] to learn dependencies between covariates, but here $\boldsymbol{\Lambda}$ is restricted to encode a specific prior structure.

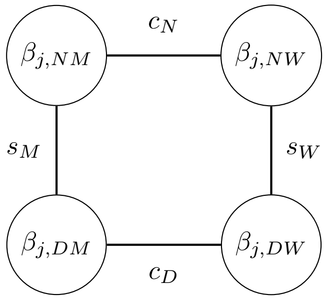


Figure 1: Prior structure for the regression coefficients of jth biomarker in the joint model.

We choose $\lambda = \frac{(2c+1)(2s+1)(2c+2s+1)}{(1+2c+2s+2cs)(c+s+1)}$ as this makes the diagonal elements of $\lambda\mathbf{\Lambda}^{-1}$ equal to 1, that is, $\lambda\mathbf{\Lambda}^{-1}$ becomes a correlation matrix. The relevance of the jth biomarker is then solely dependent on r_j.

For more insight, the prior for $\boldsymbol{\beta}_j$ may be written out as proportional to

$$\exp\left(-\frac{1}{2r_j^2\lambda}(S_2 + cS_c + sS_s)\right),$$

where $S_2 = \beta_{j,NM}^2 + \beta_{j,NW}^2 + \beta_{j,DM}^2 + \beta_{j,DW}^2$, $S_c = (\beta_{j,NM} - \beta_{j,NW})^2 + (\beta_{j,DM} - \beta_{j,DW})^2$ and $S_s = (\beta_{j,NM} - \beta_{j,DM})^2 + (\beta_{j,NW} - \beta_{j,DW})^2$. c controls the penalization in the difference between men and women, and s controls the penalization in the difference between non-diabetic and diabetic subjects. Taking negative logarithm of the prior shows that it corresponds to a specific Bayesian version of the multi-task graph regularization penalty proposed by Evgeniou et al. [19] and further studied by Sheldon [20]. The prior can also be represented in the sparse Bayesian multi-task learning framework of Archambeau et al. [21], where a zero-mean matrix-variate Gaussian density is placed on $\boldsymbol{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m]$ with row covariance $\mathbf{\Omega}$ (over the m covariates) and column covariance $\mathbf{\Sigma}$ (over the *tasks*). Here, $\mathbf{\Omega}$ is a diagonal matrix with elements r_j^2 and $\mathbf{\Sigma} = \lambda\mathbf{\Lambda}^{-1}$.

We use the following transformations of c and s: $c = (1-c')^{-1} - 1$ and $s = (1-s')^{-1} - 1$, where $c' \in [0,1)$ and $s' \in [0,1)$. At $c' = 0$, $c = 0$ and the corresponding submodels are independent. As $c' \to 1$, $c \to \infty$ and the corresponding submodels are constrained to identical. s' behaves similarly.

We can also examine the implied prior distribution of the difference between two $\beta_{X,j}$ coefficients as a function of c' and s'. First, note that the distribution of $\beta_{X,j} - \beta_{Y,j}$ is $\mathrm{N}(0, 2r_j^2(1-\rho))$, where ρ is the correlation coefficient. Specifically, the variance of the distribution is linearly dependent on ρ and, for $\rho \geq 0$, has the maximum value of $2r_j^2$ when $\rho = 0$ and the

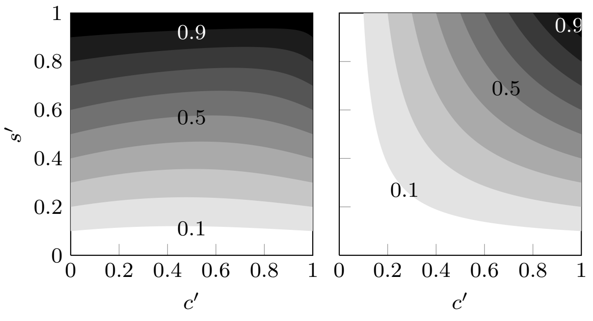


Figure 2: Contour plots of the correlation coefficient between $\beta_{j,NM}$ and $\beta_{j,DM}$ (left) and $\beta_{j,NM}$ and $\beta_{j,DW}$ (right) as a function of c' and s'.

minimum value of 0 when $\rho = 1$. In Figure 2, the implied prior correlation coefficients of some interesting pairs of $\beta_{X,j}$s are shown as functions of c' and s': s' controls almost linearly the correlation between $\beta_{j,NM}$ and $\beta_{j,DM}$, whereas the correlation between $\beta_{j,NM}$ and $\beta_{j,DW}$ is close to bilinear in c' and s'.

To complete the prior specification c' and s' are given prior distributions. We use different parameters for biomarkers (c' and s'), other covariates (c'_{bg} and s'_{bg}) and the log-scale Weibull shape parameter $\log\alpha$ (c'_α and s'_α; this encodes the third assumption):

$$\begin{aligned} c' &\sim \mathrm{Beta}(a_c, b_c), \\ s' &\sim \mathrm{Beta}(a_s, b_s), \\ c'_{bg} &\sim \mathrm{Beta}(a_c, b_c), \\ s'_{bg} &\sim \mathrm{Beta}(a_s, b_s), \\ c'_\alpha &\sim \mathrm{Beta}(a_c, b_c), \\ s'_\alpha &\sim \mathrm{Beta}(a_s, b_s). \end{aligned}$$

Finally, a_c, b_c, a_s and b_s are given $\mathrm{Gamma}(\frac{1}{2}, \frac{1}{4})$ priors.

We note that the eigendecomposition of $\mathbf{\Lambda} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}$ is of simple form, with $\boldsymbol{D}$ being a diagonal matrix with elements 1, $1+2c$, $1+2s$, $1+2c+2s$ and

$$\boldsymbol{V} = \frac{1}{2}\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

This can be useful in reparametrizing the model for Markov chain Monte Carlo sampling algorithms. It also shows that the precision matrix is positive definite.

3 METHODS FOR BIOMARKER SELECTION AND PREDICTIVE PERFORMANCE EVALUATION

The approaches used for biomarker selection and evaluation of predictive performance are described below. The model constructed in previous section is used as the reference model in the biomarker selection.

3.1 PROJECTION PREDICTIVE COVARIATE SELECTION

Assuming the availability of a reference model, which is a good representation of the predictive power of the candidate biomarkers and the related uncertainty, we seek a subset of the biomarkers, which can be used for prediction without a large loss in performance relative to the reference model. Our prior assumption of sparsity in the biomarker effects implies that this goal could be achievable. We describe the approach in two steps: 1) defining a submodel for making predictions with a specific subset of the candidate biomarkers, and 2) finding submodels with good predictive performance.

3.1.1 Projective Submodels

We use the projective approach of Dupuis and Robert [12], Goutis and Robert [22] to find the parameters of the submodel, but present an alternative derivation in the Bayesian decision theoretic framework reviewed in ref. [11]. The *projection* is posed as a solution to an optimization problem with regard to a restriction of the reference model. Let the covariates $\boldsymbol{x}$ be divided into two parts $\boldsymbol{x} = [\boldsymbol{x}_\perp, \boldsymbol{x}_\top]$ and define a submodel $M_\perp$ to be restricted to using the covariates in $\boldsymbol{x}_\perp$[4] with parameters $\boldsymbol{\theta}_\perp = (\boldsymbol{\beta}_\perp, \alpha_\perp)$ in the Weibull model. We find the submodel by maximizing the Gibbs reference utility

$$\bar{u}(M_\perp) = \int \left[\int u(M_\perp, \boldsymbol{x}_\perp, \boldsymbol{\theta}, T) p(T|\boldsymbol{\theta}, \boldsymbol{x}) dT \right] p(\boldsymbol{\theta}|D) p(\boldsymbol{x}) d(\boldsymbol{\theta}, \boldsymbol{x})$$

with respect to the unknown probability densities $f(\boldsymbol{\theta}_\perp|\boldsymbol{\theta})$ appearing in the $u(M_\perp, \boldsymbol{x}_\perp, \boldsymbol{\theta}, T) = \int f(\boldsymbol{\theta}_\perp|\boldsymbol{\theta}) \log p(T|\boldsymbol{\theta}_\perp, \boldsymbol{x}_\perp) d\boldsymbol{\theta}_\perp$. Here, $p(\boldsymbol{\theta}|D)$ is the posterior distribution of the reference model given the observed data D and $p(\boldsymbol{x})$ is the distribution of the covariates. Writing out u and changing the integration order,

$$\bar{u}(M_\perp) = \int \left[\int p(T|\boldsymbol{\theta}, \boldsymbol{x}) \log p(T|\boldsymbol{\theta}_\perp, \boldsymbol{x}_\perp) dT \right] \times f(\boldsymbol{\theta}_\perp|\boldsymbol{\theta}) p(\boldsymbol{\theta}|D) p(\boldsymbol{x}) d(\boldsymbol{\theta}_\perp, \boldsymbol{\theta}, \boldsymbol{x}).$$

Finally, to arrive at the same solution with Dupuis and Robert [12], $f(\boldsymbol{\theta}_\perp|\boldsymbol{\theta})$ can be restricted to the Dirac delta function $\delta(\boldsymbol{\theta}_\perp - \hat{\boldsymbol{\theta}}_\perp)$ with an offset $\hat{\boldsymbol{\theta}}_\perp$ that depends on $\boldsymbol{\theta}$. That is, the solution to the maximization of $\bar{u}$ is defined pointwise for each $\boldsymbol{\theta}$ as the corresponding optimal value of $\hat{\boldsymbol{\theta}}_\perp$. The pointwise solution arises from the dependence of f on $\boldsymbol{\theta}$.

As $p(\boldsymbol{\theta}|D)$ is not available analytically and $p(\boldsymbol{x})$ at all, the former is approximated with Markov chain Monte Carlo methods and the latter by using $\boldsymbol{x}_i$ samples available in the data D [12]. The obtained estimate is

$$\bar{u}(M_\perp) \approx \frac{1}{nJ} \sum_{i,j} \left[\int p(T|\boldsymbol{\theta}^{(j)}, \boldsymbol{x}_i) \log p(T|\hat{\boldsymbol{\theta}}_\perp^{(j)}, \boldsymbol{x}_{i,\perp}) dT \right],$$

where the double sum runs over the n data points and the J posterior samples. The optimization problems to find the optimal $\hat{\boldsymbol{\theta}}_\perp^{(j)}$s are independent over j. We solve them using the Newton's method.

We define the projection predictive distribution for the submodel $M_\perp$ as

$$p(T|\boldsymbol{x}_\perp, M_{ref}) = \int p(T|\boldsymbol{x}_\perp, \boldsymbol{\theta}_\perp) f(\boldsymbol{\theta}_\perp|M_{ref}) d\boldsymbol{\theta}_\perp,$$

where we explicitly emphasize the dependence on the reference model M_{ref} and which is approximated using the projected samples $\hat{\boldsymbol{\theta}}_\perp^{(j)}$s. This kind of projected predictive distribution was also considered by Nott and Leng [23].

Note that scaling the estimated $\bar{u}$ as $d(M_\perp) = \bar{u}(M_{ref}) - \bar{u}(M_\perp)$ (and minimizing instead of maximizing) does not change the optimal solution and gives otherwise the same formula as $\bar{u}$, except the term in square brackets is replaced with the Kullback–Leibler divergence between $p(T|\boldsymbol{x}, \boldsymbol{\theta})$ and $p(T|\boldsymbol{x}_\perp, \boldsymbol{\theta}_\perp)$. This gives the approach further information theoretic justification and is the basis of the formulation in Dupuis and Robert [12]. They also suggest defining the relative explanatory power of the submodel as

$$\text{relative explanatory power}(M_\perp) = 1 - \frac{d(M_\perp)}{d(M_0)},$$

where M_0 refers to the model without any of the candidate biomarkers and which transforms the $d(M_\perp)$ values to between 0 (for $M_\perp = M_0$) and 1 (for $M_\perp = M_{ref}$).

