LassoOrderSearch: Learning Directed Graphical Model Structure using L1-Penalized Regression and Order Search

Mark Schmidt
Computer Science Dept.
University of British Columbia
schmidtm@cs.ubc.ca

Kevin Murphy
Computer Science Dept.
University of British Columbia
murphyk@cs.ubc.ca

Abstract

To speed up the search for the best DAG structure given data, we propose to use the regularization paths from L1-penalized regression to rapidly find the best set of parents given a node ordering. This reduces the complexity of evaluating a node ordering from $O(Nd^3d^K)$ to $O(Nd^5)$ time ($O(Nd^4)$ in the linear-Gaussian case), where N is the number of data cases, d is the number of nodes, and K is the maximum number of parents (fan-in). Not only is this approach much faster than previous approaches, but our linear (instead of exponential) dependence on the number of parents allows us to tractably fit much more complex models. We provide experimental comparisons with several other heuristic search techniques illustrating the effectiveness of this approach.

1 Introduction

Directed graphical models are a useful tool for causal modeling (24; 29). One of the key challenges is to learn the structure of these models from data. In this paper, we view this as an optimization problem. Specifically, we want to find the DAG (directed acyclic graph) G that maximizes the BIC/MDL scoring function given some data D: $G^* = \arg\max_G \text{score}(G, D)$. The data may be purely observational, or a mix of observational and interventional (experimental); the latter is crucial for disambiguating between Markov equivalent structures, and hence learning causal structure (7).

The general problem of finding the best DAG is NP-hard (4). The most common approach is to use heuristic search (typically greedy hill climbing) through the space of DAGs (see e.g., (14; 22; 32)). However, the space of possible DAGs is enormous (there are $O(d!2^{\binom{d}{2}})$ DAGs on d nodes (26)), with many local minima. Exact methods for finding the best DAG do exist (16; 28; 27), but take $O(d2^d)$ time.

If we know the order of the nodes, finding the optimal graph structure is equivalent to d independent variable selection problems (3; 6). This is because, given an ordering, each node can choose its parents from the previous nodes in the ordering independently, without violating the global acyclicity constraint. Since the order is typically not known,Teyssier and Koller (hereafter, TK) (30), proposed to use heuristic search in the space of node orderings. This is much smaller than the space of all DAGs, and order-based search avoids all local minima except those that are consistent with an ordering. Consequently, the TK algorithm is considered one the best algorithms for learning DAG structure.

TK solved the variable selection problem, given an ordering, by exhaustive enumeration through all subsets of preceeding nodes, for sets up to size K, where $K \leq d$ is a restriction on the number of parents (fan-in) allowed for each node. This takes $O(\sum_{i=0}^{K} \binom{d}{i}) = O(d^K)$ time per node. In this paper, we propose to use L1-penalized linear/ logistic regression to perform the variable selection task in $O(d^4)$ time per node with no restriction on the fan-in. (In the linear regression case, the time reduces to $O(d^3)$.) Specifically, we penalize the regression weights in order to learn a sparse sparse set of parents. This is not only computationally much faster, but more statistically parsimonious, since regression at each node uses a number of parameters that is polynomial in the number of parents, whereas the tabular (multinomial) representation is exponential.

2 Related work

2.1 Generalized lasso as a fast variable selection method

For d regression coefficients represented by a vector θ, L1-penalized regression uses $\lambda \sum_{i=1}^{p} |\theta_i|$ as a regularizer of the log-likelihood (where λ controls the strength of the regularization). It is widely used for efficient feature selection in linear regression, under the name of LASSO (31). Although non-differentiable if any θ_i is 0, the L1-penalized log-likelihood is concave with a unique maximum, and efficient methods have been developed for L1-penalized linear regression, including the Least Angle Regression (LARS) strategy of (10). The LARS strategy allows the computation of the optimal penalized likelihood coefficients for all possible values of λ at a cost of $O(Nd^3)$ for N instances. The piecewise-linear trace of the coefficients as λ is varied is referred to as the 'regularization path'.

