Learning Graphical Model Structure using L1-Regularization Paths

Mark Schmidt[*] and **Alexandru Niculescu-Mizil**[+] and **Kevin Murphy**[*]
Department of Computer Science
[*]University of British Columbia/[+]Cornell University

Abstract

Sparsity-promoting L1-regularization has recently been succesfully used to learn the structure of undirected graphical models. In this paper, we apply this technique to learn the structure of directed graphical models. Specifically, we make three contributions. First, we show how the decomposability of the MDL score, plus the ability to quickly compute entire regularization paths, allows us to efficiently pick the optimal regularization parameter on a per-node basis. Second, we show how to use L1 variable selection to select the Markov blanket, before a DAG search stage. Finally, we show how L1 variable selection can be used inside of an order search algorithm. The effectiveness of these L1-based approaches are compared to current state of the art methods on 10 datasets.

Introduction

Learning the structure of graphical models from data is useful for a variety of tasks, ranging from density estimation to scientific discovery. Unfortunately, it is an NP-hard problem (Chi96). Consequently, many heuristic techniques have been proposed. In this paper, we propose a new heuristic technique and we show experimentally that it outperforms several existing approaches on a variety of datasets.

A simple but effective approach to learning graph structure is to learn a dependency network (HCM$^+$00). This is a series of independently learned conditional distributions of the form $p(x_j|x_{-j})$, where x_{-j} represents all variables except x_j. Typically these local conditional probability distributions (CPDs) are represented as classification/regression trees, so that each node can select just a subset of neighbors.

If the data is jointly Gaussian, one can represent the CPDs using linear regression. By imposing a sparsity-promoting L1 regularization penalty on the regression coefficients (Tib96), one can efficiently select the neighbors using algorithms such as LARS (EJHT04). Under certain assumptions, Meinshausen and Buhlmann (MB06) proved that this method correctly recovers the undirected network structure in the large sample limit.

If the data is binary, one can replace linear regression with logistic regression. One advantage of this over tabular representations of CPDs is that the number of parameters is linear in the number of parents, rather than exponential. Recently, Wainwright et al. (WRL06) extended the Meinshausen and Buhlmann consistency proof to the case of logistic regression.

The above L1 techniques learn the topology of the undirected graph, but do not learn its parameters; this makes the resulting models unsuitable for density estimation and prediction. In the case of *Gaussian* Markov networks, one can jointly estimate the structure and parameters by imposing an L1 prior on each element of the precision matrix, and then finding the MAP estimate using convex optimization (see e.g., (BGdN06)). In the case of *binary* Markov networks, one can in principle adopt the same approach, although it is in general NP-hard to estimate the parameters of discrete Markov nets, requiring approximations such as loopy belief propagation (LGK07).

One drawback of the above techniques is that they assume that each node has the same amount of L1 regularization; this constant is often set by cross validation. However, in many networks (e.g., gene regulatory networks), some nodes will have high degree (connectivity) and others low degree. Our first contribution is to propose an efficient method for selecting the optimal regularization constant on a per node basis by using the regularization path.

Another drawback of the above techniques is that they learn undirected graphical models. As we mentioned, parameter estimation in such models is NP-hard in general (except for the Gaussian case). Also, undirected models cannot be used to model causality (in the sense of (Pea00)), which is useful in many domains such as molecular biology where interventions can be performed. As far as we know, L1 techniques have not been used to learn DAG structure. Several authors (e.g., (LY05)) learn sparse Gaussian DAGs using L1 techniques, given an ordering, but they do not address the issue of learning the correct variable ordering, and instead convert the result back to an undirected graph.

