Column generation for exact BN learning: Work in progress

James Cussens[1]

1 Introduction

In existing integer linear programming (ILP) approaches to learning the structure of a Bayesian network (BN) from data ([1, 4, 2]) a binary variable $I(W \rightarrow u)$ is created for each BN variable $u \in V$ and each candidate parent set W. $I(W \rightarrow u)$ takes the value 1 iff W is the parent set for u in the optimal DAG. The number of candidate parent sets for any given BN variable is artificially restricted to keep n, the number of $I(W \rightarrow u)$ variables, reasonable and then a local score $c'(u, W)$ is pre-computed for each of them. With this approach the BN learning problem can be cast as in (1).

$$\begin{array}{l}\text{Instantiate the } I(W \rightarrow u) \text{ to maximise:} \\ \sum_{u,W} c'(u,W) I(W \rightarrow u) \\ \text{subject to the } I(W \rightarrow u) \text{ representing a DAG.}\end{array} \quad (1)$$

In this paper it will be more convenient to convert this into a minimisation problem, so set $c(u, W) = -c'(u, W)$ and consider

$$\begin{array}{l}\text{Instantiate the } I(W \rightarrow u) \text{ to minimise:} \\ \sum_{u,W} c(u,W) I(W \rightarrow u) \\ \text{subject to the } I(W \rightarrow u) \text{ representing a DAG.}\end{array}$$

The goal is thus to find a BN with the lowest negative score. To ensure that each BN variable has exactly one parent set the following $|V|$ constraints are used:

$$\forall u \in V : \sum_{W} I(W \rightarrow u) = 1 \quad (2)$$

and to ensure that the resulting graphs are acyclic, *cluster constraints* are added in the course of solving as *cutting planes*.

$$\text{Where } C \subseteq V : \sum_{u \in C} \sum_{W : W \cap C = \emptyset} I(W \rightarrow u) \geq 1 \quad (3)$$

Cluster constraints were introduced in [4] and were later used in [2]. It is not necessary to add all (exponentially many) cluster constraints. A lower bound on the negative score of the optimal BN (the dual bound) increases (or occasionally remains constant) as cluster constraints are added. Once a BN has been found whose score equals this lower bound it follows that an optimal BN has been found.

2 The simplex algorithm

The inequalities (3) can be replaced by equations using 'slack' variables w_C:

$$-w_C + \sum_{u \in C} \sum_{W : W \cap C = \emptyset} I(W \rightarrow u) = 1 \quad (4)$$

[1] University of York, email:james.cussens@york.ac.uk

where each w_C is constrained to be positive ($w_C \geq 0$). Let $\mathcal{C}$ be the current set of clusters. The full set of constraints for the ILP problem is then a collection of $|V| + |\mathcal{C}|$ linear equations involving $(n + |\mathcal{C}|)$ variables. Following the notation and presentation given in [5], let $\mathbf{x}$ be the vector of these ILP variables (in some arbitrary order), then these constraints can be written in matrix from as:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (5)$$

where $\mathbf{A}$ is a $(|V| + |\mathcal{C}|) \times (n + |\mathcal{C}|)$ matrix. Note from (2) and (4) that $\mathbf{b}$ will be a column of ones.

Consider a solution to (5) corresponding to a completely unconnected graph. The variables $I(\emptyset \rightarrow u)$ will have value 1, all other $I(W \rightarrow u)$ variables will have value 0 and the slack variables will have the values determined by the equations (4). Let $\zeta = \sum_{u,W} c(u,W) I(W \rightarrow u)$ be a variable representing the objective value and let $\bar{\zeta} = \sum_u c(u, \emptyset)$ be the constant which is the score for the completely unconnected graph. It is not difficult to see that we can write the objective function and the constraints as follows:

$$\zeta = \bar{\zeta} + \sum_{u, W : W \neq \emptyset} [c(u,W) - c(u,\emptyset)] I(W \rightarrow u) \quad (6)$$

$$I(\emptyset \rightarrow u) = 1 - \sum_{u, W : W \neq \emptyset} I(W \rightarrow u) \quad (7)$$

$$w_C = |C| - 1 - \sum_{u \in C} \sum_{W : W \cap C \neq \emptyset} I(W \rightarrow u) \quad (8)$$

where (7) represents an equation for each $u \in V$ and (8) represents an equation for each cluster $C \in \mathcal{C}$. Essentially (7) has been used to eliminate all $I(\emptyset \rightarrow u)$ variables from the RHS of each equation.

