Learning Discrete Graphical Models using L1/L2 Regularized Logistic Regression

vishvAs vAsuki

Abstract

This report examines the discrete graphical model structure learning algorithm proposed in a paper by Ravikumar et al.[6]. The purpose of this project is to check if the algorithm works, and gain intuition into the conditions under which it works. This report details early experiments in this direction which show promising results - the algorithm succeeds in correctly determining the structure of some star structured graphical models. Besides this, the report also discusses an efficient way to select a suitable regularization parameter for use by the structure learning algorithm during future experiments. Future work will involve theoretical analysis and more detailed experiments to determine conditions under which the algorithm is guaranteed to recover the graph structure.

Contents

1 Introduction

In what follows, we will first describe the concept of undirected graphical models, and the problem of learning the structure of a graphical model. We will then describe Ising models and general discrete graphical models. We will then describe the neighborhood learning algorithm. In doing this, we will establish the notation we will use throughout this report.

1.1 Undirected graphical models and structure learning

Consider the probability distribution D associated with the p-dimensional random vector X, and let $V = \{1..p\}$. Suppose that D could be factorized as $D(X = x) = \prod_{c \subseteq V} \phi_c(x_c)$. This factorization can be represented using a graph $G = (V, E)$ with p nodes V and some edges E, such that any set of variables, c, occuring in the factorization of D corresponds to cliques in G. G, together with D, is called an Undirected Graphical Model, or a Markov Random Field.

1.1.1 Pairwise graphical models.

Suppose further that every set of variables, c, occuring in the factorization $D(x) = \prod_{c \subseteq V} \phi_c(x_c)$ contains at most two variables. The corresponding graphical model is then called a pairwise graphical model. This allows us the following interpretation of G: The variables X_i and X_j are independent according to D exactly when G contains an edge between nodes i and j.

1.1.2 The structure learning problem.

The task we consider in this project is one of learning G, given n samples drawn from an unknown pairwise graphical model. This is a very general and important task in science and engineering, where one is often interested in learning the dependencies between various random variables observed. For example, given gene co-expression data, one may be interested in learning the functional dependencies between genes.

1.1.3 Discrete graphical models.

In particular, we are concerned with discrete graphical models, where the range of every random variable X_i is a finite set. Pairwise discrete graphical models are very general: One can always find a pairwise discrete graphical model equivalent to any given Discrete graphical model: one just

needs to introduce new variables X_c corresponding to sets c of than more than two variables variables. So, in this project we restrict ourselves to considering only pairwise discrete graphical models.

The probability distribution D associated with a pairwise discrete graphical model can be completely specified by

$$Pr(x|\theta) = \prod_{(i,j)\in E} \phi_{i,j}(x_i, x_j) \propto \exp(\sum_{(i,j)\in E} \theta_{i,j,x_i,x_j}), \tag{1}$$

where $\phi_{i,j}(x_i, x_j)$ is being specified as being proportional to $\exp(\theta_{i,j,x_i,x_j})$. If one were to further define $\forall (i,j) \notin E : \theta_{i,j,x_i,x_j} = 0$, we could write: $Pr(x|\theta) \propto \exp(\sum_{(i,j)} \theta_{i,j,x_i,x_j})$.

So, D can be completely described by the parameter array θ. Note that, for each vertex pair (i,j), there are $|range(X_i)| \times |range(X_j)|$ parameters. Also, if the graph G is sparse, θ is group sparse.

For desigining a structure learning algorithm, we will need an expression for the conditional probability:

$$Pr(X_i = x_i | X_{/i} = x_{/i}, \theta) \propto \exp(\sum_{j\in\Gamma(i)} \theta_{i,j,x_i,x_j}), \tag{2}$$

where $\Gamma(i)$ denotes the neihborhood of i in G and $x_{/i}$ is shorthand for $x_{V-\{i\}}$.

1.2 Ising models

Ising models are pairwise undirected graphical models associated with the random vector $X \in \{-1, 1\}^p$, whose probability distribution D can be specified by $Pr(x|\theta) \propto \exp(\sum_{(i,j)\in E} \theta_{i,j} x_i x_j)$. For this special case of pairwise undirected graphical models, there is just one parameter $\theta_{i,j}$ associated with any edge $(i,j) \in E$.

