Online Causal Structure Learning

Erich Kummerfeld and David Danks

December 9, 2010

Technical Report No. CMU-PHIL-189

Online Causal Structure Learning

Erich Kummerfeld[1] and David Danks[1,2]
[1]Carnegie Mellon University and [2]Institute for Human & Machine Cognition
Pittsburgh, PA 15213
{ekummerf, ddanks}@andrew.cmu.edu

December 9, 2010

Abstract

Causal structure learning algorithms have focused on learning in "batch-mode": i.e., when a full dataset is presented. In many domains, however, it is important to learn in an online fashion from sequential or ordered data, whether because of memory storage constraints or because of potential changes in the underlying causal structure over the course of learning. In this paper, we present *TDSL*, a novel causal structure learning algorithm that processes data sequentially. This algorithm can track changes in the generating causal structure or parameters, and requires significantly less memory in realistic settings. We show by simulation that the algorithm performs comparably to batch-mode learning when the causal structure is stationary, and significantly better in non-stationary environments.

1 Introduction

Over the past twenty years, a wide array of causal structure learning algorithms have been developed that apply under a range of assumptions, and that have been successfully applied in many different domains [1–3]. All of these algorithms share a crucial feature: they all process the data in a batch, rather than sequentially. This limitation prevents these algorithms from being used in two, potentially common, types of situations. First, they cannot be (feasibly) applied in low-memory contexts. Since the standard algorithms take the full dataset as input, they cannot be used by computational devices that are incapable of storing all of the data. For example, a sensor in a network often will not have the memory capacity to store all of the incoming information, particularly if it receives data along multiple channels. Second, these algorithms are not usable when the underlying causal structure or parameters change during the course of data collection (at least, not in a straightforward manner[1]). The standard algorithms assume all data are i.i.d. (or can be transformed into i.i.d. data); a dataset in which the causal structure changes partway through is clearly not i.i.d. At the same time, a range

[1]If one suspects that a change occurred during data collection, then one could apply a changepoint detection algorithm before doing any causal learning. This preprocessing step would have to be done by the researcher, though; there would be no signal from the causal learning algorithm that such a step was required.

of online learning algorithms have been developed to handle both of these types of situations, but none of them are capable of doing *causal* learning. They have principally focused on tracking the state of a system or variable over time, rather than learning a causal structure (that potentially varies over time) that generates sets of (locally) i.i.d. datapoints.

In this paper, we propose and assess the ***T****emporal-****D****ifference* ***S****tructure* ***L****earning* algorithm (TDSL) — the first (to our knowledge) online causal structure learning algorithm. The algorithm integrates several sophisticated tracking procedures with dynamic learning of causal (graphical) structure. In the next section, we quickly survey some related methods that provide some of the component pieces for our algorithm, though they are each insufficient for the task of online causal structure learning. We then explain the TDSL algorithm in some detail, and present simulation evidence that it can successfully learn causal structure in an online manner. In particular, TDSL automatically detects when the underlying causal structure or parameters have changed, and it tracks the causal structure in an efficient manner. Its memory requirements are also fixed, and do not change as a function of the amount of input data (in contrast to batch-mode algorithms that must store all data). At the same time, when there is a stable causal structure generating the datapoints, the performance of the TDSL algorithm is indistinguishable from a standard batch-mode causal structure learning algorithm. That is, there is no cost to using TDSL in ”normal” causal learning situations. We close with a discussion of future directions for the TDSL algorithm.

2 Related Work

2.1 Causal Learning Algorithms

Essentially all current causal structure learning algorithms output causal Bayesian networks or causal Structural Equation Models (SEMs). For simplicity, we will use “causal models” throughout to refer to both types of structures. These learning algorithms divide roughly into two distinct types: Bayesian/score-based procedures and constraint-based ones. Bayesian learning algorithms aim to find the causal model M that maximizes $P(M|Data)$. In practice, one typically has uniform priors over the possible causal models, and one assumes that the parameters are independent (in a precise technical sense), and so each causal model can be scored using a decomposable measure based on $P(Data|M)$ and the number of parameters in M [3,4]. Because the number of possible causal models is super-exponential in the number of variables, however, it is typically impossible to perform an exhaustive search of all possible causal models. Instead, one uses a greedy procedure that starts with a seed graph, scores nearby neighbors, moves to the highest-scoring neighbor, and iterates until no higher-scoring causal model can be found. If the greedy search is done properly, then it is asymptotically reliable [3].