For discrete valued data, log-likelihoods for generalized linear models (GLMs), such as (multinomial) logistic regression, can replace linear regression. The optimization remains concave, but the regularization path is no longer linear. However, efficient predictor-corrector methods have recently been explored for following such curved regularization paths, including for Logistic Regression (see (23), for example).

2.2 L1 penalties for learning undirected graph structures

We are not aware of previous work using L1 penalties to learn DAG structure. However, there has been recent work on using L1 penalties to learn *undirected* graphs. In the case of Gaussian graphical models (GGMs), an arbitrary order (say $1 : d$), is used to learn one or more directed linear models, and the result is converted to an undirected GGM (15; 20; 21). The claim is that, since the final goal is to learn an undirected model, the order used to learn the directed graph does not matter. However, this is clearly not true in the finite sample regime. (9) acknowledge this problem, and address it by sampling orderings using a heuristic based on dependency networks; given each ordering, they create a directed GGM using forward variable selection, and then convert it to an undirected graph. Alternatively, one can put an L1 penality directly on the inverse covariance (precision) matrix; this results in a convex "maxdet" optimization problem (8; 34; 2).

Learning undirected graph structure on discrete nodes is harder, since computing the partition function Z takes time exponential in the treewidth. (19) use loopy belief propagation to approximate Z, and use an L1 prior on the weights to learn a sparsely connected Markov network on binary nodes. To avoid the need for computing Z, (33) uses L1 logistic regression of each node given all others (learning a dependency network (13; 21)). Although this may learn the correct undirected topology, it is not a true density model. Also, these undirected models can not be used for causal learning.

2.3 Searching in order space

The idea of searching through orders instead of DAG-space has been exploited in several papers (18; 11; 30). TK 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, TK 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 evaluations are needed to consider the next set of successors (30).

(18) used genetic algorithms instead of hill climbing, and (11) used MCMC. All of these works have considered discrete graphical models with tabular CPDs, making the variable selection problem exponentially expensive. By moving to GLMs with L1-regularization, we can use polynomial time algorithms, and work with polynomial sized representations.

3 Lasso Order Search

Our method builds on TK, who search over orderings $\prec$ and use exhaustive enumeration to find the best set of parents for each node consistent with that order. Using $U_{i,\prec} = \{U : U \prec i, |U| \leq K\}$ as the set of all parent sets preceeding i in the ordering that have size at most K, TK defines the score of an ordering $\prec$ in terms of the best possible score for each node:

$$\begin{aligned} \text{score}(\prec) &= \sum_{i=1}^{d} \text{score}(i, \pi_{i,\prec}) && (1) \\ \pi_{i,\prec} &= \arg\max_{U \in U_{i,\prec}} \text{score}(i, U) && (2) \end{aligned}$$

Common node scores include the BIC/MDL criteria and the Bayesian Score (14). We used the BIC score:

$$\begin{aligned} \text{score-BIC}(i, U) &= LL(i, U, \theta_{i,U}^{MLE}) - \frac{|\theta_{i,U}^{MLE}|}{2} \log N && (3) \\ \theta_{i,U}^{MLE} &= \arg\max_{\theta} LL(i, U, \theta) && (4) \\ LL(i, U, \theta) &= \sum_{n=1}^{N} \log p(X_{ni} | U_n, \theta) && (5) \end{aligned}$$

In Equation 3, $fp_i = |\hat{\theta}_{i,U}^{ML}|$ is the number of free parameters in the CPD for node i. For linear or logistic regression, $fp_i = O(|U|)$, the number of parents. For linear regression, $p(X_i|U, \theta_i) = \mathcal{N}(X_i|\theta_i^T U, \sigma^2)$ (each node is standardized so $\sigma^2 = 1$), and for logistic regression, $p(X_i|U, \theta_i) = \sigma(X_i \theta_i^T U)$, where $X_i \in \{-1, +1\}$ and $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function.

We refer to a 'family evaluation' as the the task of evaluating score(i, U), the score for node i based on using a subset of the potential parents U (where U is restricted by the ordering). The limitation with the TK approach is that it takes $O(d^K)$ family evaluations to compute Equation 2 for each node. Our proposal is to approximate the exhaustive search for the best U with sets generated from L1-penalized optimization.

