Proceedings of Machine Learning Research vol 72, 272-283, 2018 PGM 2018

Learning Bayesian network classifiers with completed partially directed acyclic graphs

Bojan Mihaljević BMIHALJEVIC@FI.UPM.ES
Concha Bielza MCBIELZA@FI.UPM.ES
Pedro Larranaga PEDRO.LARRANAGA@FI.UPM.ES
Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid

Abstract

Most search and score algorithms for learning Bayesian network classifiers from data traverse the space of directed acyclic graphs (DAGs), making arbitrary yet possibly suboptimal arc directionality decisions. This can be remedied by learning in the space of DAG equivalence classes. We provide a number of contributions to existing work along this line. First, we identify the smallest subspace of DAGs that covers all possible class-posterior distributions when data is complete. All the DAGs in this space, which we call *minimal class-focused* DAGs (MC-DAGs), are such that their every arc is directed towards a child of the class variable. Second, in order to traverse the equivalence classes of MC-DAGs, we adapt the greedy equivalence search (GES) by adding operator validity criteria which ensure GES only visits states within our space. Third, we specify how to efficiently evaluate the discriminative score of a GES operator for MC-DAG in time independent of the number of variables and without converting the completed partially DAG, which represents an equivalence class, into a DAG. The adapted GES perfomed well on real-world data sets.

Keywords: equivalence class; greedy equivalence search; augmented naive Bayes.

1. Introduction

Bayesian network classifiers (Bielza and Larrañaga, 2014; Friedman et al., 1997) are interpretable models that offer competitive predictive performance (e.g., Zaidi et al., 2013). They include the popular naive Bayes (Minsky, 1961) and its augmented naive Bayes (Friedman et al., 1997) variants with arcs among the features. A common way to learn them (e.g., Keogh and Pazzani, 2002; Pazzani, 1996) is by optimizing a score in the space of directed acyclic graphs (DAGs). This may needlessly prune the search space by arbitrarily directing an arc when its reversal would yield an equivalent model. Two DAGs are equivalent if they impose identical independence constraints on the joint probability distribution $P(C, \mathbf{X})$, with C being the class and $\mathbf{X}$ the predictor variables. The equivalence relation partitions the set of DAGs into equivalence classes, and by searching in this space we only set arc direction when a reversal produces a non-equivalent model. The greedy equivalence search (GES) (Chickering, 2002a) is one algorithm that operates in this space. It represents an equivalence class with a *completed partially* DAG (CPDAG), which has both directed and undirected edges.

Acid et al. (2005) learned Bayesian network classifiers by traversing the space of equivalence classes. They pruned the space of DAGs to be considered by noticing that non-equivalent DAGs could, nonetheless, be *classification equivalent*, that is, encode an identical class-posterior distribution, $P(C \mid \mathbf{x})$, for every instance $\mathbf{x}$. They showed that, with complete training data (i.e., without missing values), a minimal classification-equivalent subgraph of any DAG, which they call a C-DAG, is such that each arc is either directed towards the class variable or a child of the class

variable. They traversed the equivalence classes of C-DAGs with a heuristic greedy search and a representation based on their previous work (Acid and de Campos, 2003), rather than the more standard CPDAG representation and its corresponding operators (Chickering, 2002a,b). Unlike the CPDAG, their representation is not a canonical representative of an equivalence class.

In this paper, we extend the work by Acid et al. (2005) in a number of ways. First, we show that the search can be reduced, without loss of generality, to a subset of the space of C-DAGs. Namely, we need not consider C-DAGs with parents for the class variable, $\mathbf{Pa}(C)$, because $P(C \mid \mathbf{x})$ is unaffected by marginal independences among $\mathbf{Pa}(C)$ while we can model the full dependencies among them conditional to C with them being children of C. Besides being smaller, we argue that this space of *minimal class-focused* DAGs, or MC-DAGs, is also more adequate for greedy forward learning of dependencies among $\mathbf{Pa}(C)$.

Second, we adapt the GES algorithm to traverse the space of MC-DAGs. We do this by providing GES operator validity conditions, that can be checked efficiently on a CPDAGs, that discard CPDAGs corresponding to equivalence classes that do not have an MC-DAG as a member.

Third, we specify how to, for complete data, efficiently evaluate the discriminative score of an operator locally, time independent of the number of variables, and on a CPDAG. This is based on a technique, described by Keogh and Pazzani (2002), for updating $P(C, \mathbf{x})$, for all $\mathbf{x}$ in our data set, for an arc addition. We adapt the technique for the GES operators over CPDAGs in the MC-DAG space.