Outside of the causal learning context, Bayesian learning algorithms—or various approximations to them—are frequently used for online learning because case-by-case Bayesian updating yields the same output as batch-mode processing (assuming the data are i.i.d.). In the cases of interest here, however, the underlying causal structure can change, and so we do not necessarily *want* to have the same output as a batch-mode algorithm. Moreover, even if we knew the structure would be stable over the course of learning, greedy search algorithms are usable for online learning only if the score for a graph can be computed using the current score of a neighbor plus the most recent datapoint. If this cannot be done, then there is no principled way to determine the appropriate score for a particular causal model when it is first considered. The measures currently used in score-based causal model search algorithms do not have this property, and so are not suitable for online causal learning.

A different approach to learning causal model structure is to leverage the fact that every causal model—more precisely, every causal graph—predicts a pattern of (conditional) independencies over the variables, though multiple causal models can predict the same pattern. Constraint-based algorithms (e.g., [1, 2]) determine a minimal set of (conditional) independencies in the data, and then find the set of causal models that best predict those (conditional) independencies. All existing constraint-based algorithms use traditional null hypothesis statistical tests to determine independencies from data, but there is no requirement that they do so. Properly speaking, constraint-based algorithms require only that some method be available that can provide the (conditional) independencies, which could be null hypothesis statistical tests, or Bayesian statistical tests, or even tracking algorithms that learn (conditional) independencies in real-time.

2.2 Realtime Tracking Algorithms

There are two primary methods for online tracking of some feature in an environment: temporal-difference learning (TDL) and change-point detection (CPD). Crucially, neither method has been applied to tracking causal structure, and both require substantial modifications to be suitable for it.

The classic TDL algorithm, TD(0) [5], provides a dynamic estimate $E_t(X)$ of a univariate random variable X using a simple update rule: $E_{t+1}(X) \leftarrow E_t(X) + \alpha(X_t - E_t(X))$, where X_t is the value of X at time t. That is, one updates the estimate by α times the error in the current estimate. The static α parameter encodes the learning rate, and must be chosen quite carefully (or somehow learned from the data). If α is too small, then $E_t(x)$ will converge too slowly; if α is too large, then $E_t(x)$ will be overly sensitive to noise of various types, even when the environment is stable. This latter problem is a particular concern for causal structure learning, since causal structures frequently have indeterministic causal connections. In general, TDL methods are good at tracking slow-moving changes in the environment, but perform suboptimally during times of either high stability or dramatic change.

Both Bayesian [6] and frequentist [7] online CPD algorithms are effective at detecting abrupt changes in the environment that indicate breaks between periods of stability. To do so, however, these algorithms must store substantial portions of the input data; the output of a Bayesian changepoint detector [6] for example is the probability of a changepoint having occurred r timesteps ago, and so the algorithm must store more than r datapoints. Furthermore, CPD algorithms assume a model of the environment that has only abrupt changes separated by periods of stability. Environments that evolve slowly but continuously will have their time-series discretized in seemingly arbitrary fashion, or not at all. As a result, these changepoint detection algorithms are not well-suited to online learning of causal structure.

3 Temporal Difference Structure Learning (TDSL) Algorithm

Given a set of continuous variables V, we assume that there is, at each moment in time, a true underlying causal model over V. A causal model will be specified by a pair $\langle G, \mathbf{F} \rangle$, where G denotes a directed acyclic graph over V, and $\mathbf{F}$ is a set of linear equations of the form $V_i = \sum_{V_j \in parents(V_i)} a_j \cdot V_j + \epsilon_i$, where $parents(V_i)$ denotes the variables $V_j \in G$ such that $V_j \rightarrow V_i$, and the ϵ_i are normally distributed noise/error terms. Such causal models are also known as recursive causal Structural Equation Models (SEMs). We assume that the data are, at any point in time, generated in an i.i.d. fashion from the true underlying causal model, though we do not assume that this causal model is stationary through time.

At a high level, the TDSL algorithm is separated into three, functionally distinct components.