Our second contribution is to show how L1 techniques can be used to improve two existing DAG learning methods. The first DAG learning method that we improve is called max-min hill-climbing (MMHC) (TBA06) which was shown to outperform many other methods, in a series of extensive experiments. MMHC first identifies a set of potential neighbors (parents and children) for each node using conditional independency tests, where the conditioning sets are grown in a greedy way; this is called the max-min parents-children (MMPC) algorithm. The output of MMPC is then used to prune out potential neighbors before performing standard hill-climbing through this restricted DAG space. In this paper, we show that using L1 regression to find a set of potential neighbors (a technique we call L1MB, for ‘L1-regularized Markov Blanket’) results in much lower false negative rates (with a comparable rate of pruning). In addition, our method is more statistically efficient, since it does not need to perform conditional independency tests on exponentially large conditioning sets.

The second DAG learning method that we improve is called order search (OS) (TK05), which was also shown to outperform some existing methods, such as DAG search with sparse candidate pruning. (The OS and MMHC papers did not compare against each other; we provide such a comparison below.) OS performs heuristic search through the space of topological orderings, rather than through the space of DAGs (c.f., (LP96; FK03)). The space of orderings is much smaller than the space of DAGs: $O(d!)$ orders vs $O(d!2^{\binom{d}{2}})$ DAGs (Rob73), where d is the number of nodes. Furthermore, given an ordering, the problem of structure learning reduces to d independent variable selection problems (each node can choose its parents from its predecessors in the ordering independently). The approach taken in the OS paper was to use exhaustive search through all possible parent sets for each node. We show that one can use L1 regression to efficiently approximate this.

Methods

We assume that we have fully observed (complete) data, which is either continuous or binary (we discuss the extension to discrete data with more than two categories at the end), and that our goal is to find the DAG G that maximizes the BIC score (HGC95), or equivalently minimizes the MDL (minimum description length) cost, defined as

$$MDL(G) = \sum_{j=1}^{d} NLL(j, \pi_j, \hat{\theta}_j^{mle}) + \frac{|\hat{\theta}_j^{mle}|}{2} \log n$$

$$NLL(j, \pi_j, \theta) = -\sum_{i=1}^{n} \log p(X_{ij} | X_{i,\pi_j}, \theta)$$

where n is the number of data cases, π_j are the parents of node j in G, $NLL(j, \pi_j, \theta)$ is the negative log-likelihood of node j with parents π_j and parameters θ, and $\hat{\theta}_j^{mle} = \arg\min_\theta NLL(j, \pi_j, \theta)$ is the maximum likelihood estimate of j's parameters. (We use the MDL objective since there is no natural conjugate prior for logistic regression, so computing the integral needed for the Bayesian score is hard.) For linear regression, $p(X_j|\pi_j, \theta_j) = \mathcal{N}(X_j|\theta_j^T \pi_j, 1)$ (the data is standardized so $\sigma^2 = 1$ for each node). For logistic regression, $p(X_j|\pi_j, \theta_j) = f(X_j \theta_j^T \pi_j)$, where $X_j \in \{-1, +1\}$ and $f(z) = 1/(1 + e^{-z})$ is the sigmoid function. The term $|\hat{\theta}_j|$ is the number of free parameters in the CPD for node j. For linear regression, $|\hat{\theta}_j| = |\pi_j|$, the number of parents; in the case of logistic regression, we also include a bias (offset) term θ_0.

We can find the MLEs of linear regression CPDs using least squares, and of logistic regression using the iteratively reweighted least squares (IRLS) algorithm. We call the operation of estimating the parameters of each CPD a "family fit". This takes $O(I(np^2 + p^3))$ time, where $p = |\pi_j|$ is the size of the parent set, and I is the number of IRLS iterations. Typically $I < 10$, so we treat this as a constant. Also, we assume $n \gg p$, so the overall cost is $O(np^2)$. When comparing running times between different algorithms below, we will treat a family fit as a primitive unit of computation. If we were to use tabular CPDs, a family fit would take $O(n)$ time but $O(2^p)$ space, assuming binary nodes. For this reason, it is standard to impose a fan-in constraint, $p \leq k$. However, since we use linear/ logistic regression CPDs, we do not need this restriction, so k is set to d.