We applied our method on thirteen real-world data sets using cross-validated accuracy as the learning score, and compared it to other greedy and non-greedy augmented naive Bayes classifiers. Our method outperformed them on three data sets while it was outperformed on one.

The rest of this paper is organized as follows. Section 2 introduces notation and terminology. Section 3 describes the MC-DAG space and discusses its advantages over the C-DAG space. Section 4 describes our adaptation of GES for learning MC-DAGs from data. Section 5 presents the local updating of $P(C, \mathbf{X})$, used to compute discriminative scores. Section 6 shows the evaluation of our algorithm on real-world data sets. We conclude in Section 7.

2. Preliminaries

We are interested in modelling a distribution over a set of variables $\{C, \mathbf{X}\}$, with C being a discrete variable representing the class, and $\mathbf{X} = (X_1, \ldots, X_n)$ being n discrete or real-valued predictor variables. We also use X, Y, Z and Q to refer to variables in $\{C, \mathbf{X}\}$. A Bayesian network $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ models a probability distribution $P_{\mathcal{G}}(\mathbf{x}, c)$ (lowercase $\mathbf{x}$ denotes an assignment to $\mathbf{X}$). $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ is a directed acyclic graph (DAG) with vertices (i.e., nodes) $\mathbf{V}$ corresponding to variables in $\{C, \mathbf{X}\}$, and directed edges (i.e., arcs) $\mathbf{E}_{\mathcal{G}}$ among the vertices. $P_{\mathcal{G}}(\mathbf{x}, c)$ factorizes according to $\mathcal{G}$,

$$P_{\mathcal{G}}(\mathbf{x}, c) = P_{\mathcal{G}}(c \mid \mathbf{pa}_{\mathcal{G}}(c)) \prod_{i=1}^{n} P_{\mathcal{G}}(x_i \mid \mathbf{pa}_{\mathcal{G}}(x_i)),$$

where $\mathbf{pa}_{\mathcal{G}}(x)$ are the values of parents of X in $\mathcal{G}$. $\mathcal{G}$ imposes independence constaints on $P_{\mathcal{G}}(\cdot)$, and they can all be derived from the constraint that each variable is independent of its non-descendents in $\mathcal{G}$ given its parents. $X_1 \perp\!\!\!\perp_{\mathcal{G}} X_2 \mid C$ denotes that X_1 is conditionally independent of X_2 given C in $P_{\mathcal{G}}(\cdot)$. The parameters $\boldsymbol{\theta}$ specify the local conditional distributions of each variable given its parents' values.

We learn $\mathcal{B}$ from a data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(N)}, c^{(N)})\}$ of N observations of $\mathbf{X}$ and C. One approach is to search a space of possible structures in order to optimize a network quality score. Learning the optimal structure with at most two parents per variable is NP-hard (Chickering et al., 2004). Thus, heuristic search algorithms, such as greedy hill-climbing, are commonly used (see e.g., Koller and Friedman, 2009). The scores are divided into generative ones, based on $P(c, \mathbf{x})$, and discriminative ones based on $P(c \mid \mathbf{x})$, such as classification accuracy. The former are decomposable with respect to $\mathcal{G}$ and thus allow for efficient local updates during the search. The latter, although not decomposable, are more suitable to learning classifiers when N is small and n large (Friedman et al., 1997). All discriminative and many generative scores are *score-equivalent*, that is, they score all equivalent DAGs equally.

Two DAGs $\mathcal{G}$ and $\mathcal{H}$ are equivalent if the independence constraints that they impose on $P_{\mathcal{G}}(\cdot)$ and $P_{\mathcal{H}}(\cdot)$, respectively, are identical. Searching in this space requires a score-equivalent function and a way to represent an equivalence class of DAGs. An acyclic partially DAG (PDAG) $\mathcal{P}$, containing both directed and undirected edges, represents the class of DAGs equivalent to a DAG $\mathcal{G}$ obtained by orienting undirected edges in $\mathcal{P}$. We say that such a $\mathcal{G}$ is a consistent extension of $\mathcal{P}$, $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$, while any DAG $\mathcal{H}$ equivalent to $\mathcal{G}$ is a member of the equivalence class of DAGs corresponding to $\mathcal{P}$, $\mathcal{H} \in \mathcal{E}(\mathcal{P})$. A completed PDAG (CPDAG) $\mathcal{P}$ for en equivalence class $\mathcal{E}(\mathcal{P})$ is the PDAG with an oriented edge for every edge that is identically oriented in every $\mathcal{G} \in \mathcal{E}(\mathcal{P})$, and an undirected edge for all other edges in $\mathcal{E}(\mathcal{P})$. We refer to the directed arcs in a CPDAG as compelled for the equivalence class, and to the undirected ones as reversible. A CPDAG $\mathcal{P}$ is unique for an equivalence class and has every $\mathcal{G} \in \mathcal{E}(\mathcal{P})$ as a consistent extension, $\mathbf{cext}(\mathcal{P}) = \mathcal{E}(\mathcal{P})$.