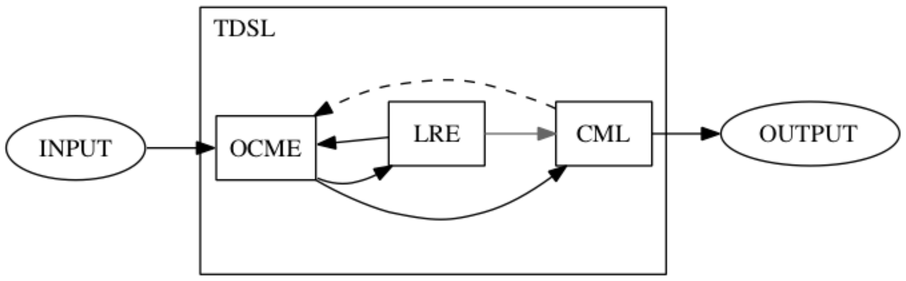


Figure 1: Basic TDSL architecture

The *O*nline *C*ovariance *M*atrix *E*stimator (OCME) receives each datapoint sequentially as input, and estimates a (possibly non-stationary) covariance matrix to provide the "raw materials" for learning the causal structure. The *L*earning *R*ate *E*stimator (LRE) tracks the divergence between recent datapoints and the estimated covariance matrix to detect changes in the environment, or significant errors in estimation. The *C*ausal *M*odel *L*earner (CML) takes the covariance matrix and learns the causal model at that point in time. The gray arrow between LRE and CML represents the information flow required for the probabilistic relearning scheduler described in Section 3.3 below. The dashed arrow from CML to OCME indicates the possibility of using the current estimated causal model to dynamically (and intelligently) influence the covariance matrix estimation. That feature is currently not part of the TDSL algorithm, but will be added in future research.

3.1 Online Covariation Matrix Estimation (OCME)

The OCME module performs the online updating of the sufficient statistics for causal learning from observational data. In particular, OCME maintains an estimated covariance matrix **C** over the variables V, and updates **C** in response to incoming datapoints. Let $N = |V|$ and M be the total number of datapoints observed. Because OCME does not store any of the incoming datapoints, its memory requirements are only $O(N^2)$ for the estimated covariance matrix, in contrast with $O(NM + N^2)$ memory for batch-mode algorithms. OCME thus has a substantial memory advantage for the common real-world situation of $M >> N$. OCME can also function as a stand-alone, single-pass covariance matrix estimator for very large datasets.

For the OCME algorithm, let $\mathbf{X}^r$ be the r-th multivariate datapoint and let X_i^r be the value of V_i for that datapoint. Because we do not assume a stationary causal model, the datapoints must potentially be weighted differently (e.g., weighting more recent datapoints more heavily after a change occurs). Let a_r be the weight on the r-th datapoint, and let $b_r = \sum_{k=1}^{r} a_k$ be the sum of the weights on each datapoint. We denote the weighted average of V_i after datapoint r by $\mu_i^r = \sum_{k=1}^{r} \frac{a_k}{b_r} X_i^k$. These means can be computed in an online fashion using the update equation:

$$\mu_i^{r+1} = \frac{b_r}{b_{r+1}} \mu_i^r + \frac{a_{r+1}}{b_{r+1}} X_i^{r+1}$$

The (weighted) covariance between V_i and V_j after datapoint r can then be proven to equal $\mathbf{C}_{V_i,V_j}^r = \sum_{k=1}^{r} \frac{a_k}{b_r}(X_i^r - \mu_i^r)(X_j^r - \mu_j^r)$. Because OCME is an online estimation method, however, we need to translate this into an update rule. The resulting update equation in terms of the current datapoint and the previous **C** can be derived to be:

$$\mathbf{C}_{X_i,X_j}^{r+1} = \frac{1}{b_{r+1}}[b_r \mathbf{C}_{X_i,X_j}^r + b_r \delta_i \delta_j + a_{r+1}(X_i^{r+1} - \mu_i^{r+1})(X_j^{r+1} - \mu_j^{r+1})]$$

where $\delta_i = \mu_i^{r+1} - \mu_i^r = \frac{a_{r+1}}{b_{r+1}}(X_i^{r+1} - \mu_i^r)$. If $a_r = c$ for all r and some constant positive real-valued c, then the estimated covariance matrix is identical to the batch-mode estimated covariance matrix. If $a_r = \alpha b_r$, then OCME tracks each covariance as though one is using TD(0) learning for the covariance with a learning rate of α.