L1 penalized regression

We now consider the problem of choosing the set of neighbors/parents for node j from some set U. This can be done by solving the following:

$$\hat{\theta}_j^{L1}(U) = \arg\min_\theta NLL(j, U, \theta) + \lambda ||\theta||_1$$

where λ is the scale of the penalty on the L1 norm of the parameter vector (excluding θ_0). In the Gaussian case, this can be formulated as a quadratic objective subject to linear constraints; in the binary case, this is a non-quadratic objective, but is still convex. The effect of these constraints is to drive many of the parameters strictly to zero (Tib96). This lets us use the technique as a variable selection strategy. In particular, we set $\pi_j \subseteq U$ to be the indices of the non-zero elements of $\hat{\theta}_j^{L1}$. We will call this technique "L1 variable selection".

A variety of different techniques have been proposed to solve this convex constrained optimization problem. For the Gaussian case, the method of choice is LARS (least angle regression and shrinkage) (EJHT04). For the binary case, we re-formulate the optimization as a bound-constrained problem, and solve it using an efficient Two-Metric Projection strategy (Ber99). We used L-BFGS to scale the gradient, and found this to be more efficient than the 'IRLS-LARS' algorithm of (LLAN06).

Choosing λ

An important issue is choosing the strength of the regularizer, λ. A simple approach is to use cross-validation, but this will be slow if we want to use a different λ for every node. However, we do not want to use the same λ for all nodes for two reasons. First, in order search techniques, nodes late in the topological ordering have more parents to choose from, and may therefore need a stronger regularizer. Second, some nodes in the true structure may have much higher connectivity than others. We have done experiments where we force all nodes to have the same λ, and performance is worse (results not shown due to lack of space).

We propose to only consider values of λ at discontinuities along the regularization path; these points can be computed efficiently (see discussion below), and the MDL-optimal value must lie at one of these points. To see why, first note that the negative log likelihood is monotonically decreasing in λ. Second, note that the number of free parameters is piecewise constant, and increases only occur at values of λ where new parameters are introduced (call this set Λ). Hence the optimal MDL score must be achievable by a point $\lambda \in \Lambda$. See Figure 1 for an illustration (this argument also applies to the Akaike Information Criterion).

In the case of linear regression, the regularization path is piecewise linear, and it is possible to compute the optimal $\hat{\theta}_j^{L1}$ for all possible parameters λ in $O(p^3 + np^2)$, where p is the number of parameters (EJHT04); this is the same asymptotic cost as a single least squares fit. In the case of generalized linear models (GLMs), the regularization path is no longer piecewise-linear. However, predictor-corrector strategies can be used for following such curved regularization paths (see (PH06)). In this paper, we adopt the simpler scheme of simply searching along a grid of values for λ. The procedure begins with $\lambda_{max} = \max |\nabla NLL(j, U, [\theta_0^{mle}; \vec{\mathbf{0}}])|$ (ie. the maximum gradient magnitude when all non-bias covariates are inactive and the bias variable is at its optimal value), since all greater values of λ will have all non-bias variables inactive. We then take p evenly-spaced decreasing steps along the λ-axis (until λ is 0). A slightly more sophisticated strategy could backtrack if we "overshoot" a discontinuity.

Since it suffices to look at $O(p)$ values of λ, the total cost to compute the whole path for a node is $O(np^3)$ per node in the GLM case, and $O(np^2)$ in the Gaussian case. We also use a "warm-start" strategy, which consists of initializing the gradient-projection algorithm at the previous value's solution for the decreasing values of λ. We found that this reduced the number of descent iterations over a "cold-start" strategy (typically we need less than 5 iterations per λ).