[4]We assume that the established risk factors are always included in this set.

3.1.2 Submodel Search

$\bar{u}$ (or equivalently d) is used to compare the submodels in the search for good subsets of biomarkers. However, exhaustive search of the model space[5] is not feasible, unless the number of candidate biomarkers is small. We choose to use the suboptimal forward selection strategy for its simplicity and its scalability to large covariate sets:

1. Begin with the submodel M_0 (no biomarkers) and set j to 0.
2. Repeat until all biomarkers have been added:
   (a) Find the projections for all submodels that are obtainable by adding one new biomarker to M_j. Select the one with largest $\bar{u}$ and set it as M_{j+1}. Set j to $j+1$.

This defines a deterministic[6] path of models from M_0 to $M_{m_{bm}}$ and gives a ranking of the biomarkers according to their projection predictive value. Dupuis and Robert [12] suggest finally choosing the smallest submodel with an acceptable loss in the explanatory power relative to the reference model (and use a slightly more elaborate search). Alternatively, one could monitor some other statistic (e.g., predictive performance) along the search path to locate good submodels. Computing the full forward selection path may not be necessary, if a suitable stopping criterion is used in the step 2 above.

3.2 PREDICTIVE PERFORMANCE EVALUATION

Given a model M with posterior predictive distribution $p(T_*|\boldsymbol{x}_*, D)$, where D is the observed data, we evaluate its predictive performance using the logarithm of the predictive density (LPD) at an actual observation $(t_*, v_*, \boldsymbol{x}_*)$. This scoring rule is proper and measures the calibration and sharpness of the predictive distribution simultaneously [24]. As the predictive densities are not available analytically for the models considered here, we estimate the LPD score from the Markov chain Monte Carlo samples of the posterior distribution:

$$\mathrm{LPD}_*(M) \approx \log \frac{1}{J} \sum_{j}^{J} p(t_*|\boldsymbol{x}_*, v_*, \boldsymbol{\beta}^{(j)}, \alpha^{(j)}),$$

where $(\boldsymbol{\beta}^{(j)}, \alpha^{(j)})$ are J posterior samples of the model given the data D.

[5]The number of subsets for m_{bm} covariates is $2^{m_{bm}}$.

[6]Given the stochastic samples from the posterior distribution of the reference model.

Stratified ten-fold cross-validation [25] is used to obtain estimates of the generalization performance: The full dataset is divided randomly into ten disjoint subsets (folds), while balancing the sets to have approximately similar age distributions and proportions of diabetic and non-diabetic individuals, men and women, and cases of cardiovascular events. Predictions for each fold are obtained using a posterior distribution based on training data, where the particular fold has been left out. Given predictions obtained this way, the predictive performance is summarized by the mean LPD over the full set of n data points (MLPD).

To reduce variance and gauge uncertainty in model comparisons, we compute Bayesian bootstrap [26] samples of the MLDP difference (ΔMLPD) between model M_a and model M_b by

$$\Delta\mathrm{MLPD}^{(j)}(M_a, M_b) = \sum_{i=1}^{n} w_i^{(j)} [\mathrm{LPD}_i(M_a) - \mathrm{LPD}_i(M_b)],$$

where $w_i^{(j)}$, $i = 1, \ldots, n$, are the bootstrap weights ($\sum_i w_i^{(j)} = 1$) for the jth bootstrap sample generated using the Dirichlet distribution with parameters set to 1 [11]. The comparison is summarized by the q-value[7]:

$$q(M_a, M_b) = \frac{1}{J} \sum_{j=1}^{J} I(\Delta\mathrm{MLPD}^{(j)} \geq 0),$$

where $I(\cdot) = 1$ if the given condition holds and 0 otherwise, and which is interpreted as the Bayesian posterior probability (under the Dirichlet model) of M_a performing better than M_b [11].

4 RESULTS

Missing values in the covariate data were multiply imputed using chained linear regressions with in-house scripts based on ref. [27]. The candidate biomarkers were log-transformed and scaled to have zero mean and unit variance. The No-U-Turn variant of the Hamiltonian Monte Carlo algorithm [28], as implemented in Stan software [29], was used to sample from the posterior distributions of the full models. The sampling was done independently for 5 imputed datasets (4 chains of 1000 samples after burn-in for each). The samples were then concatenated. The sampling process was further performed independently for each of the 10 cross-validation training sets. All shown estimates of predictive performance were computed using cross-validation (Section 3.2).

[7]We use q instead of p to avoid confusion with the frequentist p-value.

Table 1: Model comparisons on cross-validation predictions. MLPDs and q-values (Section 3.2) are shown for predictions only on diabetic women, only on diabetic men or both. q-values are calculated against the *joint horseshoe* model; color scale 0.0 🟦🟦🟦🟦⬜⬜🟧🟧🟥🟥 1.0.

	women		men		women & men	
model	MLPD	q-value	MLPD	q-value	MLPD	q-value
joint horseshoe	-0.581	NA	-0.716	NA	-0.652	NA
joint Laplace	-0.582	0.27 🟦	-0.720	0.10 🟦	-0.656	0.08 🟦
joint Gaussian	-0.585	0.22 🟦	-0.727	0.05 🟦	-0.660	0.04 🟦
joint no-biomarkers	-0.594	0.18 🟦	-0.758	0.03 🟦	-0.681	0.01 🟦
diab women&men horseshoe	-0.606	0.03 🟦	-0.719	0.44 ⬜	-0.666	0.13 🟦
diab women/men horseshoe	-0.610	0.03 🟦	-0.721	0.45 ⬜	-0.669	0.15 🟦
diab women/men no-biomarkers	-0.613	0.05 🟦	-0.765	0.04 🟦	-0.694	0.01 🟦

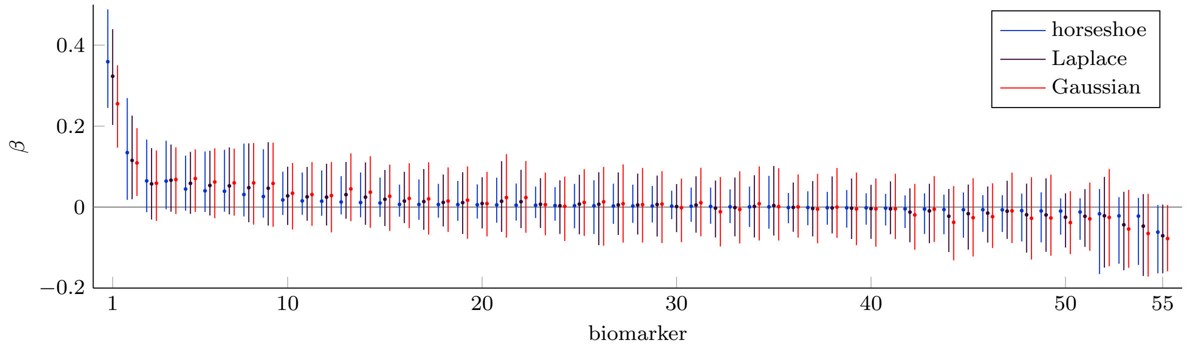


Figure 3: Biomarker regression coefficients β for the submodel of diabetic men in the joint models with the horseshoe, Laplace and Gaussian priors (full dataset). Dot is the mean and vertical line shows the 95% credible interval. Biomarkers are ordered according to the mean coefficients of the horseshoe model.

4.1 MODEL COMPARISONS

Table 1 presents results on comparing the mean log predictive densities (MLPD) of the following combinations of models: *joint* for the joint model of non-diabetic and diabetic individuals (Section 2.3), *diab women&men* for a joint model of diabetic men and women (two-group version of Section 2.3), *diab women/men* for separate models of diabetic men and women (without the extension of Section 2.3), and using the *horseshoe*, *Laplace* or *Gaussian* priors on the biomarker effects, or using only the established risk factors (*no-biomarkers*). The MLPDs and q-values were computed separately for the predictions for women and men, and for pooled predictions, and, importantly, in each case only for the predictions on the diabetic subpopulation.

The results show that there is an increase in the predictive performance when supplanting the established risk factors with the candidate biomarkers. The increase holds both when using the joint models or using only the data of diabetic individuals and seems to be greater in men. This indicates that the candidate biomarkers contain relevant information for predicting cardiovascular event risk.

Including the data of the non-diabetic individuals in the model seems to increase the predictive performance for the diabetic subpopulation, especially for women. The covariate effects in the joint models are very similar across the diabetic and non-diabetic submodels: posterior mean of s' is 0.96 for the horseshoe model. This implies that the risk factors behave similarly in both groups, but it is also possible that the dataset has limited information to distinguish between them and that larger datasets could uncover more differences.

Finally, it seems that the horseshoe prior performs better than the Laplace, and that the Gaussian is the worst of the three for this data. Figure 3 shows a comparison of the biomarker regression coefficients under these priors. The Laplace and the Gaussian priors shrink the largest coefficient more than the horseshoe as would be expected in a sparse setting [5, 16]. Furthermore, the horseshoe seems to shrink coefficients near zero more strongly than the Laplace making the credible intervals around zero narrower.

4.2 BIOMARKER SELECTION AND SUBMODEL PREDICTIVE PERFORMANCE

We applied the projection predictive covariate selection (Section 3.1) with the joint horseshoe model as the reference. The forward selection was run using only the part of the model concerning diabetic individuals. We run the forward selection jointly for women and men to get an overall biomarker ranking for the diabetic subpopulation. The forward selection was run also for each cross-validation training set separately (using the reference model fitted on the corresponding training data).

Figure 4 shows the relative explanatory power curves along the forward selection path. In the full dataset, the best candidate biomarker attains 61% explanatory power relative to the reference model, five best reach over 80% and ten biomarkers are needed to reach over 90%. The growth in the explanatory power slows with more biomarkers, indicating diminishing gains from adding more candidate biomarkers (22 are needed to reach 95% and the remaining 33 account for the last 5%).

However, choosing an acceptable loss in the explanatory power to select an appropriate minimal subset of the biomarkers for use in prediction tasks seems difficult. In Figure 5, we show MLPDs (normalized to the reference model) obtained using the projection predictive covariate selection approach within the cross-validation. Top panel shows the ΔMLPD along the forward selection path and the bottom panel by the obtained relative explanatory power (e.g., at 0.6, the predictions in each cross-validation fold was made with the smallest submodel reaching 60% power in that fold). These show a mode at 2 biomarkers and at around 0.65 relative explanatory power (which corresponds to choosing two, three or four biomarkers depending on the fold). A second peak can be seen at 10 biomarkers or correspondingly at 0.91 power (10–16 biomarkers).