The equations (6–8) are called a *dictionary* [6]. The variables on the RHS of the equations, which are all set to 0, are the *non-basic* variables, those on the LHS of (7) and (8) are the *basic* variables. Note that there are $n - |V|$ non-basic variables and $|V| + |\mathcal{C}|$ basic variables.

To improve the value of ζ the simplex algorithm considers the coefficients $[c(u,W) - c(u,\emptyset)]$ of the non-basic variables on the RHS of (6). These coefficients are called the *reduced costs* of the non-basic variables. If a variable has negative reduced cost then raising its values from zero will improve (reduce) the value of ζ. So a variable with negative reduced cost is increased until one of the basic variables becomes set to zero. These variables are called the *entering* and *leaving* variables respectively. The entering variable moves from non-basic to basic ('enters the basis') and the leaving variable moves from basic to non-basic. A new dictionary is created (implicitly in practice) where the entering variable is removed from all RHS, being replaced by a linear expression involving the leaving variable. This ensures that all basic variables and the objective value are represented

as linear functions of non-basic variables (which means that reduced costs are available for all non-basic variables). This process continues until no non-basic variable has negative reduced cost at which point the linear program is solved.

3 Column generation

The key idea behind column generation is that *it is not necessary to* explicitly *represent non-basic variables*. So an as-yet-nonexistent variable can be viewed as a non-basic variable which we have yet to consider as a possible entrant into the basis. Note that variables correspond to columns of the matrix $\mathbf{A}$ in (5) so that generating a new variable corresponds to generating a new column in that matrix. The goal of column generation is to create a new variable with negative reduced cost.

We consider now how to compute reduced costs. Recall that at each iteration of the simplex algorithm we have $|V| + |\mathcal{C}|$ basic and $n - |V|$ non-basic variables. Write $\mathbf{x} = (\mathbf{x_B}, \mathbf{x_D})$ where $\mathbf{x_B}$ are the basic variables and $\mathbf{x_D}$ the non-basic. Decompose the objective coefficient vector $\mathbf{c}$ similarly: $\mathbf{c} = (\mathbf{c_B}, \mathbf{c_D})$. Let $\mathbf{B}$ be the submatrix of the original $\mathbf{A}$ matrix formed by selecting the columns of $\mathbf{A}$ corresponding to $\mathbf{x_B}$. Let $\mathbf{D}$ be the corresponding submatrix for non-basic variables. Note that $\mathbf{B}$, called the *basis matrix*, is a $(|V| + |\mathcal{C}|) \times (|V| + |\mathcal{C}|)$ square matrix. We can now compute $\lambda^T = \mathbf{c_B^T B^{-1}}$. λ is the vector of dual values for each row (=linear constraint) in $\mathbf{A}$. The vector of reduced costs for non-basic variables, denoted $\mathbf{r_D}$, can then be computed using $\mathbf{r_D} = \mathbf{c_D} - \lambda^T \mathbf{D}$.

Note that the dual vector λ is determined by the current basis. So to compute the reduced cost of a potential new variable we just need its objective coefficient value and its coefficient for each original linear constraint (= row of $\mathbf{A}$).