There is one subtlety we have overlooked. When we compute the regularization path, we compute $\hat{\theta}_j^{L1}(\lambda)$, but the MDL score uses the unconstrained parameters $\hat{\theta}_j^{mle}$ (for the non-zero set of variables). However, we can modify LARS to compute these estimates for the same asymptotic cost (in the linear regression case) by re-using the Cholesky matrix factorization (for the GLM case, we estimate $\hat{\theta}_j^{mle}$ whenever the non-zero set of variables changes).

In summary, L1 variable selection consists of 3 components: (i) computing the L1-regularization path, (ii) computing the Maximum Likelihood parameters for all non-zero sets of variables encountered along this path, and (iii) selecting the set of variables that achieved the highest MDL score.

L1-Pruned DAG-Search

We can now explain our technique for identifying the undirected skeleton, which we call L1MB (L1-regularized Markov Blanket). In this approach, we regress each node X_j on all others ($U = x_{-j}$), using L1 variable selection. This process takes $O(nd^3)$ time per node (for the GLM case), and (ideally) finds a set that is as small as possible, but that contains all of j's parents, children and co-parents (its Markov blanket). This preprocessing yields a small set of potential parents that are then used to restrict the edges that are considered when hill-climbing in the space of DAGs (using the standard moves of addition, deletion, and reversals). Since the dependencies may be asymmetric in the finite sample scenario (i.e., the edge $a - b$ may be found when regressing on a but not on b), we use the conservative 'OR' strategy of (MB06), where an arc is only excluded if it was not found in either direction (since the cost of excluding a true edge at this stage is higher than including a spurious edge)[1].

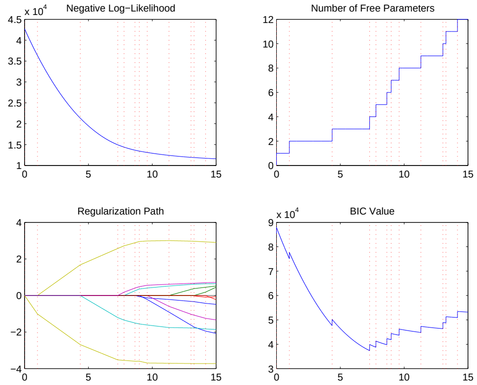
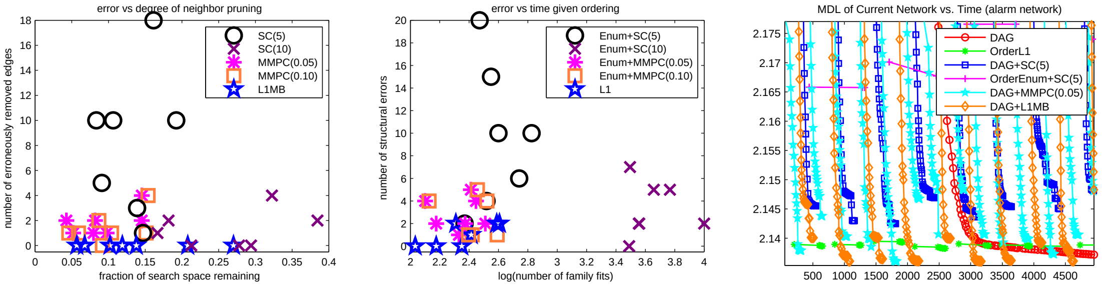


Figure 1: Regularization path for a Gaussian negative log-likelihood with an L1-regularizer. The horizontal axis is the bound on the L1 norm of the coefficient vector, i.e., at location t we have $||\theta||_1 \leq t$ (where $t \propto 1/\lambda$)). Top left: negative log likelihood. Bottom left: the parameter values. Top right: number of free parameters. Bottom right: MDL cost. We see that the minimum MDL must occur at one of the discontinuities along the regularization path (red lines).

Note that the goal of this pre-processing is not the same as when learning a dependency network, since we use L1 to tell us which edges to leave out, rather than which edges to include. The final decision about which edges to include is deferred until the DAG search stage, since some edges (e.g., between co-parents) may be artefacts of explaining away, and can be eliminated once the edge orientation is known.