We are interested in this class of graphical models, because algorithms to solve the structure learning problem for such models are the main object of study in Ravikumar et al.[6]. The conditional probability $Pr(X_i = x_i | X_{/i} = x_{/i}, \theta) \propto \exp(\sum_{j\in\Gamma(i)} \theta_{i,j} x_i x_j)$ will be used for this purpose.

2 Structure learning algorithm

In this section, we will first consider the structure learning algorithms presented in Ravikumar et al.[6], which are based on learning the neighborhood of each node in G. We will then contrast this algorithm with some other structure learning algorithms.

2.1 Neighborhood learning algorithm

One can learn the structure G of a pairwise graphical model with the distribution D by determining the neighborhood $\Gamma(i)$ of each $i \in V$. One way of determining $\Gamma(i)$ is to estimate the parameters[1] $\theta_{i,:,:,:}$, and deduce that $j \in \Gamma(i)$ if and only if $\exists x_i, x_j : \theta_{i,j,x_i,x_j} \neq 0$.

[1] MATLAB notation is being used here.

So, the only thing which remains to be specified is the algorithm used to learn these parameters $\theta_{i,:,:,:}$, given i and a set of n samples drawn from D. It turns out that logistic regression, with suitable regularization, can be used for this purpose.

2.1.1 For general discrete graphical models.

A good parameter estimate $\theta^*_{i,:,:,:}$ would maximize the likelihood (or equivalently, minimize the negative log likelihood) of $\theta_{i,:,:,:}$ given that a set of n observations $S = \left\{x^{(i)}\right\}$ is drawn from the probability distribution D. However, as noted in Section 1.1.3, a good estimate $\theta^*_{i,:,:,:}$ would also be group sparse. One poplular way of imposing group sparsity is to use ℓ_1/ℓ_2 regularization while solving the optimization problem. So, the algorithm to find $\theta^*_{i,:,:,:}$ is one which solves the convex optimization problem

$$\arg\min_{\theta_{i,:,:,:}} \left\{ nll_i(\theta_{i,:,:,:}|S) + \lambda' \sum_j \|\theta_{i,j,:,:}\|_2 \right\}$$

Here, $nll_i(\theta|S)$ denotes the negative log likelihood of $\theta_{i,:,:,:}$, given the observations S. Equivalently, in our experiments, we solve the following optimization problem:

$$\arg\min_{\theta_{i,:,:,:}} \left\{ n^{-1} \sum_{k=1}^{n} nll_i(\theta_{i,:,:,:}|x^{(k)}) + \lambda \sum_j \|\theta_{i,j,:,:}\|_2 \right\}, \tag{3}$$

where $nll_i(\theta_{i,:,:,:}|x^{(k)})$ denotes the negative log likelihood of θ given a sample $x^{(k)}$; and, from Equation 2, it can be seen to be:

$$nll_i(\theta_{i,:,:,:}|x^{(k)}) = -\sum_{j \in V} \theta_{i,j,x_i^{(k)},x_j^{(k)}} + \log[\sum_l \exp(\sum_{j \in V} \theta_{i,j,l,x_j^{(k)}})]$$

In some of our experiments, we solve the above optimization problem by rewriting it as a constrained optimization problem:

$$\arg\min_{\theta_{i,:,:,:}:\sum_j \|\theta_{i,j,:,:}\|_2 \leq c} \left\{ n^{-1} \sum_{k=1}^{n} nll_i(\theta_{i,:,:,:}|x^{(k)}) \right\}. \tag{4}$$

Whether we solve the problem using formulation 3 or using formulation 4, the solution will depend on the choice of λ or c respectively. We refer to these problem parameters as *regularization parameters*, to distinguish them from the *model parameters* θ.

2.1.2 For Ising models

As noted in Section 1.2, in this case, we just have one parameter $\theta_{i,j}$ corresponding to each pair of random variables, (i, j). As earlier, we want to find parameters $\theta_{i,:}$ which are most likely given the set of n observations $S = \left\{x^{(i)}\right\}$, but subject to prior belief that $\theta_{i,:}$ is sparse. Here, the following optimization problem is solved:

$$\arg\min_{\theta_{i,:}} \left\{ n^{-1} \sum_{k=1}^{n} nll_i(\theta_{i,:}|x^{(k)}) + \lambda \sum_j \|\theta_{i,j}\|_2 \right\}, \tag{5}$$

where $nll_i(\theta_{i,:}|x^{(k)})$ denotes the negative log likelihood of $\theta_{i,:}$ given a sample $x^{(k)}$. This is described and analyzed in detail in Ravikumar et al.[6].