We now construct an ILP to identify a new variable with negative reduced cost. A new variable $I(W \to u)$ is determined by a choice of the child u and also the parents W. Let $I_{\mathrm{ch}}(u)$ indicate that u is chosen as the child and let $I_{\mathrm{pa}}(u)$ represent that u is chosen as a parent. We have the obvious constraints:

$$\sum_{u \in V} I_{\mathrm{ch}}(u) = 1 \tag{9}$$

$$\forall u \in V : I_{\mathrm{ch}}(u) + I_{\mathrm{pa}}(u) \leq 1 \tag{10}$$

A new variable $I(W \to u)$ will appear in a cluster constraint for $C \in \mathcal{C}$ (4) with coefficient 1 iff $u \in C$ and $v \notin C$ for each $v \in W$. Create a variable x_C for each $C \in \mathcal{C}$ indicating whether the new variable is involved in the constraint for C. We have:

$$x_C \geq \sum_{u \in C} I_{\mathrm{ch}}(u) - \sum_{u \in C} I_{\mathrm{pa}}(u) \tag{11}$$

$$\sum_{u \in C} I_{\mathrm{ch}}(u) \geq x_C \tag{12}$$

$$1 - \sum_{u \in C} I_{\mathrm{pa}}(u) \geq x_C \tag{13}$$

Let λ_C be the dual value corresponding to the constraint for cluster C. Let λ_u be the dual value for the convexity constraint (2) for variable u. The reduced cost for a new variable $I(W \to u)$ is then:

$$c(u, W) - \sum_u \lambda_u I_{\mathrm{ch}}(u) - \sum_{C \in \mathcal{C}} \lambda_C x_C \tag{14}$$

To find the best new variable to introduce we want to minimise (14) subject to constraints (9-13): an ILP. However, in (14), $c(u, W)$ the objective coefficient for the new variable, is unknown. The sensible option is to view $c(u, W)$ as an additional (real-valued) ILP variable. For the column generation method to produce useful new variables, it is essential that we can put a reasonably tight lower bound on $c(u, W)$ in terms of the variables $I_{\mathrm{ch}}(u)$, $I_{\mathrm{pa}}(u)$ and x_C and a small number of constants easily computable from the data. Since we only have a bound on $c(u, W)$ solving the ILP will generate a new variable together with an over-optimistic value for its reduced cost. If even this over-optimistic value is positive it follows that there are no new variables worth introducing and so the current set of $c(u, W)$ variables are all we need to identify the optimal BN. If the over-optimistic value is negative there is at least the possibility that the *actual* reduced cost is also negative. This makes it worth the effort to consult the data to compute the true reduced cost. If this turns out to be negative, the variable is created. Otherwise constraints should be added to rule out this choice for $c(u, W)$ and the ILP re-solved in the hope of finding a different variable with a (true) negative reduced cost.

The key to this approach is the tightness of the sought bound. Recent work in a companion paper [3] has produced the following lower bound on $c(u, W)$:

$$\begin{aligned} -c(u, W) \\ \leq\ & (\alpha - qr/2) \log r + (q(r-1)/2) \log(\frac{\alpha}{q}) \\ & -N H_{\tilde{p}}(u|W) \qquad (15) \\ & -\frac{1}{2} \sum_k \log \left(n_k + \frac{\alpha}{qr} \right) + \frac{1}{2} \sum_j \log \left(n_j + \frac{\alpha}{q} \right) \qquad (16) \end{aligned}$$

where: α is the *effective sample size*, r is the arity of u, q is the product of the arities of the variables in W, $H_{\tilde{p}}(u|W)$ is the conditional entropy of u given W according to a certain distribution $\tilde{p}$, N is the size of the data, the n_j are the counts in the contingency table for the variables W and the n_k the contingency table counts for $\{u\} \cup W$. A useful bound for the data-dependent (15) is available, but not, at present, for (16). It is hoped that with further work this lower bound on $c(u, W)$ can be used to allow effective column generation.

REFERENCES

[1] James Cussens. Maximum likelihood pedigree reconstruction using integer programming. In *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics (WCB-10)*, Edinburgh, July 2010.

[2] James Cussens. Bayesian network learning with cutting planes. In Fabio G. Cozman and Avi Pfeffer, editors, *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 153–160, Barcelona, 2011. AUAI Press.

[3] James Cussens. An upper bound for BDeu local scores. Submitted, May 2012.

[4] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 358–365, 2010. Journal of Machine Learning Research Workshop and Conference Proceedings.

[5] David G. Luenberger and Yinyun Ye. *Linear and nonlinear programming*. Springer, 2008.

[6] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, third edition, 2008.