The L1MB preprocessing step is analogous to the first iteration of the Sparse Candidate (SC) algorithm (FNP99), and the Max-Min Parents Children (MMPC) algorithm (TBA06). The SC algorithm requires the user to specify a bound, c, on the number of possible parents to choose from; it then selects the c most "relevant" nodes for each x_j using an association measure (in the full SC algorithm, the set of relevant nodes is then re-estimated after each round of structure learning). The main problem with the SC algorithm is that it requires a uniform bound on the number of parents, which is problematic for the same reasons as using a uniform value of λ. In contrast, for each x_j the MMPC algorithm attempts to prune nodes y where, $\exists z.y \perp x_j | z$ (ie. y is independent of x_j given z). The conditioning sets z are chosen in a greedy way (to maximize the minimum degree of association between y and x_j). The conditional independency test used is G^2 (similar to χ^2), using a significance level cutoff α on the p-value. Note that this creates conditioning sets of size exponential in the number of neighbors. Also, if one were to do a Bonferroni correction to account for the $O(d^2)$ multiple tests, the statistical power would be very low. L1MB suffers from neither of these problems.

[1] We also experimented with the 'AND' strategy (requiring edges to be found in both directions), finding that it prunes more edges but is more likely to prune correct edges



(a) (b) (c)

Figure 2: (a) Neighbor pruning experiments on 10k binary samples. We plot number of edges removed that were in the true graph vs size of remaining search space, defined as $1 - r/e$, where r is the number of edges removed, and $e = d^2 - d$ is the total number of possible edges (each point is one of 7 BNR data sets). An ideal pruning strategy would be in the bottom left. We see that L1MB (blue pentagons) prunes substantial portions of the space and did not prune any true edges. (b) Parent selection given a topological ordering on 10k binary samples. We plot the number of wrong edges (parents) in the learned structure vs the log-number of family fits for several methods (each point is one of 7 BNR data sets). We show enumeration with 3 different pruning strategies and a max fan-in of 5, and L1 variable selection without pruning or restriction. (c) The MDL score of the current network being evaluated vs the total number of family fits. We see that OrderL1 quickly achieves a good score, but that DAG+MMPC and DAG+L1MB eventually achieve better scores.

L1-Pruned Order-Search

Order search is an alternative to searching through DAG space. Following (TK05), we use hill climbing to find the best ordering, where the moves considered are adjacent swaps in the ordering ('twiddles'):

$$(X_{i_1}, \ldots, X_{i_j}, X_{i_{j+1}}, \ldots) \rightarrow (X_{i_1}, \ldots, X_{i_{j+1}}, X_{i_j}, \ldots)$$

At each step, we consider all $d - 1$ successors of the current ordering and greedily pick the best. This can be implemented such that after each twiddle only 4 new family fits are needed (TK05), which makes it efficient to move through order space (other moves are obviously possible, but would be more expensive to evaluate).

Given an ordering, (TK05) find the best set of parents for each node using the SC algorithm followed by exhaustive search, which takes $O(c^k)$ family fits (assuming a fan-in bound of k). Our approach is to replace the initial pruning and exhaustive search with L1 variable selection, which takes $O(d)$ family fits. In addition to speed, our method can handle much higher in-degrees (we do not need a bound c or k), since we use linear/logistic regression instead of tabular CPDs.