The L1 regularization path provides a continuous range of parameter values. Typically, one would use an expensive cross-validation step from among a set of discrete values of λ. However, for GLMs we can select the λ optimizing the BIC score from a finite sized set of points along the regularization path. (In the linear regression case, this set has size $O(d)$. We conjecture this is also true in the GLM case.) To see why we can restrict attention to this finite set of λ values, first note that the negative log likelihood is monotonically decreasing with the L1 bound on the parameters $||\theta||_1 < t$ (where $t \propto 1/\lambda$). Second, note that the number of free parameters $fp_i(t)$ is piecewise constant, and increases only occur at locations Λ_i where new parameters are introduced. Hence the negative BIC score for N instances, $J(\theta) = -2LL(\theta) + fp_i(\theta) \log(N)$, only has discontinuities at Λ_i, so it suffices to search these points. See Figure 1 for an illustration.[1] Note that using the same λ value for all nodes results in much worse performance than letting each node choose its own optimal value. Intuitively this is because nodes later in the order need a higher λ penalty to avoid an overly large fan-in. See Figure 4 for a graph of the chosen optimal λ values for one experiment.

Formalizing the 'Lasso Order Search', we are proposing to simply replace the search through the set $U_{i,\prec}$, which has size $O(d^K)$, with $U_{i,\prec}^{L1}$, the subsets encountered along the L1-regularization path for node i given the ordering $\prec$. This set has size $O(d)$ in the linear regression case, and we conjecture also in the GLM case. More precisely, our new scoring function is

$$\begin{aligned} \text{score-L1}(\prec) &= \sum_{i=1}^{d} \text{score-BIC}(i, \pi_{i,\prec}^{L1}) && (6) \\ \pi_{i,\prec}^{L1} &= \arg\max_{U \in U_{i,\prec}^{L1}} \text{score-BIC}(i, U) && (7) \end{aligned}$$

To justify the restriction to the subsets on the regularization path, we note that recently it has been shown that under reasonable assumptions L1 regularization is a consistent estimator of undirected graph structure (21; 33), and hence of parent sets, given the right value of λ. Furthermore, the BIC method for choosing regularization constants such as λ is known to be a consistent estimator. We conjecture that our combination of L1 regularization and BIC selection is a consistent estimator of DAG structure given an ordering $\prec$.

By restricting ourselves to parent sets on the regularization path, we can save significant time. Specifically, in the linear regression setting, we can evaluate the score of a node in $O(Nd^3)$ time, and the score of an ordering in $O(Nd^4)$. In the GLM setting, we conjecture it takes at most $O(Nd^4)$ per node and $O(Nd^5)$ overall. (It is certainly polynomial in d, and independent of the number of parents.) By contrast, the TK algorithm takes $O(Nd^3d^K)$ time even in the linear case. This restricts the TK algorithm to small K, but models with overly resticted fan-in often fit the data rather poorly.

Note that we only have to evaluate the score of an order from scratch when we do a restart of the algorithm. If we restrict the order search to twiddle operations, then we only have to update the score of four nodes, rather than all d, whenever we take a local move.

For linear and binary logistic regression, there is a 1:1 correspondence between weights and edges in the graph. For multinomial logistic regression, each edge has a vector of weights associated with it, so it is desirable to penalize groups of variables (as in the 'Group Lasso' (23)). This technique can also be used to handle non-linear relationships between nodes, by performing basis function expansions, and assigning basis functions to appropriate groups. This

[1] The above argument assumes that we are evaluating BIC using the penalized maximum likelihood parameter estimates. It is more common to use the unpenalized MLE. To this end, we modified the LARS algorithm such that its updated/downdated (Cholesky) matrix factorization is used to serve the dual purpose of computing the regularization path, *and* computing the MLE parameters for all subsets encountered along the regularization path. The total cost of LARS with this modfication remains $O(Nd^3)$, rather than $O(Nd^4)$ if the MLEs were computed independently. (In the GLM case, it seems we cannot use this trick, so we conjecture the cost is $O(Nd^4)$.)