2.1.3 Resolving inconsistencies

From the sparsity pattern of the parameters $\theta_{i,:,:,:,:}$ learned using logistic regression, the algorithm deduces the neighbors of each node in G. The algorithm can encounter the following inconsistency when deducing graph structure: The neighborhood set $\Gamma(u)$ learned for node u, may include v, but the neighborhood set $\Gamma(v)$ learned for v may not include u.

For the Ising model case, this is usually not a problem, as the analysis by Ravikumar et al.[6] guarantees consistent signed edge recovery when certain conditions are met. In the case of general discrete graphical models, during our early experimetns, we resolve inconsistencies using the following rule: $[v \in \Gamma(u)] \vee [u \in \Gamma(v)] \implies (u, v) \in G$.

2.2 Related work

We now briefly mention of a couple of other approaches to graphical model structure learning. As the project is at an early stage, this survey of related work is not meant to be complete, but is meant to give a flavor of the kind of structure learning algorithms used in the past. There are two major ways of tractably learning structures of graphical models: either the topology is assumed, or one learns the neighborhood of each node, one node at a time. As an example of the first category of algorithms, early work by Chow and Liu[1] tried to learn tree structured graphical models. The neighborhood learning algorithms are better in the sense that it works for cases where G has cycles. In Dahinden et al.[2], a decompositional approach has been proposed for learning graphical models, in which after initially estimating the graphical model using node-wise regression, then the algorithm decomposes the graph into subgraphs and end up estimating and combining structures of small sub-graphs. This raises the question of whether a similar decompositional approach can be used with the structure learning algorithm studied in this project, but that is deferred for future research.

3 Experiment setup

3.1 Test distributions.

In order to test structure learning algorithms, we design distributions modelled by pairwise graphical models. In order to specify such a distribution completely, we need to specify the graph G (that is, the topology), the size of the range of each random variable X_i and the parameters associated with each edge, as described in Equation 1. We take the size of the range of each random variable X_i to be three. Also, for our early experiments, we tried the following topologies:

- Star graph: All nodes are connected to one node.
- Chain: The nodes form a chain.

Then, for each edge (i, j), we select the edge-potentials $e_{\theta_{i,j,:,:}}$ uniformly from the range $[0, k]$. Note that the distribution specified by these parameters is invariant to translation. For some of our experiments, we pick $k = 3$, and for other experiments, we set $k = 1$. The choice of k is important because, as in the case of the Ising model structure learning algorithm, we expect that the structure learning algorithm for general discrete graphical models will only work when the array θ satisfies certain conditions. One suggestion Ravikumar [5] has been to select $k \propto \frac{1}{\sqrt{p}}$ or $k \propto \frac{1}{\sqrt{d}}$, where p denotes the number of variables, and d denotes the maximum degree of any node in G.

3.2 Drawing samples.

Having specified the distribution D corresponding to the graphical model, we will need to efficiently draw samples from it for the purpose of testing the structure learning algorithm. For our early exeriments, we are using tree structured graphical models; so we are able to efficiently and accurately sample from D. To do this, we use software provided by Schmidt [8]. In future experiments, where we may deal with cases where G has loops, we can use MCMC for efficient approximate sampling.

3.3 Goals and Evaluation

One of the goals of our experiments is to understand how many examples are needed for the algorithm to learn the structure of a pairwise discrete graphical model with p nodes with high probability. Another of the goals of our experiments is to understand the dependency between the number of samples n, the number p of nodes in G, and the parameter λ used by the neighborhood learning algorithm.

In order to address the first goal, we draw multiple sample sets of increasing size ($n \in [2^5, 2^{14}]$) from the distribution D and do the following for each choice of n: We fix a suitable λ_n for use in solving the problem specified in Equation 3 either manually or using validation, and finally we empirically determine the probability of success of the structure learning algorithm. In order to address the second goal, we plot the λ_n used during the previous process either against n or against $\sqrt{\frac{\log p}{n}}$.