Experimental results

Our experimental protocol was to acquire 7 networks from the Bayes net repository (BNR)[2] with 27–61 nodes each, and a maximum fan in of 5.[3] We then parameterized the networks using linear regression or logistic regression CPDs, with random parameters.[4] We chose the random parameters such that there tended to be non-negligible correlation between parents and children. In particular, we sampled the regression weights from $\pm 1 + \mathcal{N}(0,1)/4$; the offset (bias term) was zero in both cases. We then sampled data from these networks and attempted to recover the original structure. To be fair, all search methods methods used linear/logistic CPDs (except the MMPC pruning step where we used "causal explorer"[5]). In addition to the above synthetic data, we considered 3 real publicly-available binary data sets: (i) a subset of the 20-newsgroups data (indicating the presence/absence of 100 words in 16,242 documents) [6], (ii) the Anonymous Microsoft Web Data Database (indicating whether 294 websites were visited by 32,711 users, we restricted to the 57 websites with greater than 250 visits)[7], and (iii) binarized handwritten digits of the number '0' from the MNIST database (containing 6903 images that we resized to be 16 by 16, and focused on predicting the 131 pixels than were non-empty in more than 200 images) [8]. On these real data sets, we cannot assess structural error, but we can measure the MDL and test-set negative log likelihood achieved by the different methods.

We performed a large number of experiments, but here we only have space to summarize a few of the key findings. We focus on binary observational datasets of size $10,000$ for the synthetic data experiments, and $5,000$ for the real data experiments. Details and results for other scenarios (Gaussian, interventional, different sample sizes, etc.), the raw experimental results, experiments on different variations of the pruning/searching strategy, and source code to ensure repro-

[2] `http://www.cs.huji.ac.il/labs/compbio/Repository`

[3] Specifically, we used the following 7 networks (number of nodes/ edges in brackets): 1. insurance (27/52), 2. water (32/66), 3. mildew (35/46), 4. alarm (37/46), 5. barley (48/84), 6. hailfinder (56/66) 7. carpo (61/74)

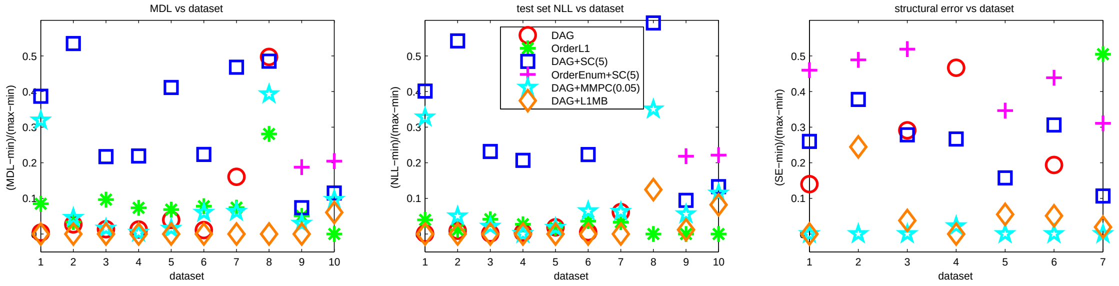
[4] We did not use the "standard" parameters that come with some of these models, as they assume tabular CPDs and non-binary nodes.

[5] `http://discover1.mc.vanderbilt.edu/discover/public/causal\_explorer`

[6] `http://www.cs.toronto.edu/~roweis/data.html`

[7] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[8] `http://yann.lecun.com/exdb/mnist/`



(a) (b) (c)

Figure 3: (a) MDL on training set. (b) Negative log likelihood on test set. (c) Structural error. All quantities are scaled so the best is 0 and the worst is 1, i.e., we plot $(x_i - \min_i(x_i))/(\max_i(x_i) - \min_i(x_i))$, where x_i is the quantity for method i. We have truncated the vertical scale at 0.6 for clarity, so if a method is missing, its relative score is worse than 0.6. All methods have 10k family fits on 10k binary training samples (5k for data sets 8-10). Test sets are 100k binary samples (except data sets 8-10, where the test set is the remaining examples not used in training). Datasets: 1. insurance, 2. water, 3. mildew, 4. alarm, 5. barley, 6. hailfinder, 7. carpo, 8. msweb, 9. news, 10. mnist.

ducible research are available on the author's webpage[9].