Unfortunately, the variance in the cross-validation estimates is quite large for making a definite choice based on them. Figure 6 shows the full set of pairwise comparisons between the submodels along the forward selection path (by number of biomarkers; same as in Figure 5 top panel). This indicates that two biomarkers is overall the best choice, but the difference to the 10

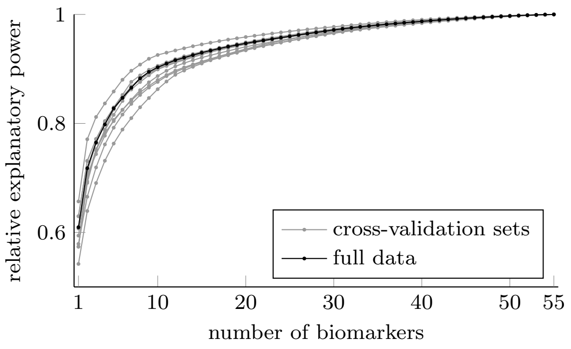
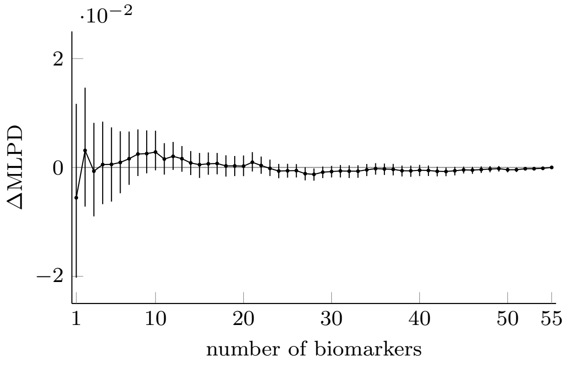


Figure 4: Relative explanatory powers along the forward selection path.



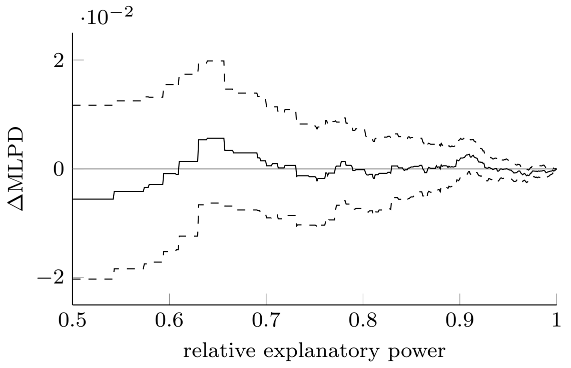


Figure 5: ΔMLPD values (in reference to the full model) by number of biomarkers (top) or by explanatory power threshold (bottom). Top: vertical lines are 95% Bayesian bootstrap credible intervals for the ΔMLPD. Bottom: dashed curves show the (pointwise) credible interval.

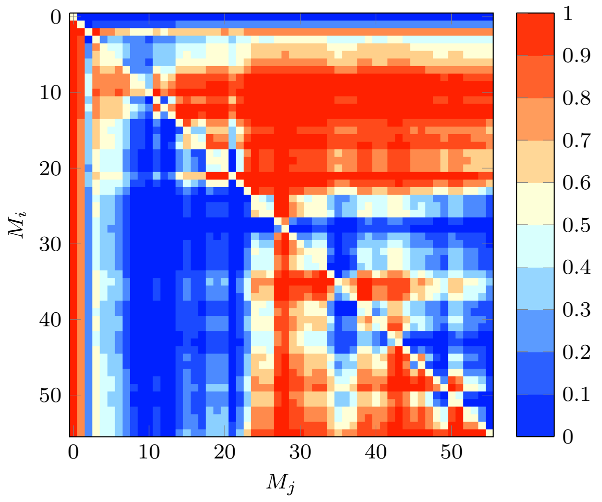


Figure 6: q-value matrix for M_i is better than M_j with regard to MLPD (Section 3.2; M.s refer to the submodels along the forward selection path).

biomarker selection is not large (q-value $= 0.52$). However, on comparing these to the full model or generally models with 11 or more biomarkers, the 10 biomarker selection is more confidently better (q-values mostly > 0.9) than the 2 biomarker selection (q-values mostly within 0.7–0.8).

Nevertheless, the analysis seems to support two clearly predictively relevant biomarkers for the cardiovascular risk prediction, with further 8 possibly interesting candidate biomarkers, but with some uncertainty about their relevance. Figure 3 also supports this conclusion with two of the biomarkers having clearly non-zero effects.

5 DISCUSSION

This paper presented a Bayesian analysis of cardiovascular-event-free survival in diabetic individuals, with the aim of identifying biomarkers with predictive value. We presented a comparison of the horseshoe, Laplace and Gaussian priors on the candidate biomarker effects and demonstrated empirically an expected [5, 16] difference in their behaviour. We further extended the model hierarchically to include data of non-diabetic individuals and examined the use of projection predictive covariate selection to find biomarker subsets with good predictive performance.

We could also hope that the predictive biomarkers capture some part of the state of the underlying disease process and as such could be used to speculate about causal disease pathways and to prioritize biomarkers for further study. However, the analysis approach does not warrant any formal causal inferences. Moreover, the inclusion of the data of non-diabetic individuals may bias the inferences on the diabetic subpopulation towards the general population, when the dataset has limited information to distinguish them. Nevertheless, the presented predictive comparisons, being independent of the model assumptions, justify studying the joint model.

The submodels in projection predictive covariate selection depend on the observed data only through the reference model. Thus, finding the submodel parameters and the covariate selection itself do not cause further fitting to the data, but rely on the information provided by the reference model [11]. The projected submodels may also be able to retain some predictive features of the reference model that would not be available, if the submodels were independently fitted to the data [11]: importantly, from Bayesian point of view, the submodel may be able to account for uncertainty due to the omission of some covariates.

However, selecting a single submodel for future prediction tasks may be difficult. We examined using the projection approach within cross-validation to obtain estimates of the submodel predictive performances. A disadvantage of this procedure is that the performance estimates are for the selection process and not for some particular combination of selected biomarkers. Furthermore, if selection is based on these estimates, the performance estimate for the chosen submodel will not anymore be unbiased for out-of-sample prediction unless nested cross-validation is used [11].

Acknowledgements

We acknowledge the computational resources provided by Aalto Science-IT project.

References

[1] Erkki Vartiainen, Tiina Laatikainen, Markku Peltonen, Anne Juolevi, Satu Männistö, Jouko Sundvall, Pekka Jousilahti, Veikko Salomaa, Liisa Valsta, and Pekka Puska. Thirty-five-year trends in cardiovascular risk factors in Finland. *International Journal of Epidemiology*, 39(2):504–518, 2010.

[2] T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404): 1023–1032, 1988.

[3] Trevor Park and George Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[4] Jim E. Griffin and Philip J. Brown. Inference

with normal-gamma prior distributions in regression problems. *Bayesian Analysis*, 5(1):171–188, 2010.

[5] Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480, 2010.

[6] Zhihua Zhang, Shusen Wang, Dehua Liu, and Michael I. Jordan. EP-GIG priors and applications in Bayesian sparse learning. *The Journal of Machine Learning Research*, 13(1):2031–2061, 2012.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1): 267–288, 1996.

[8] S. L. van der Pas, B. J. K. Kleijn, and A. W. van der Vaart. The horseshoe estimator: Posterior concentration around nearly black vectors. *arXiv preprint arXiv:1404.0202*, 2014.

[9] Emerging Risk Factors Collaboration. Diabetes mellitus, fasting blood glucose concentration, and risk of vascular disease: a collaborative meta-analysis of 102 prospective studies. *The Lancet*, 375(9733):2215–2222, 2010.

[10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[11] Aki Vehtari and Janne Ojanen. A survey of Bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 6:142–228, 2012.

[12] Jérome A. Dupuis and Christian P. Robert. Variable selection in qualitative models via an entropic explanatory power. *Journal of Statistical Planning and Inference*, 111(1):77–94, 2003.

[13] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. CRC press, third edition, 2014.

[14] Joseph G. Ibrahim, Ming-Hui Chen, and Debajyoti Sinha. *Bayesian Survival Analysis*. Springer, 2001.

[15] Andrew Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3):515–533, 2006.

[16] Nicholas G. Polson and James G. Scott. Shrink globally, act locally: Sparse Bayesian regularization and prediction. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 9*, pages 501–538. Oxford University Press, 2011.

[17] Nicholas G. Polson and James G. Scott. On the half-Cauchy prior for a global scale parameter. *Bayesian Analysis*, 7(4):887–902, 2012.

[18] Fei Liu, Sounak Chakraborty, Fan Li, Yan Liu, and Aurelie C. Lozano. Bayesian regularization via graph Laplacian. *Bayesian Analysis*, 9(2): 449–474, 2014.

[19] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[20] Daniel Sheldon. Graphical multi-task learning. In *NIPS 2008 Workshop: "Structured Input – Structured Output"*, 2008.

[21] Cédric Archambeau, Shengbo Guo, and Onno Zoeter. Sparse Bayesian multi-task learning. In *Advances in Neural Information Processing Systems 24*, pages 1755–1763, 2011.

[22] Constantinos Goutis and Christian P. Robert. Model choice in generalised linear models: A Bayesian approach via Kullback–Leibler projections. *Biometrika*, 85(1):29–37, 1998.

[23] David J. Nott and Chenlei Leng. Bayesian projection approaches to variable selection in generalized linear models. *Computational Statistics & Data Analysis*, 54(12):3227–3241, 2010.

[24] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[25] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1145, 1995.

[26] Donald B. Rubin. The Bayesian bootstrap. *The Annals of Statistics*, 9(1):130–134, 1981.

[27] Stef van Buuren and Karin Groothuis-Oudshoorn. MICE: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3), 2011.

[28] Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

[29] Stan Development Team. Stan: A C++ library for probability and sampling, version 2.2, 2014. URL `http://mc-stan.org/`.

Dynamic Bayesian Network Modeling of Vascularization in Engineered Tissues

Caner Komurlu
Computer Science Department
Illinois Institute of Technology
Chicago, IL, 60616
ckomurlu@hawk.iit.edu

Jinjian Shao
Computer Science Department
Illinois Institute of Technology
Chicago, IL, 60616
jshao3@hawk.iit.edu

Mustafa Bilgic
Computer Science Department
Illinois Institute of Technology
Chicago, IL, 60616
mbilgic@iit.edu

Abstract

In this paper, we present a dynamic Bayesian network (DBN) approach to modeling vascularization in engineered tissues. Injuries and diseases can cause significant tissue loss to the degree where the body is unable to heal itself. Tissue engineering aims to replace the lost tissue through use of stem cells and biomaterials. For tissue cells to multiply and migrate, they need to be close to blood vessels, and hence proper vascularization of the tissue is an essential component of the engineering process. We model vascularization through a DBN whose structure and parameters are elicited from experts. The DBN provides spatial and temporal probabilistic reasoning, enabling tissue engineers to test sensitivity of vascularization to various factors and gain useful insights into the vascularization process. We present initial results in this paper and then discuss a number of future research problems and challenges.

1 INTRODUCTION

People lose tissue due to accidents, medical operations, treatments, and illnesses. While some organs, e.g. liver, can replace the lost tissue most cannot especially when the damage is too severe. For these kinds of tissue damages, the lost tissue can be replaced by engineering a new tissue through stem cells and biomaterials [18].

An essential process for engineering a healthy tissue is the proper vascularization (formation of new blood vessels) of the tissue, as the tissue cells need to be close to the blood vessels both to discharge their waste and to receive nutrition and oxygen. The blood vessels need to spread out in the tissue, invade into the depths of the tissue, and form connections to allow blood circulation.

The formation of new blood vessels are triggered and affected by growth factors that are released by distressed cells that are far from the existing blood vessels. When these growth factors reach existing blood vessels, they sprout new branches and these branches "search" for the distressed cells by following the gradient of the growth factor. This process, however, is stochastic for at least two reasons: i) even though growth factors are the main ingredients for causing sprouts, they are not the only elements that affect vascularization, and ii) the growth factors are increasingly more uniformly distributed as they go further away from the distressed cells, and hence the gradient is almost uniform, hindering the capability of the blood vessel finding its way correctly.

This inherent stochasticity in the vascularization process, the spatial nature of the tissue, and the temporal aspect of the vascularization make temporal graphical models a great fit for reasoning with uncertainty in vascularization. In this paper, we present a dynamic Bayesian network (DBN) for modeling vascularization in engineered tissues. We elicit the structure of the DBN from tissue engineering experts and we experiment with various parameter settings to provide further insights into the vascularization process. Because this is a first and novel application of DBNs to tissue engineering, it avails itself to many interesting future research directions and challenges.