3.2 Learning Rate Estimator (LRE)

The LRE module tracks the "fit" between the current estimated covariance matrix and the input data to determine the covariance matrix learning rates, represented in OCME as datapoint weights (i.e., the a_r's). Specifically, the "fit" between each incoming datapoint $\mathbf{X}^r$ and the current estimated covariance matrix $\mathbf{C}^r$ is given by the Mahalanobis distance [8]:

$$D_M = (\mathbf{X}^r - \vec{\mu})(\mathbf{C}^r)^{-1}(\mathbf{X}^r - \vec{\mu})^T$$

where $\vec{\mu}$ represents the current estimate of the means. A large Mahalanobis distance for any particular datapoint could simply indicate an outlier; consistently large Mahalanobis distances over multiple datapoints imply that the current estimated covariance matrix is a poor fit to the underlying causal model, and so new datapoints should be weighted more heavily. The Mahalanobis distance is a univariate value, and so we track it with a variant on TD(0) that we call Momentum-based Temporal-Difference Learning (MTDL)[2], which uses the same update rule as TD(0) but uses a varying α_r parameter:

$$E_{r+1}(D_M) \leftarrow E_r(D_M) + \alpha_r(D_M - E_r(D_M))$$

To set the α_r value, we use TD(0) to track the mean error of the $E_r(D_M)$ estimates; this provides "momentum" to our learning because a single large (or small) error after many small (or large) errors will have relatively little effect on α_r. More precisely, let $M_{r+1} \leftarrow M_r + \beta(D_M - E_r(D_M))$; note that β is *not* a function of r. We then pass M_r through a gamma distribution cdf in order to scale it to the $[0, 1]$ interval; specifically, $\alpha_r = \mathit{gamcdf}(|M_r|, k, \theta)$ for given k, θ. Even for a random variable with high variance, the momentum will be close to 0 if $E(D_M) \approx \mu(D_M)$, since the positive and negative errors will cancel each other out. By using strictly positive k and θ, these small momentum values will then result in α_r very near to 0, making this tracking algorithm extremely stable. At the same time, systematic errors of the same sign will cause M to quickly become large, resulting in a large α_r, and enabling MTDL to converge rapidly to new $\mu(D_M)$.

In the context of the LRE, if the Mahalanobis distance is relatively stable, then MTDL will have small errors and so α_r will be small. If the Mahalanobis suddenly grows for multiple datapoints (e.g., if the estimated covariance matrix suddenly fits the data poorly), then α_r will become quite large. We thus use these α_r values to set the a_r weights on the incoming datapoints. For $a_r > a_{r-1}$, old data will be partially forgotten since the incoming datapoint is being given greater weight. For technical reasons, causal learning can fail if $a_r < a_{r-1}$. We thus set: $a_{r+1} = Max(a_r, \alpha_r b_r)$.

3.3 Causal Model Learner (CML)

The CML module actually learns the causal model. Any causal model search algorithm requires the sample size in addition to the estimated covariance matrix. For the TDSL algorithm, different datapoints can receive different weights and so the number of datapoints is not necessarily the correct "effective sample size," though that should never be greater than the *actual* sample size. Let S_r be the sample size at time r. We assume the incoming datapoint contributes 1 to the sample size, and adjust the *previous* sample size accordingly. Specifically, we update S_r with the following rule: $S_{r+1} = \frac{a_r}{a_{r+1}} S_r + 1$. Since LRE ensures that $a_{r+1} \geq a_r$ for all r, we have that $S_{r+1} \leq S_r + 1$. Also, if $a_{r+1} = a_r$ for all r, then $S_r = r$; that is, if the datapoint weights are constant, then S_r is the true sample size.

CML uses a standard constraint-based causal structure learning algorithm, the PC algorithm [2], to learn the graphical structure based on the estimated covariance matrix (from OCME) and the effective sample size. Alternative constraint-based structure learning algorithms could also be used. Because learning causal models is computationally expensive [9], one does not want to relearn the graphical structure after each datapoint. Instead, one should use information from the LRE module to determine

[2]We define MTDL in terms of D_M here, but it works with any univariate random variable.

when one "ought" to perform these searches by balancing the accuracy of the current learned causal model against the computational cost of relearning the causal model. Recall that α_r denotes the current learning rate for the MTDL algorithm that is tracking the Mahalanobis distance, and so will be small when the estimated covariance matrix is judged to be approximately correct. Let γ_α denote the sum of the absolute changes in the α_r values since the *last* graph relearning. We then use a probabilistic scheduler in which

$$P(\text{relearning after datapoint } r) = \mathit{gamcdf}(\gamma_\alpha, k_2, \theta_2)$$

Immediately after a change occurs in the underlying causal structure, the Mahalanobis distance will be largest, and so the α_r values will quickly grow. Observing additional datapoints from the new structure will allow for better predictions, and α_r will quickly become small again. Both types of movement will inflate γ_α substantially. As a result, graph relearning should be most common after an underlying change, though there should be a non-zero chance of relearning even when the causal structure is relatively stable. This performance of the probabilistic scheduler is confirmed in the third set of simulations below.