Neighbor pruning

In Figure 2(a), we compare L1MB, MMPC and SC as methods for estimating the neighbors of each node on the BNR structures. We use two different candidate sizes of $c = 5$ and $c = 10$ for sparse candidate (SC) (with pairwise correlation as the association measure), and two different threshold levels $\alpha = 0.05$ and $\alpha = 0.1$ for MMPC. In contrast, L1MB has no free parameters. The goal is to maximize the number of edge removals (in order to constrain the search space) while minimizing the number of edges that are removed but which are in the true graph (false negatives). We see that many methods reduce the search space by a significant amount, but L1MB did so without introducing false negatives. A low false negative rate is important, since any edges falsely excluded by an edge-pruning strategy can never be added back.

Parent selection given an ordering

In Figure 2(b), we compare two different methods for choosing the set of parents given a correct node ordering on the BNR structures using either exhaustive enumeration or L1 variable selection. Since enumeration takes time exponential in the size of the set being searched, we first applied the neighbor-pruning strategies just discussed, and restricted the max fan-in to $k = 5$ (the max fan-in among the networks). We measure how long it takes to fit the model in terms of the number of family fits, and then we count the number of false positive and false negative edges. Since the ordering is known, it is possible to get zero structural error even using observational data (i.e. we do not need to worry about Markov equivalence). Figure 2(b) shows that L1 variable selection is more efficient than using SC pruning, and is as efficient while making fewer errors than enumeration with MMPC pruning. Note that we don't count the cost of SC or MMPC pruning for the enumeration methods, since this is a constant cost which will be amortized over many orderings.

Structure Search

Finally, we examine the performance of several search algorithms, in terms of MDL score and test set negative log-likelihood. For the BNR structures, we generated 100,000 examples to measure test set likelihood, while for the real data sets we used all data instances except the 5,000 used for training. We also measured structural error on the BNR structures, by converting the true DAG to a PDAG (since the DAG can only be recovered up to Markov Equivalence without interventions).

We compared our proposed strategies of running DAG-Search after L1MB pruning and running Order-Search with L1 variable selection to a standard unpruned DAG-Search and three state-of-the-art methods: DAG-Search after SC pruning (FNP99), DAG-Search after MMPC pruning (TBA06), and Order-Search after SC pruning (with exhaustive enumeration and where we set the fan-in bound to 5) (TK05).

Since the MDL structure score is not convex, both DAG-Search and Order-Search methods get stuck in local minima. Thus it is not very meaningful to plot the error vs running time to reach convergence, since the results would depend too much on the starting point. (We cannot start all algorithms from the same point, since some search in DAG space, and others in order space.) We therefore run each hill-climbing algorithm as many times as it can (restarting at local minima), up to a fixed time limit of 10,000 family fits (this value was chosen to give DAG search enough time to explore at least one local minimum on the largest BNR structure). Once again, we did not include the startup cost of the initial pruning in these plots, since this is a small constant relative to the overall cost of an extensive search. In Figure 2(c), we show a trace of the MDL cost vs number of family fits for different methods on the alarm network.

To compare results across datasets, we look at the final performance after 10,000 steps. In Figure 3(a), we plot the MDL cost as a function of the data set. We see that the DAG+L1MB method is consistently the best or very close to the best, except on the MNIST data. Due to its high number of nodes, on the MNIST data set only the DAG+MMPC and

[9] `http://www.cs.ubc.ca/~murphyk/L1structure`

OrderL1 methods had reached a local minimum (with the latter reaching a higher scoring local minima).

Although MDL is the objective function we are optimizing, often we care more about predictive performance. We can measure this by negative log likelihood (NLL) on a test set: see Figure 3(b). Here we see that good MDL scores correlate with good predictive performance on the BNR data, but that on the real data sets MDL was not always a good indicator of predictive performance. In terms of test set NLL, the OrderL1 tended to have the best performance overall[10], while DAG+L1MB tended to outperform, on average, the remainder of the methods.