Our contributions in this paper include:

- We present a novel application of DBNs to vascularization in engineered tissues
- We present initial results and insights, where we experiment with various parameter settings, and
- We discuss several future research challenges and opportunities in detail.

The rest of the paper is organized as follows: in Section 2, we provide a brief background on tissue engineering and vascularization. In Section 3, we describe our DBN model for vascularization. We present our experimental setup and results in Section 4. In Section 5, we briefly discuss related work. We then discuss future research directions and challenges in detail in Section 6, and then conclude.

2 BACKGROUND

In this section, we first provide a brief background on tissue engineering and vascularization and then discuss briefly why dynamic Bayesian networks (DBNs) are a good fit for modeling vascularization.

People lose tissue due to accidents, treatments, and illnesses. Some organs, e.g. liver, can replace the lost tissue while others cannot. Sometimes, the damage can be so severe that the body cannot heal itself. For example, bones can heal after smooth fractures. Yet, some fractures damage bone body so severely that the bone cannot regenerate. For these kinds of damages, the lost tissue can be replaced by engineering a new tissue through stem cells and biomaterials.

Stem cells are generic types of cells that have the ability to replicate and transform to any tissue. Stem cells, like all other cells, need to be close enough to the blood vessels so that they can forward their biological wastes to the vessels and they can be fed with nutrition and oxygen carried by the blood vessels. When a tissue is engineered through replication and transformation of stem and tissue cells, there is no existing blood vessel web in the environment; the only blood vessels available are the original vessels located at the edges, ready to sprout and progress to the depths of the newly-formed tissue.

The stem cells that do not have access to blood vessels will not be able to discharge waste and receive nutrition and oxygen. In such cases, a cell starts signaling about its needs by means of emitting chemicals called vascular endothelial growth factor (VEGF). VEGF diffuses and disperses in the environment. When it contacts a blood vessel, it triggers a new sprout of blood vessel towards the source of emission. The tip of these new sprouts typically follow the gradient of the VEGF to find the distressed cell. During this process, the newly-formed blood vessel can also branch and sprout new blood vessels. When the branches meet with other branches, they merge (this process is called anastomosis) and a blood circulation through the new vessel starts. The blood circulation helps nearby stem and tissue cells, which then stop emitting growth factors. This event is called angiogenesis or vascularization. Please see Figure 1 for an illustration of this process.

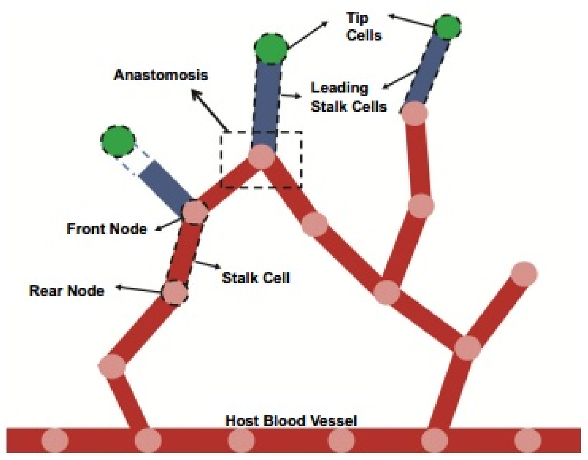


Figure 1: Illustration of vascularization, including the tip cells (active cells), the fixed cells (stalk cells), and anastomosis. [19]

Vascularization is a key process in tissue development. When cells that are emitting VEGF cannot be reached in time by the new blood vessels, the cells first fall in hypoxia (i.e., lack of Oxygen) and then start dying. Hence the formation of healthy tissue depends on appropriate vascularization; the blood vessels need to spread out in the newly-formed tissue, invade into the depth, and need to form connections to allow blood circulation.

Though it is well-known that the VEGF is a major contributor to sprouting of new blood vessels and that the tip of the blood vessel typically follows the gradient of the VEGF, there are still unknown factors that affect vascularization. Moreover, the VEGF distribution becomes more uniform as we get further away from the source of the emission and hence the gradient does not necessarily point to the distressed cell. Therefore, given our knowledge of the VEGF distribution the environment, the blood vessels do not necessarily follow a deterministic path; they also do a bit of exploration. This is where the uncertainty reasoning capabilities of probabilistic graphical models become handy for modeling vascularization.

In this paper, we model the vascularization process through dynamic Bayesian networks (DBNs) to enable tissue engineering researchers to reason with spatial and temporal growth of blood vessels. With the help of DBNs, the researchers can formulate and query the DBNs and try a number of parameter settings, without the need to experiment with every one of them in the lab. This process allows the researchers to gain further insights and formulate new in-vivo (on animals) and in-vitro (on glass) experiments.

3 APPROACH

In this section, we describe our DBN model for vascularization. We made a number of assumptions to simplify the model. In this model, we assume a 2D structure, whereas in real-life scenarios, the tissue obviously has a 3D structure. In this 2D structure, which is illustrated in Figure 2, as also assumed in [1], we assume that the blood vessel grows bottom-up towards north. Therefore, the status of a location at time t depends on: i) its status at time t, and ii) the statuses of its south neighbors at t.

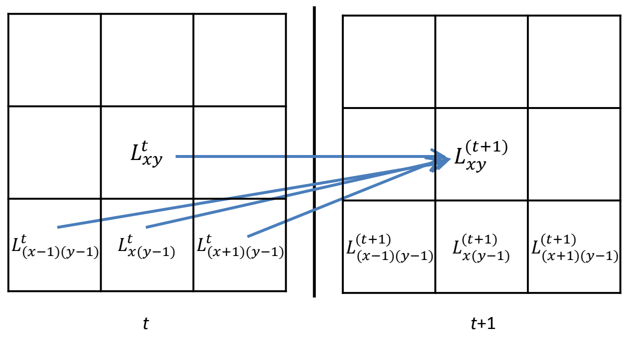


Figure 2: The tissue grid. Each cell of the grid represents a location, which can be `Empty`, or can be occupied with an `Active Cell` or `Stalk Cell`. Each location is represented as a random variable in DBN.

To simplify the notation, when we refer to a generic location L^t_{xy}, we will drop the subscripts and hence simply use L^t, and when we refer to its neighbors at its south $L^t_{(x-1)(y-1)}$, $L^t_{x(y-1)}$, and $L^t_{(x+1)(y-1)}$ we will simply use L^t_{SW}, L^t_S, and L^t_{SE}, corresponding to neighbors at south west, south, and south east, respectively. We illustrate the relevant 2-time slice dynamic Bayesian network in Figure 3.

Each location on the 2D grid is a random variable, representing whether that location is `Empty`, or occupied by a blood vessel cell. Blood vessel cells are two types: the tip of a blood vessel that has the potential to grow (henceforth called an `Active Cell`) or the body of the blood vessel (henceforth called the `Stalk Cell`). Therefore, the domain of random variable is [`Active Cell`, `Stalk Cell`, `Empty`], abbreviated henceforth as [AC, SC, E].

We model the conditional probability distribution, (CPD), $P\left(L^{(t+1)}|L^t, L^t_{SW}, L^t_S, L^t_{SE}\right)$ as a tree CPD as illustrated in Figure 4. To give a simple overview, at each step in time, an `Active Cell` elongates and moves into a nearby `Empty` location, forming the body of the blood vessel (i.e., `Stalk Cell`) in the process. The transitions are:

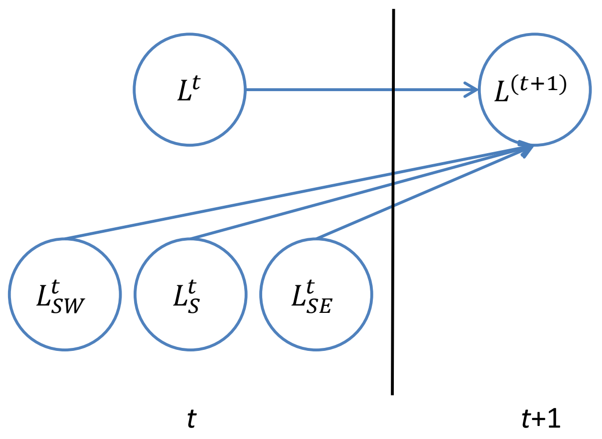
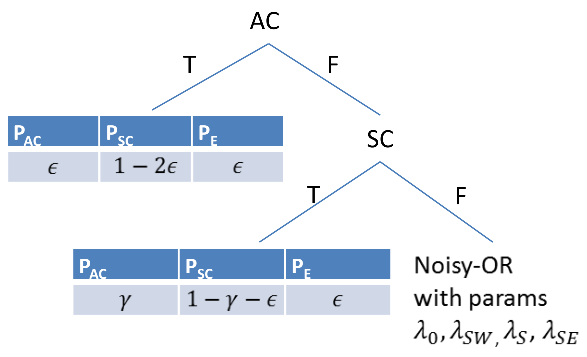


Figure 3: A two-time slice representation of the DBN. A location at a time $t+1$ has four parents: itself at time t and its lower neighbors at time t.



AC
T
F
P_{AC} P_{SC} P_E
ϵ $1-2\epsilon$ ϵ
SC
T
F
P_{AC} P_{SC} P_E
γ $1-\gamma-\epsilon$ ϵ
Noisy-OR
with params
$\lambda_0, \lambda_{SW}, \lambda_S, \lambda_{SE}$

Figure 4: The CPD for $P(L^{(t+1)}|L^t, L^t_{SW}, L^t_S, L^t_{SE})$.

- The tip of a blood vessel (AC) at time t becomes the body (SC) at time $t+1$. That is $P\left(L^{(t+1)}|L^t = AC, L^t_{SW}, L^t_S, L^t_{SE}\right) = P\left(L^{(t+1)}|L^t = AC\right) = \langle\epsilon, 1-2\epsilon, \epsilon\rangle$, where ϵ is a small noise parameter.

- A `Stalk Cell` at time t either continues to remain a `Stalk Cell` at time $t+1$ or it might become `Active Cell` with probability γ to sprout a new blood vessel branch. That is, $P\left(L^{(t+1)}|L^t = SC, L^t_{SW}, L^t_S, L^t_{SE}\right) = P\left(L^{(t+1)}|L^t = SC\right) = \langle\gamma, 1-\gamma-\epsilon, \epsilon\rangle$. We refer to γ as the sprout possibility.

- An `Empty` location at time t will remain `Empty` at time $t+1$ if none of its SW, S, or SE neighbors are `Active Cell` at time t; if there is an `Active Cell` at one or more of those neighboring locations at time t, one of them might elongate to this `Empty` location at time $t+1$. The probability of that an `Empty` location being occupied by an `Active Cell` at time $t+1$ is modeled as a Noisy-OR of its neighboring locations. That is $P\left(L^{(t+1)} = AC | L^t = E, L^t_{SW}, L^t_S, L^t_{SE}\right)$ is a Noisy-OR of $L^t_{SW}, L^t_S, L^t_{SE}$, with parameters λ_0, λ_{SW}, λ_S, and λ_{SE}, where λ_0 is leak parameter, and λ_{SW}, λ_S, and λ_{SE} corresponds to the possibility that an `Active Cell` elongates in the NE, N, or NW direction.[1] The magnitude of λ_{SW}, λ_S, and λ_{SE} are determined by the VEGF gradient. We refer to various configurations of the λ parameters as the growth patterns.