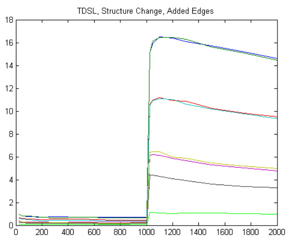
4 Simulation Results

We used synthetic data to evaluate the performance of TDSL given known ground truth. All simulations used scenarios in which either the ground truth parameters or ground truth graph (and parameters) changed during the course of data collection. Before the first change occurs, there should be no significant difference between TDSL and a standard batch-mode learner, since those datapoints are completely i.i.d. Performance on these datapoints thus provides information about the performance cost (if any) of online learning using TDSL, relative to traditional algorithms. After a changepoint, one is interested both in the absolute performance of TDSL (i.e., can it track the changes?) and in its performance relative to a standard batch-mode algorithm (i.e., what performance *gain* does it provide?). We used the PC algorithm [2] as our baseline batch-mode learning algorithm; for the graph and sample sizes in our simulations, any standard causal model learning algorithms would perform similarly.

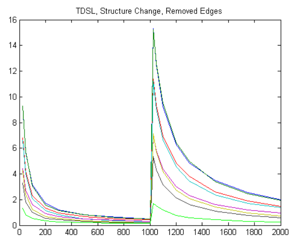
In order to directly compare the performance of TDSL and PC, the first two sets of simulations imposed a fixed "graph relearning" schedule on TDSL[3]; the probabilistic relearning schedule is explored in the third set of simulations. In the first set of simulations, we used datasets with 2000 datapoints, where the causal SEM graph and parameters both changed after the first 1000 datapoints. We generated 500 datasets for each $\langle \#variables, MaxDegree\rangle$ of $\langle 4, 3\rangle$, $\langle 8, 3\rangle$, $\langle 10, 3\rangle$, $\langle 10, 7\rangle$, $\langle 15, 4\rangle$, $\langle 15, 9\rangle$, $\langle 20, 5\rangle$, and $\langle 20, 12\rangle$, where each dataset had two different, randomly generated causal SEMs of the specified size and degree.

Figures 2(a-c) show the mean edge addition, removal, and orientation errors (respectively) by TDSL as a function of time, and Figures 2(d-f) show the mean of

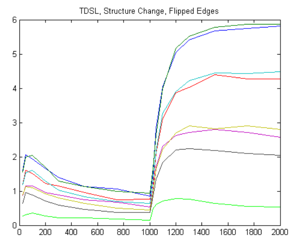
[3]TDSL relearned graphs and PC was rerun after datapoints $\{25, 50, 100, 200, 300, 500, 750, 1000, 1025,$ $1050, 1100, 1200, 1300, 1500, 1750, 2000\}$.



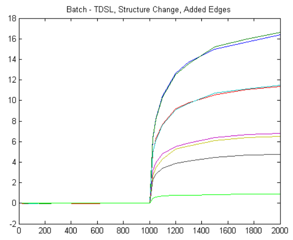
(a)



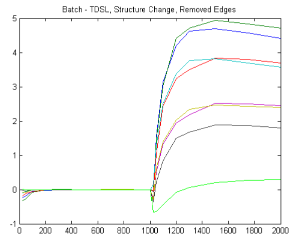
(b)



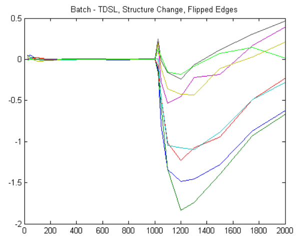
(c)



(d)



(e)

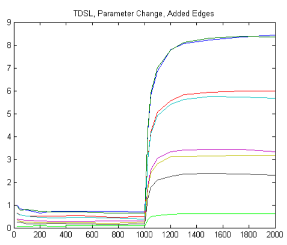


(f)