It is also interesting to look at the number of PDAG structural errors, the number of incorrect edges in the estimated model compared to the true model. Again this is not the objective function that is being explicitly minimized, but it is often what one we are interested in (especially given interventional data, when the DAG structural error can be driven to zero). In Figure 3(c), we see that DAG+MMPC is the best by this measure. However, since MMPC makes more false negatives than L1MB (see Figure 2(a)), given more time we would expect L1MB pruned methods to outperform MMPC pruned methods.

Based on these search results, we observed that Order-Search methods quickly achieve a high score by finding the best DAG consistent with a random ordering. However, the Order-Search methods subsequently make little progress. In constrast, random DAGs generally have a very low score, but DAG-Search makes significant progress and typically reaches better local minima than Order-Search (see Figure 2(c)). We explored intializing DAG-Search by finding the best DAG consistent with a random ordering (combining the good properties of both approaches), and found that this is more efficient than both DAG-Search and Order-Search when the number of nodes is large (results omitted due to space constraints).

Conclusions and future work

We have shown that L1 regression is a useful technique for structure learning, whether for speeding up order search, or for pruning the set of possible edges. We have omitted some details due to lack of space. In our on-line report, we included extended experimental results and additional details on the different methods.

Although we have considered binary models that are linear in the parent values, in the future, we would like to extend these techniques to handle multi-state discrete variables as well as modeling parent interactions and nonlinear effects. All of these extensions will require group-L1 strategies (e.g., (YM06)) since there will no longer be a 1:1 correspondence between parameters and edges.

[10] In Figure 3(c), we see that OrderL1 does poorly in terms of structural error, even though it does well in terms of NLL on real data. We believe that OrderL1 is not able to prune edges very easily, since it cannot explore the space of graphs very well, but that this actually helps its performance on real data, where having more edges may help overcome the limited modeling power of using logistic CPDs.

References

[Ber99] D. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 2000.

[BGdN06] O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. In *Intl. Conf. on Machine Learning*, 2006.

[Chi96] D. Chickering. Learning Bayesian networks is NP-Complete. In *AI/Stats V*, 1996.

[EJHT04] B. Efron, I. Johnstone, T. Hastie, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[FK03] N. Friedman and D. Koller. Being Bayesian about Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 50:95–126, 2003.

[FNP99] N. Friedman, I. Nachman, and D. Peer. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *UAI*, 1999.

[HCM$^+$00] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for density estimation, collaborative filtering, and data visualization. *J. of Machine Learning Research*, 1:49–75, 2000.

[HGC95] D. Heckerman, D. Geiger, and M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 1995.

[LGK07] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *NIPS*, 2007.

[LLAN06] S. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient L1 Regularized Logistic Regression. In *AAAI*, 2006.

[LP96] P. Larrañaga and M. Poza. Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.

[LY05] Fan Li and Yiming Yang. Using modified lasso regression to learn large undirected graphs in a probabilistic framework. In *AAAI*, 2005.

[MB06] N. Meinshausen and P. Buhlmann. High dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34:1436–1462, 2006.

[Pea00] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge Univ. Press, 2000.

[PH06] M. Park and T. Hastie. L1 regularization path algorithm for generalized linear models. Technical report, Dept. Statistics, Stanford, 2006.

[Rob73] R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, 1973.

[TBA06] I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 2006. To appear.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B*, 58(1):267–288, 1996.

[TK05] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *UAI*, pages 584–590, 2005.

[WRL06] M. Wainwright, P. Ravikumar, and J. Lafferty. Inferring graphical model structure using ℓ_1-regularized pseudo-likelihood. In *NIPS*, 2006.

[YM06] M. Yuan and Y. Lin. Model Selection and Estimation in Regression with Grouped Variables. *J. Royal. Statist. Soc B*, 68(19):49-67, 2006.