4 EXPERIMENTAL SETUP, RESULTS, AND INSIGHTS

In this section, we describe the experiments we performed using various settings for the growth pattern (λ) and sprout (γ) parameters. In all the experiments to follow, we set the noise ϵ and the leak λ_0 parameters to 0.01. For the growth pattern, we present results for two settings:

- `straight-growth`: $\langle \lambda_{SW}, \lambda_S, \lambda_{SE} \rangle = \langle 0.01, 0.98, 0.01 \rangle$. For this pattern, the blood vessel follows a straight line, growing towards north.
- `uniform-growth`: $\langle \lambda_{SW}, \lambda_S, \lambda_{SE} \rangle = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$. For this pattern, the blood vessel has equal chance of growing towards north, north west, or north east.

For the sprout possibility, that is a `Stalk Cell` turning into an `Active Cell`, we present results for two settings:

- `seldom-sprout`: $\gamma = 0.01$. For this setting, the `Stalk Cell` has very small chance (probability of 0.01) of becoming an `Active Cell` in the next time step.
- `always-sprout`: $\gamma = 0.98$. For this setting, the `Stalk Cell` has 0.98 probability of becoming active in the next step. This is quite an unrealistic setting; we present it only for didactic purposes.

We present results for four possible configurations: the cross-product of the growth patterns and sprout possibilities. We first provide detailed results on a 3×3 grid over three time slices. Then, we present results on a bit larger scale, 9×9, over nine time slices. Finally, we present a framework where we quantify the uncertainty over the predictions on the last time slice and discuss how it is affected by the growth patterns and sprout possibilities.

For inference, in the 3×3 case, we used exact inference. For the 9×9 case, we used forward sampling. Note that we are able to use forward sampling in our settings because we provide the initial condition (all locations at time $t = 0$) as evidence and compute probabilities for the remaining time slices.

4.1 Detailed Results for 3×3

In this toy setting, we provide the evidence for the initial configuration of the experiment, i.e., we provide evidence for all locations for time $t = 0$, and compute probabilities for all locations for all future time slices. That is, we compute $P(\mathcal{L}^1, \mathcal{L}^2 | \mathcal{L}^0)$, where $\mathcal{L}^t$ denotes *all* locations at time t. For $t = 0$, we provide the evidence as follows: the middle of the bottom row is set as the tip of the blood vessel (i.e, $L^0_{x=1,y=0} = AC$) and the rest of the locations are set as `Empty`. Figure 5 illustrates this setting.

E	E	E
E	E	E
E	AC	E

Figure 5: The initial configuration for the 3×3 grid.

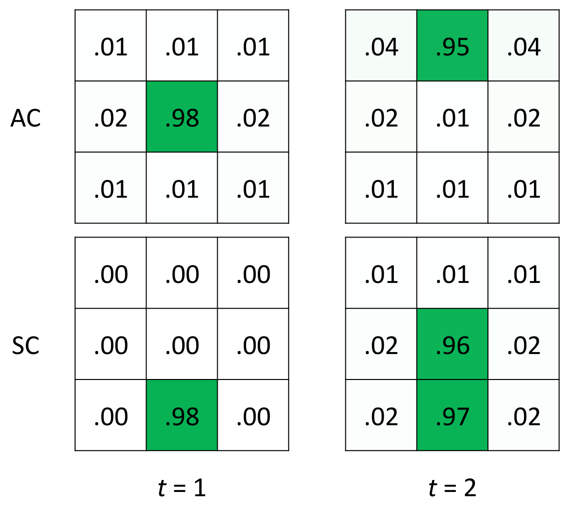
The `straight-growth` results are presented in Figures 6 and 7, and `uniform-growth` results are presented in Figures 8 and 9.

The simplest setting where the blood vessel grows in a straight path and that does not sprout at all (Figure 6) is fairly straightforward to analyze. The tip of the blood vessel migrates one location towards north at each step, forming the body of the vessel along the process. This setting, therefore, serves as a sanity check.

In the next setting, which is presented in Figure 7, we keep the growth pattern the same (`straight-growth`) but increase the sprout possibility to 0.98 (`always-sprout`). In this setting, the blood vessel grows towards north as expected. Unlike the `seldom-sprout` case, however, a `Stalk Cell` at time $t = 1$ became active at time $t = 2$.

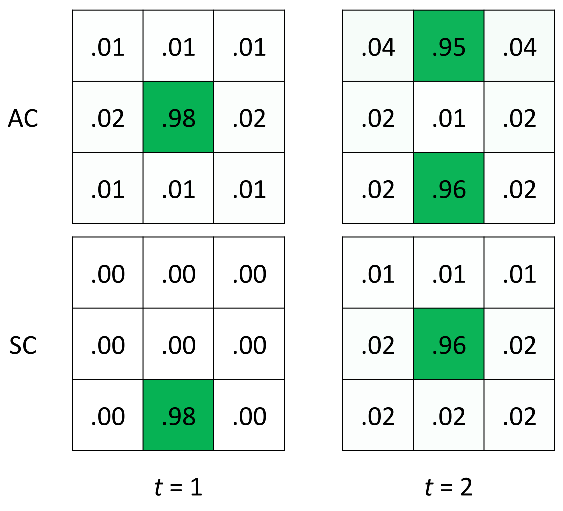
Next, we present results for the `uniform-growth` cases. In this setting, the blood vessel has uniform

[1]Note that λ_{SW} denotes the probability that an `Active Cell` at the SW of an `Empty` location will move to this `Empty` location; hence λ_{SW} denotes the possibility that an `Active Cell` at SW moves in the NE direction to occupy an `Empty` location.



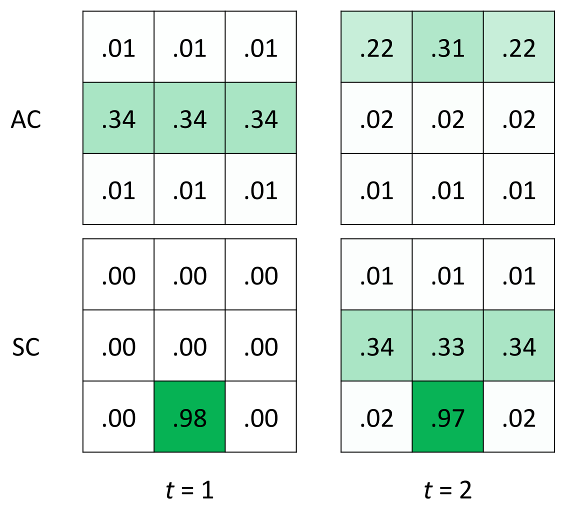
	t = 1			t = 2		
AC	.01	.01	.01	.04	.95	.04
	.02	.98	.02	.02	.01	.02
	.01	.01	.01	.01	.01	.01
SC	.00	.00	.00	.01	.01	.01
	.00	.00	.00	.02	.96	.02
	.00	.98	.00	.02	.97	.02

Figure 6: `straight-growth`, `seldom-sprout`. This is the most straightforward setting where the blood vessel grows one step at a time towards north.



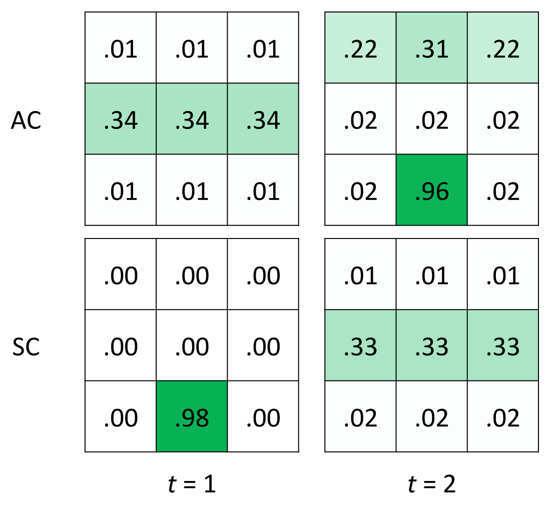
	t = 1			t = 2		
AC	.01	.01	.01	.04	.95	.04
	.02	.98	.02	.02	.01	.02
	.01	.01	.01	.02	.96	.02
SC	.00	.00	.00	.01	.01	.01
	.00	.00	.00	.02	.96	.02
	.00	.98	.00	.02	.02	.02

Figure 7: `straight-growth`, `always-sprout`. The blood vessel grows towards north. A location that is a `Stalk Cell` at time $t = 1$ ($L_{x=1,y=0}$) becomes `Active Cell` at time $t = 2$.



	t = 1			t = 2		
AC	.01	.01	.01	.22	.31	.22
	.34	.34	.34	.02	.02	.02
	.01	.01	.01	.01	.01	.01
SC	.00	.00	.00	.01	.01	.01
	.00	.00	.00	.34	.33	.34
	.00	.98	.00	.02	.97	.02

Figure 8: `uniform-growth`, `seldom-sprout`. The blood vessel has equal probability to grow in all three directions. A `Stalk Cell` at time t will most likely remain as `Stalk Cell` at $t + 1$.



	t = 1			t = 2		
AC	.01	.01	.01	.22	.31	.22
	.34	.34	.34	.02	.02	.02
	.01	.01	.01	.02	.96	.02
SC	.00	.00	.00	.01	.01	.01
	.00	.00	.00	.33	.33	.33
	.00	.98	.00	.02	.02	.02

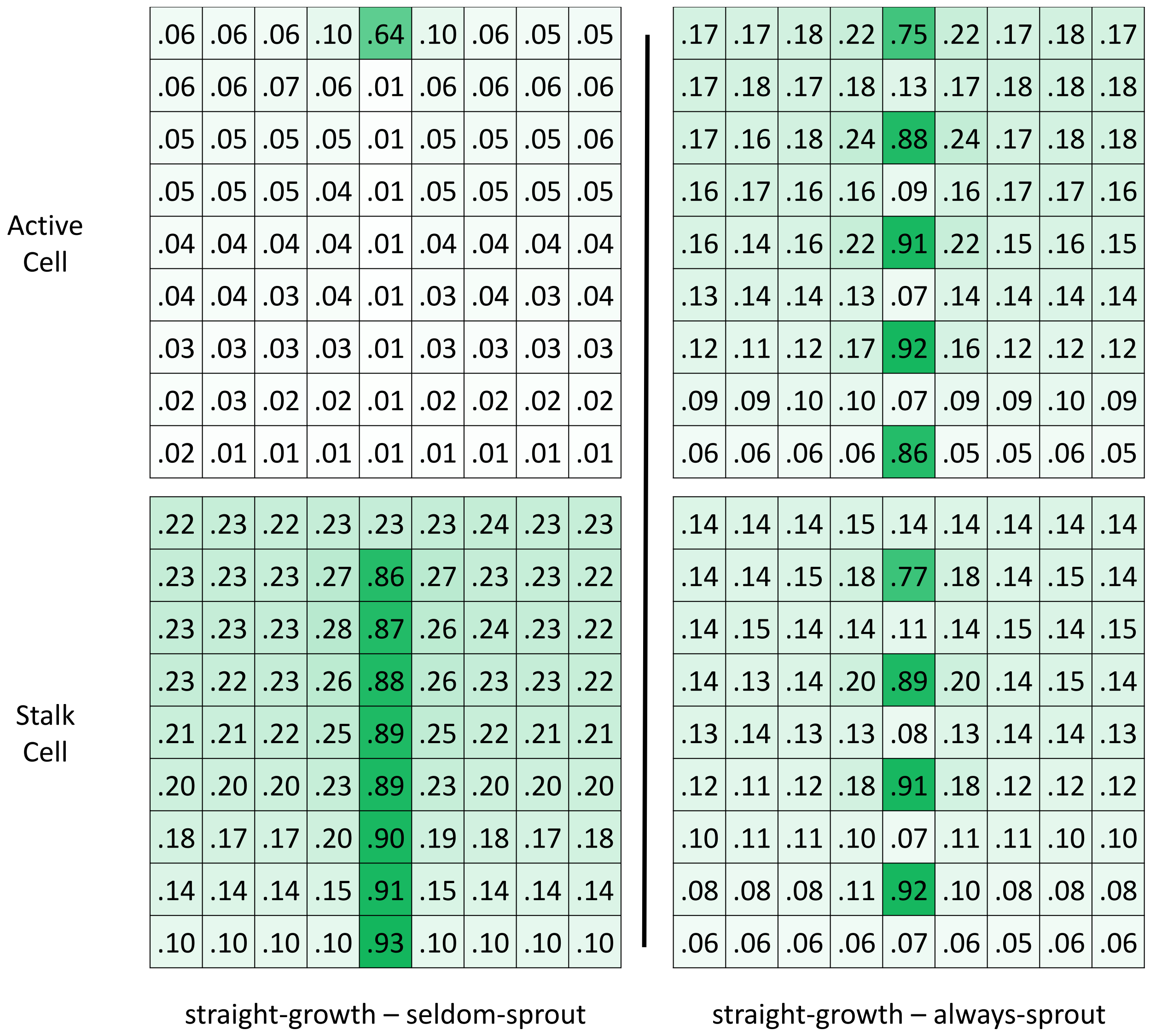
Figure 9: `uniform-growth`, `always-sprout`. The blood vessel has equal probability to grow in all three directions. A `Stalk Cell` will most likely become `Active Cell` in the next time slice.