We follow the same procedure for the case where we tackle the logistic regression problem in its constrained optimization form (Equation 4), except that we now need to pick the parameter c, rather than λ.

3.4 Choosing the regularization parameter

As described in Section 2.1, in solving the logistic regression problem, the choice of the regularization parameter, which is c or λ depending on the formulation used, affects the quality of the solution we get. In the following discussion, even though we consider the problem of picking λ used in Formulation 3, similar observations apply to picking c used in Formulation 4.

If we pick a λ which is too small, we end up allowing θ to be dense, and the graph we learn as a consequence will have too many edges; and we may get a complete graph! If we pick a λ which is too big, we end up getting parameters θ which are too sparse - even all zeros, and the graph we learn as a consequence will have too few - perhaps zero - edges. So, we must pick a λ which lies in between these two extremes. The choice of λ has been a vexing problem in this project, one for which we do not yet have a completely satisfying solution.

Obozinski et al.[4] analyze a problem which is identical to Equation 3 except that the negative log likelihood part of the objective is replaced with $\|Y - X\theta\|_F$. They use $\lambda = \sqrt{\frac{f(p)\log p}{n}}$, for any $f(p) \to \infty$ as $p \to \infty$. It is likely that a similar choice for λ may work in our case too.

3.4.1 The validation procedure

One can manually pick λ - by picking the one which yields a model of expected sparsity, and we do this in some of our experiments in order to prove the point that the structure learning algorithm actually works, given suitable λ. Otherwise, one can use validation to programmatically pick λ. The validation process involves assigning scores to λ based on how good it is. The goodness of a given regularization parameter λ depends on the goodness of the model parameters θ learned using that regularization parameter.

This raises the question: How good is a given model parameter array θ? The goodness of the model parameters θ can either be judged based on its sparsity, or based on its likelihood given some observations S.

We will consider the latter first. As determining the likelihood of θ, which is $\prod_{x \in S} \frac{exp(\sum_{i,j} \theta_{i,j,x_i,x_j})}{Z(\theta)}$, requires the computation of the partition function $Z(\theta) = \sum_x exp(\sum_{i,j} \theta_{i,j,x_i,x_j})$, we find it better to instead compute the pseudolikelihood of the parameter $\prod_{x \in S} \prod_i Pr(x_i | x_{/i}, \theta)$, where $Pr(x_i | x_{/i}, \theta)$ is the conditional probability specified in Equation 2.

We can also rate model parameters θ based on its ability to yield the desired sparsity. For some of our experiments, we use this method - not only does it seem to be more reliable, it is also faster because there is no need to evaluate the pseudo-likelihood of θ given some observations S: only its sparsity properties need to be abalysed. We use binary search to find the ideal λ using this goodness measure: thus, we take advantage of the fact that the sparsity of the resultant θ varies monotonically with λ. However, using this method during structure learning tasks in practice will require prior information about the sparsity of the graphical model being learned. Hopefully, analysis will help us identify the parameters even without knowing beforehand the sparsity structure of the graphical model generating the observations.

Irrespective of the mechanism of choosing the best λ, in our experiments, for the purpose of robustness to the randomness involved in empirically evaluating a certain value of λ, we do *k-fold validation*. So, we evaluate the goodness of any given λ based on the goodness of parameter arrays θ learned from k different sample sets.

3.5 Solving the optimization problem efficiently

Now, we consider the problem of solving the ℓ_1/ℓ_2 regularized logistic regression problem efficiently. This is especially important when dealing with problems which occur in practice, where p and n tend to be huge. When we choose to specify the problem using Formulation 4, we use a Projected Quasi-Newton algorithm proposed by Schmidt et al.[7]. In our experiments, we use software provided by the authors[8].