For a PDAG $\mathcal{P}$, $\mathbf{Ch}_{\mathcal{P}}(X)$ denotes the children of X in $\mathcal{P}$, $\mathbf{Nbr}_{\mathcal{P}}(X)$ the neighbour nodes connected to X in $\mathcal{P}$ by an undirected edge, and $\mathbf{Adj}_{\mathcal{P}}(X) = \mathbf{Ch}_{\mathcal{P}}(X) \cup \mathbf{Pa}_{\mathcal{P}}(X) \cup \mathbf{Nbr}_{\mathcal{P}}(X)$ the nodes adjacent to X in $\mathcal{P}$. A v-structure is an ordered triple of nodes (X, Y, Z) such that $\mathcal{P}$ contains the edges $X \to Y$ and $Z \to Y$, and X and Z are not adjacent in $\mathcal{P}$.[1]

The GES algorithm starts from a CPDAG $\mathcal{P} = (\mathbf{V}, \mathbf{E}_{\mathcal{P}} = \emptyset)$ and proceeds with the Insert$(X, Y, \mathbf{T})$ operator (to be defined below) considering all arc additions to every DAG in the current equivalence class $\mathcal{E}(\mathcal{P})$. It adds the best among the considered arcs and sets the equivalence class of the obtained DAG as the new state $\mathcal{P}'$, and applies the Insert$(X, Y, \mathbf{T})$ operator to $\mathcal{P}'$. Once it reaches a local optimum, it starts its backward phase, with the Delete$(X, Y, \mathbf{H})$ operator (to be defined below) considering the removal of every arc in every DAG in the current equivalence class. The operators are scored locally on the CPDAG $\mathcal{P}$, without generating the DAGs to which the visited states correspond. A set of conditions verifiable on $\mathcal{P}$ ensure that the operators correspond to valid DAGs in the desired neighbourhood of $\mathcal{P}$.

3. Minimal C-DAGs

We begin with the definitions of classification equivalence and class-focused DAGs, or C-DAGs.

Definition 1 (Acid et al. (2005)) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ and $\mathcal{G}' = (\mathbf{V}, \mathbf{E}_{\mathcal{G}'})$ be two DAGs. Let P be any joint probability distribution on $\mathbf{V}$, and $P_{\mathcal{G}}$ and $P_{\mathcal{G}'}$ be the probability distributions that factorize according to $\mathcal{G}$ and $\mathcal{G}'$, respectively, defined as $P_{\mathcal{G}}(V \mid \mathbf{pa}_{\mathcal{G}}(V)) = P(V \mid \mathbf{pa}_{\mathcal{G}}(V))$ and $P_{\mathcal{G}'}(V \mid \mathbf{pa}_{\mathcal{G}'}(V)) = P(V \mid \mathbf{pa}_{\mathcal{G}'}(V))$, $\forall V \in \mathbf{V}$. If $P_{\mathcal{G}}(C \mid \mathbf{x}) = P_{\mathcal{G}'}(C \mid \mathbf{x})$ $\forall \mathbf{x}$, we say that $\mathcal{G}$ and $\mathcal{G}'$ are classification-equivalent.*

1. These definition are easily specialized for DAGs.

That is, $\mathcal{G}$ and $\mathcal{G}'$ are classification-equivalent if their class-posterior distributions are identical for any value of $\mathbf{x}$.

Definition 2 (Acid et al. (2005)) *A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E}_\mathcal{G})$ is a class-focused DAG (C-DAG) with respect to the variable C if and only if it satisfies the following condition: $\forall X, Y \in \mathbf{V}$, if $X \to Y \in \mathbf{E}_\mathcal{G}$ then either $Y = C$ or $X = C$ or $C \to Y \in \mathbf{E}_\mathcal{G}$.*

In words, in a C-DAG only C and children of C can have parents (see Figure 1). Acid et al. (2005) showed that, for any DAG $\mathcal{H}$, its C-DAG subgraph $\mathcal{H}_C$, induced by including only arcs that match Definition 2, is its minimal classification-equivalent subgraph. Acid et al. (2005) searched the space of C-DAGs to learn Bayesian network classifiers because it covers all possible class-posterior distributions.