probability of growing towards NW, N, and NE. In the `seldom-sprout` case (Figure 8), the `Active Cell` at $t = 0$ turned into a `Stalk Cell` at time $t = 1$ and remained a `Stalk Cell` at time $t = 2$. The `Active Cell`, unlike the `straight-growth` case, has equal probability of moving in all three directions. In the last time step, the middle of the top row has higher probability (.31) than the sides (.22) simply because the middle location can be reached from more locations compared to the side locations. The `always-sprout` case (Figure 9) is similar except a `Stalk Cell` at $t = 1$ becomes an `Active Cell` at $t = 2$.

These toy experiments provide insights into how the process typically works. Next, we present results for the 9×9 grid.

4.2 Summary Results for 9×9

Similar to the 3×3 grid, we provide evidence for $t = 0$ case and compute probabilities for the remaining eight time slices. In the initial configuration, the middle


straight-growth – seldom-sprout

Active Cell

.06	.06	.06	.10	.64	.10	.06	.05	.05
.06	.06	.07	.06	.01	.06	.06	.06	.06
.05	.05	.05	.05	.01	.05	.05	.05	.06
.05	.05	.05	.04	.01	.05	.05	.05	.05
.04	.04	.04	.04	.01	.04	.04	.04	.04
.04	.04	.03	.04	.01	.03	.04	.03	.04
.03	.03	.03	.03	.01	.03	.03	.03	.03
.02	.03	.02	.02	.01	.02	.02	.02	.02
.02	.01	.01	.01	.01	.01	.01	.01	.01

Stalk Cell

.22	.23	.22	.23	.23	.23	.24	.23	.23
.23	.23	.23	.27	.86	.27	.23	.23	.22
.23	.23	.23	.28	.87	.26	.24	.23	.22
.23	.22	.23	.26	.88	.26	.23	.23	.22
.21	.21	.22	.25	.89	.25	.22	.21	.21
.20	.20	.20	.23	.89	.23	.20	.20	.20
.18	.17	.17	.20	.90	.19	.18	.17	.18
.14	.14	.14	.15	.91	.15	.14	.14	.14
.10	.10	.10	.10	.93	.10	.10	.10	.10

straight-growth – always-sprout

Active Cell

.17	.17	.18	.22	.75	.22	.17	.18	.17
.17	.18	.17	.18	.13	.17	.18	.18	.18
.17	.16	.18	.24	.88	.24	.17	.18	.18
.16	.17	.16	.16	.09	.16	.17	.17	.16
.16	.14	.16	.22	.91	.22	.15	.16	.15
.13	.14	.14	.13	.07	.14	.14	.14	.14
.12	.11	.12	.17	.92	.16	.12	.12	.12
.09	.09	.10	.10	.07	.09	.09	.10	.09
.06	.06	.06	.06	.86	.05	.05	.06	.05

Stalk Cell

.14	.14	.14	.15	.14	.14	.14	.14	.14
.14	.14	.15	.18	.77	.18	.14	.15	.14
.14	.15	.14	.14	.11	.14	.15	.14	.15
.14	.13	.14	.20	.89	.20	.14	.15	.14
.13	.14	.13	.13	.08	.13	.14	.14	.13
.12	.11	.12	.18	.91	.18	.12	.12	.12
.10	.11	.11	.10	.07	.11	.11	.10	.10
.08	.08	.08	.11	.92	.10	.08	.08	.08
.06	.06	.06	.06	.07	.06	.05	.06	.06

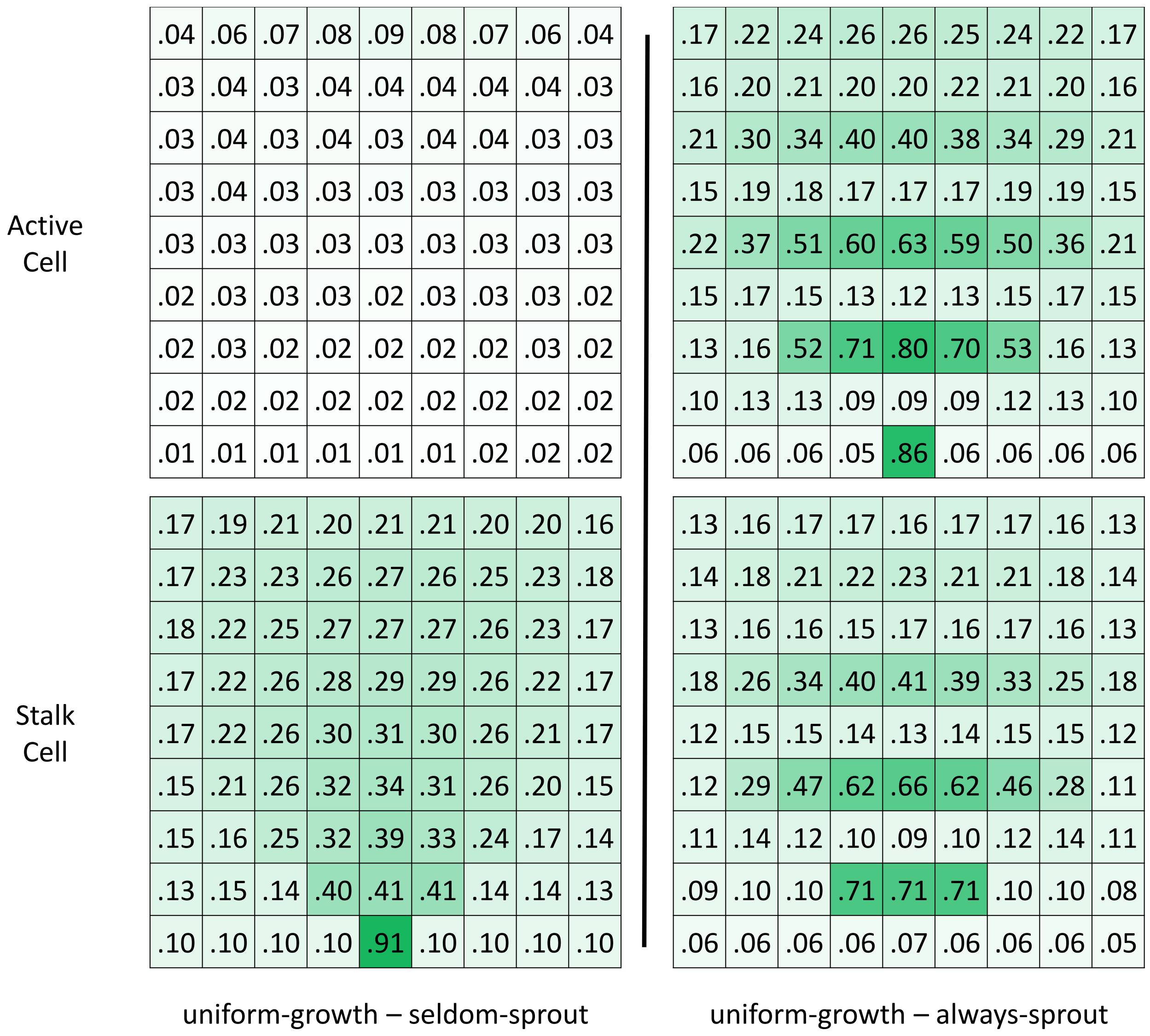
Figure 10: AC and SC probabilities for the 9×9 grid at the last time slice (t=8) for `straight-growth`. Left: `seldom-sprout`. Right: `always-sprout`. On the right, it is apparent that the `Stalk Cells` and `Active Cells` alternate.

of the bottom row is set as an `Active Cell` and the remaining locations are set as `Empty`. Due to space limitations, we present results for only the last time slice, $t = 8$. The `straight-growth` case is shown in Figure 10 and the `uniform-growth` case is shown in Figure 11.

In the `straight-growth seldom-sprout` case (the left side of Figure 10), we see a straight blood vessel for the middle of the grid, where every cell of the blood vessel except the tip is a `Stalk Cell` and the tip is an `Active Cell`, as expected. In the `always-sprout` case (the right side of Figure 10), the `Stalk Cells` and `Active Cells` alternate, again as expected.

In the `uniform-growth seldom-sprout` case (the left side of Figure 11), the blood vessel can be anywhere on the grid, except, as expected, the middle locations have higher probability. In the `always-sprout` case (the right side of Figure 11), the `Stalk Cells` and `Active Cells` alternate, as expected. Additionally, the probabilities for locations being a blood vessel (either Stalk to Active) are higher in the `always-sprout` case compared to the `seldom-sprout` case, again as expected.

The results so far have been nothing surprising, but only confirming our expectations. The value of the DBNs, however, lies at their capability to reason with spatial and temporal uncertainty as well as their potential for future directions. We discuss one of the



Active Cell — uniform-growth – seldom-sprout

.04	.06	.07	.08	.09	.08	.07	.06	.04
.03	.04	.03	.04	.04	.04	.04	.04	.03
.03	.04	.03	.04	.03	.04	.04	.03	.03
.03	.04	.03	.03	.03	.03	.03	.03	.03
.03	.03	.03	.03	.03	.03	.03	.03	.03
.02	.03	.03	.03	.02	.03	.03	.03	.02
.02	.03	.02	.02	.02	.02	.02	.03	.02
.02	.02	.02	.02	.02	.02	.02	.02	.02
.01	.01	.01	.01	.01	.01	.02	.02	.02

Active Cell — uniform-growth – always-sprout

.17	.22	.24	.26	.26	.25	.24	.22	.17
.16	.20	.21	.20	.20	.22	.21	.20	.16
.21	.30	.34	.40	.40	.38	.34	.29	.21
.15	.19	.18	.17	.17	.17	.19	.19	.15
.22	.37	.51	.60	.63	.59	.50	.36	.21
.15	.17	.15	.13	.12	.13	.15	.17	.15
.13	.16	.52	.71	.80	.70	.53	.16	.13
.10	.13	.13	.09	.09	.09	.12	.13	.10
.06	.06	.06	.05	.86	.06	.06	.06	.06

Stalk Cell — uniform-growth – seldom-sprout

.17	.19	.21	.20	.21	.21	.20	.20	.16
.17	.23	.23	.26	.27	.26	.25	.23	.18
.18	.22	.25	.27	.27	.27	.26	.23	.17
.17	.22	.26	.28	.29	.29	.26	.22	.17
.17	.22	.26	.30	.31	.30	.26	.21	.17
.15	.21	.26	.32	.34	.31	.26	.20	.15
.15	.16	.25	.32	.39	.33	.24	.17	.14
.13	.15	.14	.40	.41	.41	.14	.14	.13
.10	.10	.10	.10	.91	.10	.10	.10	.10

Stalk Cell — uniform-growth – always-sprout

.13	.16	.17	.17	.16	.17	.17	.16	.13
.14	.18	.21	.22	.23	.21	.21	.18	.14
.13	.16	.16	.15	.17	.16	.17	.16	.13
.18	.26	.34	.40	.41	.39	.33	.25	.18
.12	.15	.15	.14	.13	.14	.15	.15	.12
.12	.29	.47	.62	.66	.62	.46	.28	.11
.11	.14	.12	.10	.09	.10	.12	.14	.11
.09	.10	.10	.71	.71	.71	.10	.10	.08
.06	.06	.06	.06	.07	.06	.06	.06	.05

Figure 11: AC and SC probabilities for the 9×9 grid at the last time slice (t=8) for `uniform-growth`. Left: `seldom-sprout`. Right: `always-sprout`. In both cases, the blood vessel can grow uniformly in each direction and the middle locations have higher probability of having a blood vessel simply because they can be reached from multiple locations. In the `always-sprout` case, `Stalk Cells` and `Active Cells` alternate.

future directions here supplemented with some preliminary results, and discuss more future directions in Section 6.