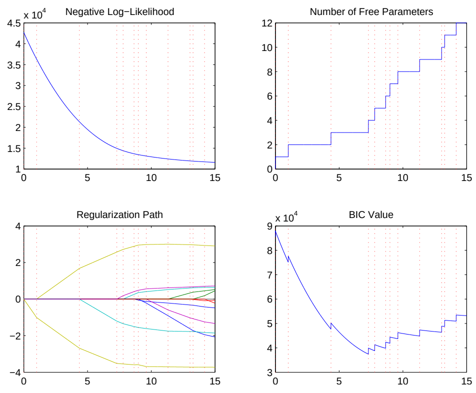

Figure 1: Regularization path for a linear regression problem. The horiztonal axis is the bound on the L1 norm of the coefficient vector, i.e., at location t we have $||\theta||_1 \leq t$. The complexity factor for the number of parameters, which is usually $\frac{\log N}{2}$, has been artifically scaled up to make the steps easier to see.

strategy allows a higher expressive power, without the exponential cost associated with a tabular CPD representation.

4 Experimental results

We compared three different methods for finding a high scoring DAG: (i) DAGSearch: searching in DAG space by picking the best edge deletion/addition/reversal, (ii) SS(K): searching through order space with twiddle operations and considering all possible subsets of parents up to size K (similar to the TK method)[2], and (iii) LassoDAG: our proposed strategy of searching through order space with twiddle operations and considering all subsets that occur along the nodes' regularization paths. Our search algorithm consisted of the simple strategy of running the method until it reached a local minimum, then randomly restarting with a new ordering/DAG. For the ordering-based methods, the same random orderings are given to SS(K) and LassoDAG in order to ensure that the observed difference in performance is due to the methodology rather than the random seed.

Since our proposed method is a means to reduce the number of family evaluations while not severely restricting the model class, we measured performance in terms of the best BIC score found relative to the number of family evaluations (ie. number of regression problems solved) during the search. We have not yet finished the discrete version, so we will report results using continuous data. However, we expect the results to be qualitatively similar in the discrete case.

We first looked at the well-known alarm network (37 nodes). We generated 10,000 samples with linear Gaussian CPDs with zero mean, unit variance, and randomly chosen regression weights (sampled from $1 + \mathcal{N}(0, 1)/4$ to ensure that there is non-negligible correlation between parents and children). We then tried to find the best BIC-scoring model. The results are shown in Figure 2 and 3. The many small line segments in Figure 2 indicate that LassoDAG and SS(K) both rapidly reach local optima and then restart. However, the SS(K) methods are limited in their performance because of their fan-in bound: the alarm network has some nodes with 4 parents, so $K = 1$ or $K = 2$ hurts performance. DAGSearch takes longer to reach a local optimum, and consequently performs fewer restarts. However it eventually reaches solutions of the same quality as LassoDAG, but moves more slowly though the search space and explores DAGs that are pruned with order-based strategies (as can be seen by the fact that DAGsearch starts at a much higher cost each time).

Next we tried sampling an interventional dataset from the alarm network, by performing perfect interventions (24) on randomly chosen nodes (these nodes were clamped to random values). Such interventional data is essential to choose between members of the same Markov equivalence class. We modified the BIC scoring function to handle interventional data using the technique described in (7). In Figure 6 we show the structure that was learned after 100,000 family evaluations of LassoDAG. We see that it has correctly recovered much of the original structure, including edge orientations (compare this to the true generating structure in Figure 5).

[2]Note that the optimal SS(1) could be found using the minimum spanning tree algorithm (5).)

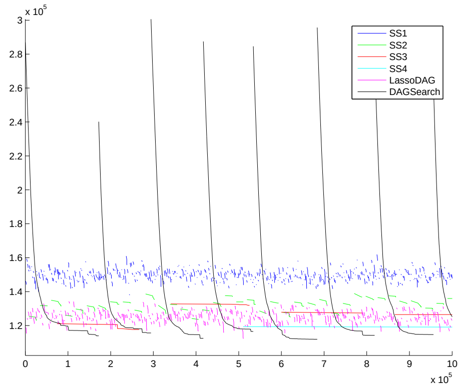

Figure 2: Cost (-BIC) versus number of family evaluations on the alarm network with 10,000 Gaussian samples, showing the effect of restarts. This figure is best viewed in colour.