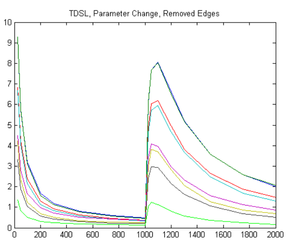
Figure 2: (a-c) TDSL errors and (d-f) TDSL improvement over batch when structure changes

$\#errors_{PC} - \#errors_{TDSL}$ for each error type (i.e., higher numbers imply TDSL outperforms PC). In all graphs, each $\langle variable, degree \rangle$ pair is a distinct line. As expected, TDSL was basically indistinguishable from PC for the first 1000 datapoints; the lines for those datapoints in Figures 2(d-f) are all essentially zero. After the underlying causal SEM changes, however, there are significant differences. The PC algorithm performs quite poorly because the full dataset is essentially a mixture from two different distributions which induces a large number of spurious associations. In contrast, the TDSL algorithm finds large Mahalanobis distances for those datapoints, which lead to higher weights, which lead it to learn (approximately) the new underlying causal model. In practice, TDSL typically stabilized on a new causal model by roughly 250 datapoints after the changepoint.

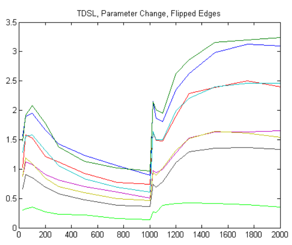
The second set of simulations was identical to the first (500 runs each for various pairs of variable number and edge degree), except that the graph was held constant throughout and only the causal SEM parameters changed after 1000 datapoints. Figures 3(a-c) and 3(d-f) report, for these simulations, the same measures as Figures 2(a-c) and 2(d-f). Again, TDSL and PC performed basically identically for the first 1000 datapoints. Performance after the parameter change did not follow the same pattern as before, however. TDSL again does much better on edge addition and orientation errors, but performed worse on the edge removal errors for the first 400 points following the change. The reason is that TDSL weights the new data very heavily immediately after the changepoint, which reduces the effective sample size. As a result, TDSL initially finds more independencies than are correct. Over the course of the 1000 datapoints,



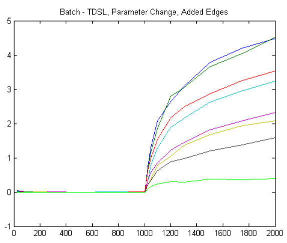
(a)



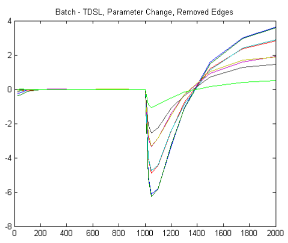
(b)



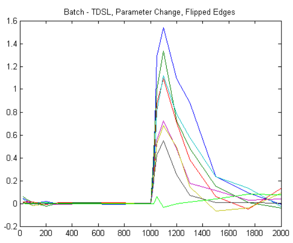
(c)



(d)



(e)



(f)

Figure 3: (a-c) TDSL errors and (d-f) TDSL improvement over batch when parameters change

however, TDSL converges towards the correct new parameters, and so outperforms PC on all measures.

The third set of simulations was designed to explore in detail the performance of the probabilistic relearning scheduler. We randomly generated a single dataset with 10,000 datapoints, where the underlying causal SEM graph and parameters changed after every 1000 datapoints. Each causal SEM had 12 variables and maximum degree of 4. We then ran TDSL with the probabilistic relearning schedule 500 times on this dataset. Figure 4(a) shows the (observed) expected number of "relearnings" in each 50-datapoint window. As expected, there are substantial relearning peaks after each structure shift, and the expected number of relearnings persisted at roughly 0.4 per 50 datapoints throughout the 1000-datapoint stable period. Figures 4(b-d) provide error information: the smooth green lines indicate the mean edge addition, removal, and orientation errors (respectively) during learning, and the blocky blue lines indicate the TDSL errors if graph relearning occurred after every datapoint (i.e., optimal TDSL performance). Although there are many fewer graph relearnings with the probabilistic schedule, overall errors did not significantly increase.