When we choose to include the ℓ_1/ℓ_2 constraint in the objective, as specified in Formulation 3, we solve the problem using a simple adaptation of the block coordinate gradient descent algorithm proposed by Meier et al.[3]. In Meier et al.[3], the authors aim to solve ℓ_1/ℓ_2 regularized logistic regression for the case where the response variable is binary; whereas in our case the response variable is not necessarily binary. Solving the logistic regression problem in this form is important because it will allow us to examine the relationship between λ, p and n; and prior analysis for related optimization problems has been done using such a formulation. In our experiments, we use our own MATLAB implementation of this algorithm.[2]

4 Results and discussion

In Figure 1, we present results of early experiments where Formulation 4 was used for nieghborhood identification. Besides the probability of success, the regularization parameter, c, which was used in these experiments is also presented; however, this parameter was not chosen in any systematic manner for these expermients. These form a sort of *proof of concept* for the structure learning algorithm.

In Figure 2, we show the results of some experiments using star structured graphs. In one of these experiments, the sparsity-based fast validation method (which utilized binary search) described in Section 3.4.1 was used to learn the λ to be used in learning the neighborhood structure of nodes given n examples. This experiment demonstrates the efficacy of this validation procedure in selecting a suitable λ in case we have prior knowledge of the sparsity of the target graphical model. (However, other experiments seem to indicate that the k we use in k-fold validation must be larger than 5.) From the λ values learned by this validation procedure, we observe the dependence of the suitable range of λ values on p and n. In particular, rough calculations using linear interpolation indicate that $\lambda = 0.075(\frac{\log p}{n})^{1/2} + 0.1495$ will be a good choice for future experiments.

We also conducted a few experiments on chain structured graphical models, whose parameters were drawn uniformly at random from $[0, 1]$. Unfortunately, the structure learning algorithm did not succeed in this case. Future analysis will probably reveal the reason behind this failure.

[2]Note that we cannot use the R code provided by the authors of Meier et al.[3], as they assume that the response variable is binary.

5 Conclusion

In this project, we examined the discrete graphical model structure learning algorithm proposed in a paper by Ravikumar et al. [6]. The purpose of this project was to check if the algorithm works, and gain intuition into the conditions under which it works. This report described early experiments in this direction, and the results were promising - the algorithm succeeds in correctly determining the structure of some star structured graphical models from random samples. Besides this, we also proposed and demonstrated an efficient way to select a suitable regularization parameter for use by the structure learning algorithm during future experiments.

5.1 Future work

Future work will involve theoretical analysis and more detailed experiments to determine conditions under which the algorithm is guaranteed to recover the graph structure; and this will involve further experimentation on higher dimensional graphical models. This will require more efficient implimentation of the structure learning algorithm than what was used for this project. Another direction for future work is to examine the efficacy of the structure learning algorithm by applying it to a fairly large problem of practical interest.

6 Acknowledgements

The author thanks Prof. Pradeep Kumar Ravikumar for giving him an opprotunity to work on this problem. He also thanks Shruthi Viswanath for proof-reading this report, and Ali Jalali for his close scrutiny of his implementation of the block coordinate descent algorithm from Meier et al. [3]: this led to the resolution of some bugs in the code.

References

[1] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1054142`.

[2] Corinne Dahinden, Markus Kalisch, and Peter Bühlmann. Decomposition and model selection for large contingency tables. *Biometrical Journal*, 52(2):233–252, March 2010. ISSN 1521-4036. doi: 10.1002/bimj.200900083. URL `http://dx.doi.org/10.1002/bimj.200900083`.

[3] Lukas Meier, Sara van de Geer, and Peter Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, February 2008. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2007.00627.x. URL `http://dx.doi.org/10.1111/j.1467-9868.2007.00627.x`.

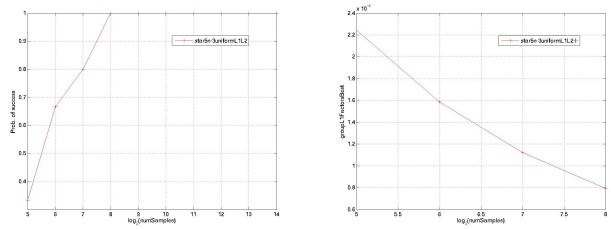
[4] Guillaume Obozinski, Martin J. Wainwright, and Michael I. Jordan. Union support recovery in high-dimensional multivariate regression. Aug 2008. URL http://arxiv.org/abs/0808.0711.

[5] Pradeep Ravikumar. Private communication, 2010.

[6] Pradeep Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Annals of Statistics*, 2009.