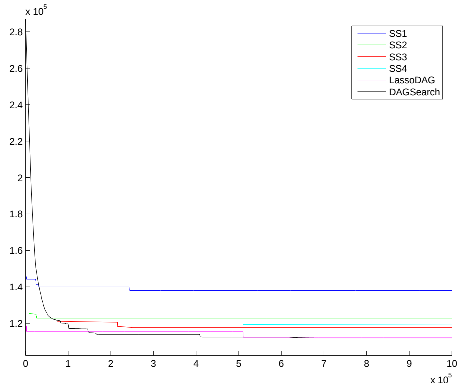

Figure 3: Lower envelope of Figure 2.

To assess the ability of the algorithm to scale to larger problems, we tested our method on the preprocessed version of the '20 Newsgroups' data available on Sam Roweis' webpage. This consists of 16,242 instances of 100 word occurence indicators, where we expect a generative model to be very densely connected. Although we ultimately would like to use a discrete model, one can still fit a Gaussian model to this $\{0, 1\}$ data. The final costs (negative BIC scores) after 100 thousand and 1 million family evaluations were (in multiples of 10^6): DAGsearch $1.59, 1.53$, SS(1) $1.55, 1.55$, SS(2)

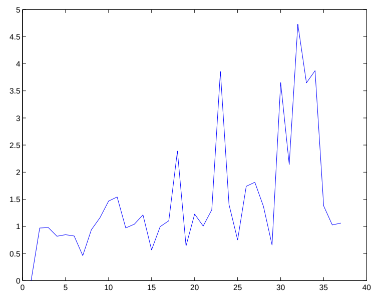

Figure 4: Plot of the optimal L1 bound on the weights, $||\theta_i||_1 \leq t$, versus node number i for the best scoring alarm network found by LassoDAG. We found that using the same (mean) value of t for all nodes resulted in much worse performance.

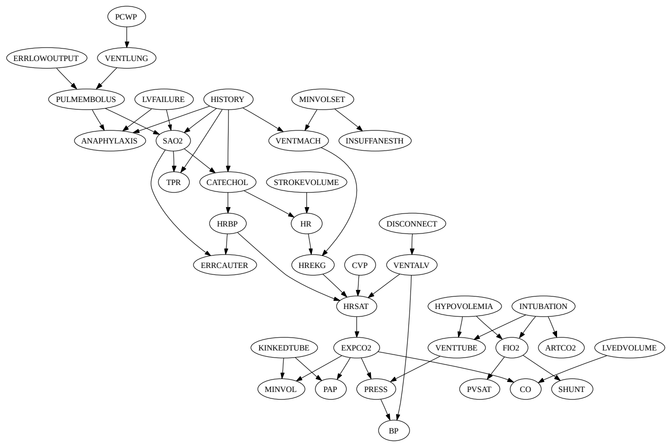

Figure 5: Alarm network (true structure).

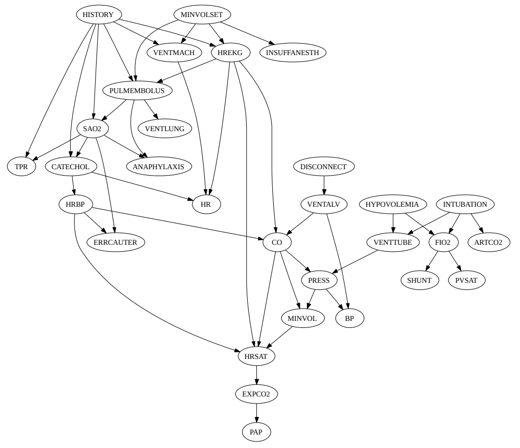

Figure 6: Alarm network learned on interventional data using LassoDAG after 100,000 family evaluations. We used a threshold of 0.1 on the weights, so that we do not visualize very weak connections.

$1.53, 1.53$, and LassoDAG $1.49, 1.49$. Thus LassoDAG finds the best local optimum (amongst these methods) in the given amount of time.