5 Discussion and Future Research

While the TDSL algorithm does not assume that the data are globally i.i.d., it clearly will work only for data that are "locally" i.i.d. If the underlying causal structure changes very rapidly during data collection, or if the datapoints are a random-order

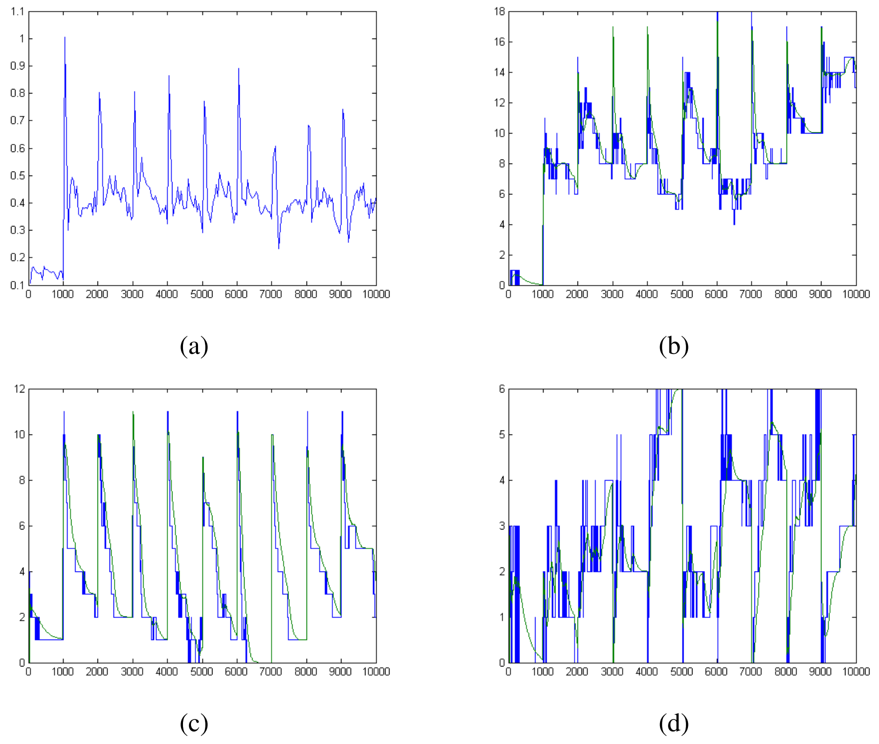


Figure 4: (a) TDSL expected relearnings for 50-point window; (b-d) errors and optimal performance

mixture from multiple causal structures, then the TDSL algorithm will continually have high Mahalanobis distance in the LRE module, and so high learning rates in the OCME module. As a result, the overall algorithm is unlikely to stabilize for any length of time on a particular causal structure. An important future research direction is to determine how to have TDSL efficiently track smoothly or rapidly varying causal structures.

The TDSL algorithm also allows theoretically for the possibility that the current learned causal model can be used to influence the a_r weights. If there are certain causal connections that have not changed over a long period of time, or have been stable over multiple relearnings, then one might plausibly conclude that those connections are stable and less likely to change. Thus, much greater error should be required to substantially change the estimates for those connections. In practice, implementing this intuition requires allowing for the a_r weights to vary across $\langle V_i, V_j \rangle$ pairs. The mathematics of the OCME become much more complicated when this is allowed, and much more information must be tracked. It is currently unknown whether the (presumably) improved tracking would compensate for the additional computational and memory cost.

We have focused on causal models represented by SEMs, but there is also a long history of successful causal modeling using causal Bayes nets defined over discrete variables with conditional probability tables. Tracking the sufficient statistics for causal Bayes net structure learning is substantially more costly, and we are currently investigating ways to learn the necessary information in a tractable, online fashion. Similarly, we have focused on constraint-based structure learning since the relevant scores in score-based methods (such as [3]) do not decompose in a manner that is suitable for online learning. We are thus investigating alternative scores, as well as heuristic approximations to principled score-based search.

There are many real-world contexts in which batch-mode causal structure learning is either infeasible or inappropriate. The online causal structure learning algorithm that we presented has great potential to perform well in a range of challenging contexts, and at little cost in "traditional" settings.

References

[1] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[2] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000.

[3] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[4] D. Heckerman, C. Meek, and G. Cooper. A bayesian approach to causal discovery. In C. Glymour and G. Cooper, editors, *Computation, Causation, and Discovery*, pages 141–165. MIT Press, 1999.

[5] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[6] R. P. Adams and D. J. C. MacKay. Bayesian online changepoint detection. Technical report, University of Cambridge, Cambridge, UK, 2007. arXiv:0710.3742v1 [stat.ML].

[7] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 8:2961–2974, 2005.

[8] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2:49–55, 1936.

[9] D. M. Chickering. Learning Bayesian networks is NP-complete. In *Proceedings of AI and Statistics*, 1995.