4.3 Quantifying Uncertainty

Given an initial condition, $\mathcal{L}^0$, the tissue engineers are interested in the final status of the tissue, $\mathcal{L}^T$, where T denotes the final step of the experiment. Because real-world experiment take a long time, mostly weeks, they would like to be able to stop an experiment at time $t < T$ and still be able to reason about time T. Therefore, they are interested in the following question: *given an initial condition $\mathcal{L}^0$, if we stop the experiment at time t, what is the uncertainty over $\mathcal{L}^T$?* More practically: *when is the earliest time we can stop an experiment so that the uncertainty over the last time slice is below a pre-specified target σ?*. It is important to note that when an experiment is stopped, the researchers dissect the tissue to analyze its properties, such as vascularization, and hence the experiment cannot continue beyond that point.

Given an uncertainty measure, this question can be formulated rather straightforwardly using DBNs. Let $UNC\left(P\left(\mathcal{L}^T|l^0, l^t\right)\right)$ denote the uncertainty over the predictions over the last time slice, given the initial condition $\mathcal{L}^0 = l^0$ and the status of the experiment at time t, $\mathcal{L}^t = l^t$. Then, we simply need to find

$$\underset{t<T}{\operatorname{argmin}}\, UNC\left(P\left(\mathcal{L}^T|l^0, l^t\right)\right) < \sigma \tag{1}$$

Obviously, even though we know the initial conditions l^0, we do not know the status of the experiment at time $t > 0$ unless we stop the experiment. Therefore, we need to take an expectation over all possible outcomes at time t:

$$\underset{t<T}{\operatorname{argmin}} \sum_{l^t} P\left(\mathcal{L}^t = l^t|l^0\right) UNC\left(P\left(\mathcal{L}^T|l^0, l^t\right)\right) < \sigma \tag{2}$$

where the subscript l^t in the summation ranges over all possible configurations of $\mathcal{L}^t$.

Unfortunately the number of all possible configurations for an $n \times n$ grid is $3^{n\times n}$, which is clearly intractable to solve. We leave a more systematic solution for future direction and present results for the case where the summation is replaced with the most probable $l^t|l^0$. For the UNC measure, there are a number of possibilities, including the entropy. We present results where we compute the conditional error of the most probable blood vessel path. That is, for the most-likely blood vessel path, we sum $1 - P(SC|l^t, l^0)$ for the body of the vessel and add $1 - P(AC|l^t, l^0)$ for the tip of the blood vessel.

We experimented with the 9×9 grid and we set the sprout possibility to $\gamma = 0.01$ so that the most probably path does not have any branches. We present the uncertainty values for `straight-growth` and `uniform-growth` patterns in Figure 12. The x axis represents the time we would stop the experiment and the y axis plots the uncertainty. As expected, the uncertainty is much higher for the `uniform-growth` case and that uncertainty goes down for both growth patterns as we provide evidence for later time steps.

We scratched only the surface of this important problem, leaving many interesting research problems for future work, some of which are discussed in Section 6.

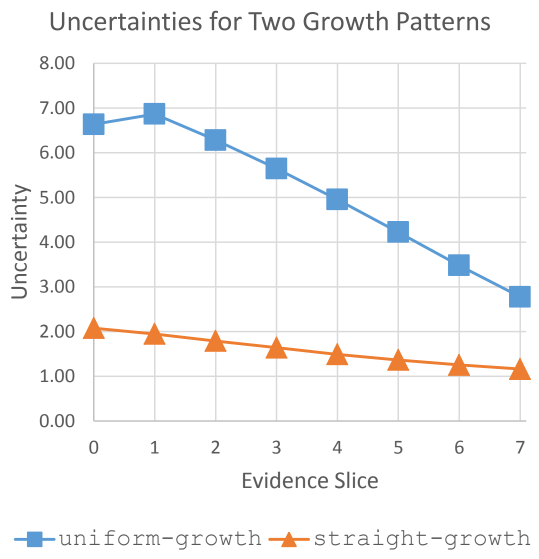


Figure 12: Uncertainties for two growth patterns. As expected, the `straight-growth` pattern is more predictable than the `uniform-growth` pattern. Additionally, providing evidence for later time slices results in lower uncertainties in the last time slice.

5 RELATED WORK

Tissue engineering experiments typically are performed in-vivo usually on mice and in-vitro in glass on lab. Researchers experiment with various settings including the porosity of the scaffold that the tissue is expected to hold on to, the VEGF distribution, and initial blood vessel sprout locations [24, 13, 16, 17].

On the computational side, various researchers have used agent-based modeling to simulate the tissue engineering process [1, 19, 3, 2, 9]. In these simulations, stem cells, tissue cells, and blood cells are modeled as agents and are provided rules that are often elicited from experts. These simulations allow researchers to experiment with a varying number of parameters, without having to perform in-vivo or in-vitro experiments. Some of the parameter settings that produce promising results are then tried in the lab. Based on the results obtained in the lab, the rules for the agents are updated and thus there is often a continuous feedback loop between the simulations and experiments.

Our DBN modeling is a complementary approach to the lab experiments and computational simulations. Because the whole process is inherently stochastic, obtaining the average behavior through experiments and simulations require many trials whereas DBNs provide a systematic, transparent, and modular mechanism to reason with uncertainty.

DBNs have been previously used for many practical applications. Examples include managing water resources [8], modeling environmental problems [23], driverless cars [14], gene regulatory networks [20, 22, 15], figure tracking [21], ranking [10], and speech recognition [25] to name a few. To the best of our knowledge, ours is the first probabilistic graphical model approach for modeling the tissue engineering process.

6 CURRENT LIMITATIONS AND FUTURE DIRECTIONS

There are two lines of work that we would like to pursue in the future. The first type is enriching the model, lifting some of the assumptions we made. The second type of work is a new line of research that we refer to as active inference, which we will describe shortly.

We made a series of simplifying assumptions in our current DBN model. One such assumption is that the tissue space is 2D, whereas in reality it is obviously 3D. The 2D assumption allowed us to work with much fewer random variables. Additionally, in 2D, the number of parents for a variable is four whereas in 3D, the number of parents is ten (itself in the previous time slice and nine locations under it). It is rather straightforward to move from 2D to 3D from a representation perspective. However, scalability both in terms of memory and computational time is a challenge.

Another assumption we made is that the gradient of the VEGF is fixed throughout the grid. That is, we assumed the λ and the γ values are fixed across the grid. In reality, however, the growth factor is expected to have steeper gradient when it is closer to the source of the distressed tissue cell and it is expected to be more uniform as we get further away from the distressed cell. Our simplifying assumption can be easily lifted by providing a growth factor distribution across the grid and then translating it into the necessary λ and γ parameters.

A limitation that is harder to address is scalability. In our experiment section (Section 4), we experimented with 3×3 and 9×9 grids. These were trivial to experiment with. In reality, however, we need to deal with thousands if not millions of random variables over a much longer period of time. This will raise scalability issues both in terms of memory and in terms of computation time. Lifted inference [7] can be used to address some of these challenges.

Another line of research is to formulate and run active inference for dynamic Bayesian networks [6, 5, 4, 12, 11]. Active inference is interested in the following question: *if we are given the opportunity to gather evidence to condition on but gathering evidence is costly, which variables and what time frames are the most cost-effective ones to condition on?*. We discussed the initial formulation of active inference and preliminary results in Section 4.3. However, many questions and challenges remain to be addressed. For example, given a target uncertainty threshold σ, how can we *efficiently* find the smallest time t, where $UNC(P(\mathcal{L}^T|l^0, l^t) < \sigma$, without searching all possible t values?

7 CONCLUSIONS

We presented a dynamic Bayesian network model for vascularization in engineered tissues. This DBN enables i) spatial and temporal reasoning for understanding of vascularization, ii) formulation and investigation of various parameter settings for vascularization, and iii) formulation of uncertainty and active information gathering to minimize uncertainty. We presented initial results that provide insights in to the vascularization process. Though the DBN model currently represents an oversimplification of the reality, it is the first and hence novel application of DBNs to vascularization. As such, it avails itself to many interesting research challenges and opportunities.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant no. IIS-1125412. We thank Ali Cinar, Judith Zawojewski, Eric Brey, Hamidreza Mehdizadeh, and Elif Bayrak for providing information and insights on tissue engineering.

References

[1] Arsun Artel, Hamidreza Mehdizadeh, Yu-Chieh Chiu, Eric M. Brey, and Ali Cinar. An agent-based model for the investigation of neovascularization within porous scaffolds. *Tissue Engineering Part A*, 17(17-18):2133–2141, 2011.

[2] Alexander M. Bailey, Bryan C. Thorne, and Shayn M. Peirce. Multi-cell agent-based simulation of the microvasculature to study the dynamics of circulating inflammatory cell trafficking. *The Journal of the Biomedical Engineering Society*, 35(6):916 – 936, 2007.

[3] Katie Bentley, Holger Gerhardt, and Paul A. Bates. Agent-based simulation of notch-mediated tip cell selection in angiogenic sprout initialisation. *Journal of Theoretical Biology*, 250(1):25 – 36, 2008.

[4] Mustafa Bilgic and Lise Getoor. Effective label acquisition for collective classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–51, 2008.

[5] Mustafa Bilgic and Lise Getoor. Reflect and correct: A misclassification prediction approach to active inference. *ACM Transactions on Knowledge Discovery from Data*, 3(4):1–32, November 2009.

[6] Mustafa Bilgic and Lise Getoor. Active inference for collective classification. In *Twenty-Fourth*

Conference on Artificial Intelligence (AAAI NECTAR Track), pages 1652–1655, 2010.

[7] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proceedings of IJCAI-05, 19th International Joint Conference on Artificial Intelligence*, 2005.

[8] J. Bromley, N. A. Jackson, O. J. Clymer, A. M. Giacomello, and F. V. Jensen. The use of Hugin to develop Bayesian networks as an aid to integrated water resource planning. *Environmental Modelling & Software*, 20(2):231–242, 2005.

[9] Damien P. Byrne, Damien Lacroix, Josep A. Planell, Daniel J. Kelly, and Patrick J. Prendergast. Simulation of tissue differentiation in a scaffold as a function of porosity, young's modulus and dissolution rate: Application of mechanobiological models in tissue engineering. *Biomaterials*, 28(36):5544 – 5554, 2007.