The resulting graph structures are fairly intuitive. For example, there is a cluster of nodes about sports (hockey, baseball, NHL), a cluster about computers (dos, windows, pc), and these clusters are connected via the word "win" (as in to "win in sports", or as in "winXP"). Unfortunately 100 node graphs are much too large to display in this paper, but we will make them available (along with source code) on the web.

5 Summary and future work

We have shown that by using GLMs and L1 penalized regression, we can find high scoring models much faster than other techniques. However, eventually DAGsearch catches up and sometimes outperforms us (especially in terms of structural error with respective to the truth). This suggest an obvious hybrid strategy which we will try in the future, namely to use LassoDAG to get to a local optimum, and then to use DAGsearch to refine the search. This should work well since it will restrict DAGsearch to starting with DAGs that are amongst the best $d!$ graphs, instead of starting from a random DAG. The other extension we will try in the future is to replace linear regression with logistic regression, so we can handle discrete data.

References

[1] J. F. and. R. Z. Li. Variable selection via non-concave penalized likelihood and its oracle properties. *J. of the Am. Stat. Assoc.*, 96(456):1348–1360, 2001.

[2] O. Banerjee, L. E. Ghaoui, et al. Convex optimization techniques for fitting sparse gaussian graphical models. In *Intl. Conf. on Machine Learning*. 2006.

[3] W. Buntine. Theory refinemement on Bayesian networks. In *UAI*. 1991.

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

[5] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–67, 1968.

[6] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[7] G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *UAI*. 1999.

[8] J. Dahl, V. Roychowdhury, et al. Maximum likelihood estimation of gaussian graphical models: Numerical implementation and topology selection. Tech. rep., UCLA, 2005.

[9] D. Dobra, C. Hans, et al. Sparse graphical models for exploring gene expression data. *J. Multivariate analysis*, 90:196–212, 2004.

[10] B. Efron, I. Johnstone, et al. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[11] 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.

[12] A. Genkin, D. Lewis, et al. Large-scale bayesian logistic regression for text categorization. Submitted.

[13] D. Heckerman, D. Chickering, et al. Dependency networks for density estimation, collaborative filtering, and data visualization. *J. of Machine Learning Research*, 1:49–75, 2000.

[14] D. Heckerman, D. Geiger, et al. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 1995.

[15] J. Huang, N. Liu, et al. Covariance selection and estimation via penalized normal likelihood. *Biometrika*, 93(1):85–98, 2006.

[16] M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *J. of Machine Learning Research*, 5:549–573, 2004.

[17] B. Krishnapuram, L. Carin, et al. Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2005.

[18] 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.

[19] S.-I. Lee, V. Ganapathi, et al. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems (NIPS 2006)*. 2007.

[20] F. Li and Y. Yang. Using modified lasso regression to learn large undirected graphs in a probabilistic framework. In *AAAI*. 2005.

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

[22] A. Moore and W.-K. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Intl. Conf. on Machine Learning*, pp. 552–559. 2003.

[23] M. Park and T. Hastie. L1 regularization path algorithm for generlaized linear models. Tech. rep., Dept. Statistics, Stanford, 2006.

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

[25] S. Perkins, K. Lacker, et al. Grafting: Fast, incremental feature selection by gradient descent in function space. *JMLR*, pp. 1333–1356, 2003.

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

[27] T. Silander and P. Myllmaki. A simple approach for finding the globally optimal Bayesian network structure. In *UAI*. 2006.

[28] A. Singh and A. Moore. Finding optimal bayesian networks by dynamic programming. Tech. rep., Carnegie Mellon University, June 2005.

[29] P. Spirtes, C. Glymour, et al. *Causation, Prediction, and Search*. MIT Press, 2000. 2nd edition.

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

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

[32] I. Tsamardinos, L. Brown, et al. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 2006. To appear.

[33] M. Wainwright, P. Ravikumar, et al. Inferring graphical model structure using ℓ_1-regularized pseudo-likelihood. In *NIPS*. 2006.

[34] M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. Tech. rep., Georgia Tech, 2005.