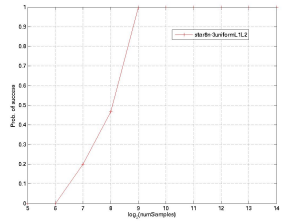
[7] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. 2009.

[8] Mark Schmidt. Graphical models software, June 2010. URL http://www.cs.ubc.ca/~schmidtm/. http://www.cs.ubc.ca/ schmidtm/.

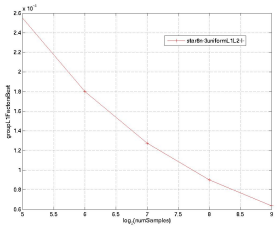


(a) 5 node star.

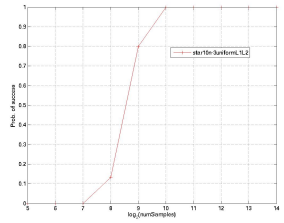
(b) 5 node star.



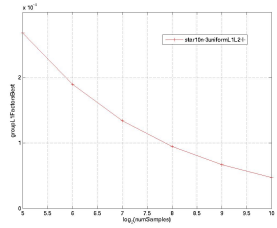
(c) 8 node star.



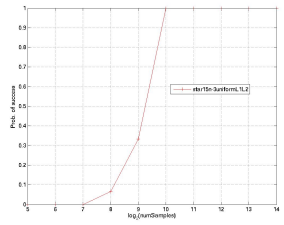
(d) 8 node star.



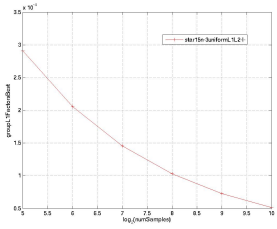
(e) 10 node star.



(f) 10 node star.

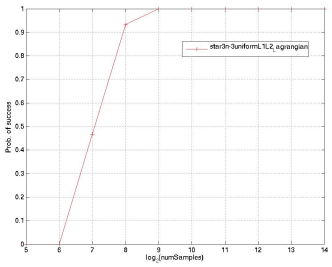


(g) 15 node star.

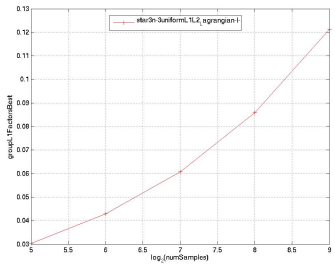


(h) 15 node star.

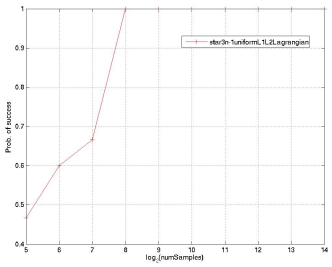
Figure 1: Experiments on star structured graphical models, using Formulation 4 for neighborhood identification. Besides the probability of success, the regularization parameter, c, which was used in these experiments is also presented. Experiment setup is described in 3. In all cases, the parameters constituting θ were selected uniformly at random from $[0, 3]$. These early experiments provided us the evidence that the structure learning algorithm actually works in certain cases, and paved the way for more detailed experimetns using Formulation 3.



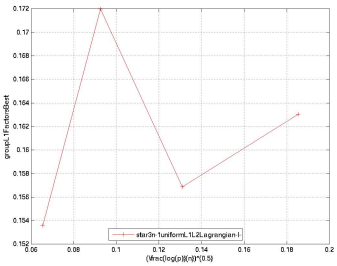
(a) 3 node star, $k = 3$: Structure learned using manually selected λ.



(b) 3 node star, $k = 3$: λ selected manually.



(c) 3 node star: Structure learned using programatically selected λ.



(d) 3 node star. λ learned using binary search based on sparsity.

Figure 2: Experiments on star structured graphical models, using Formulation 3 for nieghborhood identification. In one set of experiments, Parameters were chosen Besides the probability of success, the regularization parameter, c, which was used in these experiments is also presented. Experiment setup is described in 3. Samples were drawn from a 3 node star whose parameters θ were selected uniformly at random from $[0, k]$. For one experiment, $k = 3$ was used, for all other experiments, $k = 1$ was used. One of these experiments demonstrate a solution to problem of selecting an appropriate λ using validation. For details, please see Section 4.