[10] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1–10. ACM, 2009. ISBN 978-1-60558-487-4.

[11] Daozheng Chen, Mustafa Bilgic, Lise Getoor, and David Jacobs. Dynamic processing allocation in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:2174–2187, 2011.

[12] Daozheng Chen, Mustafa Bilgic, Lise Getoor, David Jacobs, Lilyana Mihalkova, and Tom Yeh. Active inference for retrieval in camera networks. In *Workshop on Person Oriented Vision*, 2011.

[13] Yu-Chieh Chiu, Sevi Kocagoz, Jeffery C Larson, and Eric M Brey. Evaluation of physical and mechanical properties of porous poly (ethylene glycol)-co-(l-lactic acid) hydrogels during degradation. *PLoS One*, 8:4, 2013.

[14] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The BATmobile: Towards a Bayesian automated taxi. In *IJCAI*, volume 95, 1995.

[15] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[16] B. Jiang, B. Akar, T.M. Waller, J.C. Larson, A.A. Appel, and E.M. Brey. Design of a composite biomaterial system for tissue engineering applications. *Acta Biomaterialia*, 2013.

[17] Bin Jiang, Thomas M Waller, Jeffery C Larson, Alyssa A Appel, and Eric M Brey. Fibrin-loaded porous poly(ethylene glycol) hydrogels as scaffold materials for vascularized tissue formation. *Tissue engineering. Part A*, 19:224–234, 2013.

[18] R Langer and JP Vacanti. Tissue engineering. *Science*, 260(5110):920–926, 1993.

[19] Hamidreza Mehdizadeh, Sami Sumo, Elif S. Bayrak, Eric M. Brey, and Ali Cinar. Three-dimensional modeling of angiogenesis in porous biomaterial scaffolds. *Biomaterials*, 34(12):2875–2887, 2013.

[20] Zou Min and Suzanne D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.

[21] Vladimir Pavlovic, James M. Rehg, Tat-Jen Cham, and Kevin P. Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 94–101 vol.1, 1999.

[22] Bruno-Edouard Perrin, Liva Ralaivola, Aurelien Mazurie, Samuele Bottani, Jacques Mallet, and Florence d'Alche Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19(suppl 2):ii138–ii148, 2003.

[23] L Uusitalo. Advantages and challenges of Bayesian networks in environmental modelling. *Ecological modelling*, 3(203):312–318, 2007.

[24] Shoufeng Yang, Kah-Fai Leong, Zhaohui Du, and Chee-Kai Chua. The design of scaffolds for use in tissue engineering. part I. traditional factors. *Tissue Engineering*, 7:679–689, 2004.

[25] Geoffrey Zweig and Stuat Russell. Speech recognition with dynamic Bayesian networks. In *AAAI-98 Proceedings*, 1998.

Trade-Based Asset Models for Combinatorial Prediction Markets

Wei Sun, Shou Matsumoto, Robin Hanson, Kathryn Laskey, Charles Twardy
George Mason University
Fairfax, VA 22030

Brandon Goldfedder
Gold Brand Software
Herndon, VA 20170

Abstract

A prediction market allows a group of traders to form a consensus probability distribution by entering into agreements that pay off contingent on events of interest. A combinatorial prediction market allows conditional trades or trades on Boolean combinations of events to form a joint distribution over many related events. Sun et al. (2012) showed how to use a junction tree to update both the consensus joint distribution and each user's assets in a combinatorial prediction market. Because a separate asset junction tree is maintained for each user on the joint space, this approach is very inefficient in the typical case where most users trade sparsely with respect to the joint space. Further, any changes to the global junction tree must be mirrored across all users. We demonstrate large efficiency gains from divorcing the probability and asset data structures, dynamically building a separate asset junction tree for each user. The trade-based asset model has asset blocks as the basic units involving questions being traded only. We compare a simple block-iteration method against a more sophisticated user-specific junction tree, analyzing conditions under which each approach is faster. Our asset model has been deployed in SciCast[1], a combinatorial prediction market for science and technology forecasting.

1 Introduction

A prediction market is a market formed for the purpose of making predictions about events of interest. Participants provide inputs either by directly editing a consensus probability distribution or by buying and selling assets whose prices can be interpreted as probabilities. Prediction markets have demonstrated their value for aggregating collective expertise (Arrow et al., 2008).

Combinatorial prediction markets allow forecasts not only on base events, but also on conditional and/or Boolean combinations of events (Hanson, 2007). A market-maker-based combinatorial market (Hanson, 2007) allows a user to trade on any event at any time by interacting with an automated market maker which sets the price according to a market scoring rule. The market maker provides a number of functions: it processes trades on demand, manages a consistent joint probability distribution over the base events, can be queried for any user's expected assets, disallows any trade that could allow a user's assets negative, and pays off users when the state of any event becomes known.

Sun et al. (2012) presented an approach to performing these market maker functions under the assumption that the joint distribution can be represented in factored form as a junction tree, and trades are required to respect the conditional independence relationships encoded in the junction tree. Their approach maintains parallel junction trees, one for the joint distribution and one for the assets of each user. In practice, most users tend to trade sparsely relative to the joint probability space. Therefore, using the same global junction tree for all users is very inefficient for both storage and computation. Another problem is that any change to the probability structure (e.g., adding or resolving a question; adding or removing a link) must be mirrored across all users' asset structures.

This paper describes a new approach to managing assets in a market-maker-based combinatorial prediction market. The basic data structure is the asset block, which compactly represents a set of trades made by a user. A user's asset model consists of a set of asset blocks representing the user's entire trade history.

[1]https://SciCast.org/

Graph transformations are applied to transform the collection of asset blocks into an asset junction tree. The asset junction tree serves as the computational framework for computing the user's minimum assets, expected assets, and other quantities of interest.

2 Trade-based Asset Model

An individual asset model for each user is constructed from the user's trade history and updated incrementally with each trade. A data structure called an *asset block* groups the user's trades on a set of questions and represents gains and losses from those trades. An asset block $B = (\mathbf{V}_B, \delta_B)$ consists of block variables $\mathbf{V}_B$ and a block asset function δ_B that maps states $\mathbf{v}_B$ of $\mathbf{V}_B$ to real numbers $\delta_B(\mathbf{v}_B)$.

A collection of asset blocks is a compact representation of a user's gains or losses in any joint state. This user-specific asset representation can be exploited for efficient calculation of expected and conditional minimum assets. If asset blocks are organized according to trades, it can be shown that the user's assets $a^u_{\mathbf{v}}$ are *additively decomposable* with respect to the set $\{B\}_{B\in\mathcal{B}}$ of asset blocks. For any arbitrary edit $x(t|\mathbf{H})$ (H can be empty), logarithmic market scoring rule provides

$$a^u_{\mathbf{v}} + b \log \frac{x(t|\mathbf{H})}{p(t|\mathbf{H})} \tag{1}$$

When assets are additively decomposable according to $\{B\}_{B\in\mathcal{B}}$, the asset blocks can be assembled into a computational structure that supports asset management computations.

The *Dynamic Asset Cluster (DAC) model* begins with an undirected asset graph assembled from the user's trades, where each node in the graph is associated with an asset block. The asset blocks are constructed in a manner that ensures additive separability. The asset graph is transformed into an asset junction tree, guaranteeing that the original asset blocks will not be split when new cliques are formed in the asset junction tree. The steps are:

1. Create an undirected trade graph $\mathcal{G}$ by pairwise connecting all variables in each asset block.
2. Triangulate $\mathcal{G}$ to make a triangulated graph $\mathcal{T}$, and identify all cliques from $\mathcal{T}$.
3. Use a standard algorithm to form a junction tree $\mathcal{J}$ from the triangulated graph $\mathcal{T}$.
4. Assign the asset function for each asset block to exactly one clique that contains all the block variables for the asset block.
5. Create an asset table for each clique by adding the block asset functions for blocks assigned to the clique.

Given the asset junction tree, local propagation can be used to perform the following tasks:

- *Calculate conditional minimum assets.* Min-propagation is used to return a global asset minimum, and the user is prevented from making trades that allow minimum assets to become negative. (Sun et al., 2012)
- *Calculate expected assets.* Expected assets are calculated by finding the joint consensus probability for each clique, calculating clique expected assets, and summing the over cliques.

3 Conclusion

To test the algorithm, we designed several different scenarios and conducted empirical comparisons between *DAC* and a simpler solution in which we iterate over a joint set of all overlapping variables (called global separator *GS*). Experimental results show both advantages and disadvantages for different cases. Inference for each is exponential in its respective treewidth, with *DAC* eventually winning due to its generally smaller treewidth. Analysis of expected use cases and empirical comparisons show *GS* is preferable when the number of overlaps is less than about 8, or the number of entries in clique tables is below about 500. These limits are likely to hold in SciCast for the near future.

References

Arrow, K., Forsythe, R., Gorham, M., Hahn, R., Hanson, R., Ledyard, J. O., Levmore, S., Litan, R., Milgrom, P., Nelson, F. D., Neumann, G. R., Ottaviani, M., Schelling, T. C., Shiller, R. J., Smith, V. L., Snowberg, E., Sunstein, C. R., Tetlock, P. C., Tetlock, P. E., Varian, H. R., Wolfers, J., and Zitzewitz, E. (2008). The promise of prediction markets. *Science*, 320(5878):877–878.

Hanson, R. (2007). Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15.

Sun, W., Hanson, R., Laskey, K. B., and Twardy, C. (2012). Probability and asset updating using bayesian networks for combinatorial prediction markets. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12)*, Catalina, CA.

An Object-Oriented Dynamic Bayesian Decision Network Model for Grasslands Adaptive Management

Owen Woodberry
Bayesian Intelligence Ptd Ltd

Jess Millett-Riley, Steve Sinclair
Arthur Rylah Institute for Environmental Research
Dept. of Environment and Primary Industries Victoria, Australia

Ann Nicholson
Faculty of IT Monash University, Australia

Abstract

The Victorian State Government is reserving 15,000 hectares of land to protect native grasslands in to the west of Melbourne, Australia, to be managed by the Department of Environment and Primary Industries (DEPI). The WGR currently contains a mixture of high quality native grasslands, degraded grasslands and non-native vegetation including improved pasture and cropland. Managing these areas for conservation will require a complex management approach involving weed control, biomass management using fire and grazing, cropland retirement and restoration involving the re-introduction of native plants and their seeds. The reserve must be managed as soon as land is required, but the best management techniques are largely unknown, thus an adaptive management approach -- where management and monitoring are adjusted overtime as understanding of the ecosystem's response to management improves. In order to assist adaptive management, Bayesian network technology was chosen to model ecological change in grassland ecosystem, to provide probabilistic predictions to evaluate management actions (e.g. weed control, fire) and to justify the choice of actions to be trialed within the reserve.

The Western Grasslands model is a complex BN, employing a number of extensions to the basic BN structure, as it is: a dynamic BN, representing the change in state variables over time, with a seasonal time steps, rolled-out for a 30 year prediction window; a decision network, with decision nodes representing management options grouped into management strategies, which are sequences of actions across seasons, and utility nodes which represent the costs associated with interventions and the environmental value of the site; an object oriented model, to manage the complexity of the number of species and seasonal transitions. In this paper, we present the Western Grasslands dynamic object-oriented Bayesian decision network. The Grasslands model is now deployed and being used by DEPI to: make predictions about changes in the grassland ecosystem; act as a repository of knowledge, to be updated as understanding of the grassland ecosystem improves; quantitatively evaluate the ecological and financial consequences of management actions; and rank management options with the highest probability of success for trialing.

This paper is published as abstract only.