We now show that a smaller space is sufficient. Namely, we need not consider parents for the class variable C. That is, for each DAG $\mathcal{H}$ and its C-DAG subgraph $\mathcal{H}_C$, there is a classification-equivalent C-DAG $\mathcal{G}$ with no parents for C. We first formalize such a minimal class-focused DAG (MC-DAG) structure, and then show how to convert a C-DAG into a classification-equivalent MC-DAG.

Definition 3 *A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E}_\mathcal{G})$ is a minimal class-focused DAG (MC-DAG) with respect to the variable C if and only if it satisfies the following condition: $\forall X, Y \in \mathbf{V}$, if $X \to Y \in \mathbf{E}_\mathcal{G}$ then $C \to Y \in \mathbf{E}_\mathcal{G}$.*

In words, an MC-DAG is a C-DAG that only allows children of C to have parents (see Figure 1). Dispensing with the parents of C, $\mathbf{Pa}_\mathcal{H}(C)$, while maintaining classification equivalence, is possible by observing that, unlike $P_\mathcal{H}(c, \mathbf{x})$, $P_\mathcal{H}(c \mid \mathbf{x})$ is unaffected by conditional independence constaints that hold only when C is not observed. This is because we compute $P(c \mid \mathbf{x})$ as $\propto P(c)P(\mathbf{x} \mid c)$, setting each value of C as evidence. Thus, for the C-DAG $\mathcal{H}$, where $X \perp\!\!\!\perp_\mathcal{H} Y$ necessarily holds for all $X, Y \in \mathbf{Pa}_\mathcal{H}(C)$, there exists a DAG $\mathcal{G}$ such that $X \not\perp\!\!\!\perp_\mathcal{G} Y$ for all $X, Y \in \mathbf{Pa}_\mathcal{H}(C)$ while nonetheless $P_\mathcal{H}(c \mid \mathbf{x}) = P_\mathcal{G}(c \mid \mathbf{x})$. To account for $X \not\perp\!\!\!\perp_\mathcal{H} Y \mid C$ for all $X, Y \in \mathbf{Pa}_\mathcal{H}(C)$ it suffices for $\mathcal{G}$ to have (1) every $X \in \mathbf{Pa}_\mathcal{H}(C)$ as a child of C and (2) an arc for every pair $\{X, Y\}$, for all $X, Y \in \mathbf{Pa}_\mathcal{H}(C)$. Thus, any C-DAG $\mathcal{H}$ can be represented by an MC-DAG $\mathcal{G}$ with $\mathbf{Pa}_\mathcal{G}(C) = \emptyset$. We prove this by showing that a C-DAG $\mathcal{H}$ can be converted into a classification-equivalent C-DAG $\mathcal{H}'$ with one class parent less (Proposition 4). The conversion to an MC-DAG follows by repeating such parent removals until there are no more parents of C (Proposition 5). Figure 1 (above) shows a C-DAG $\mathcal{H}$ and its classification-equivalent MC-DAG $\mathcal{G}$.

Proposition 4 *For a C-DAG $\mathcal{H}$ with $X \to C$ there is a classification-equivalent C-DAG $\mathcal{H}'$ such that $X \to C$ is reversed to $C \to X$.*

Proof Consider a C-DAG $\mathcal{H}$ with $X \in \mathbf{Pa}_\mathcal{H}(C)$. Let $\mathcal{H}' = \mathcal{H}$. By Definition 2, $\mathbf{Pa}_{\mathcal{H}'}(X) = \emptyset$ and $\mathbf{Ch}_{\mathcal{H}'}(X) \supseteq C$. Now reverse the arc $X \to C$ in $\mathcal{H}'$ into $C \to X$. $\mathcal{H}'$ is a valid DAG because we did not introduce a cycle: a $Y \in \mathbf{Ch}_{\mathcal{H}'}(X)$ cannot be an ancestor of C since, by Definition 2, $Y \in \mathbf{Ch}_{\mathcal{H}'}(C)$ and a path from Y to C would imply that there was a cycle in $\mathcal{H}'$ before the reversal. Since $\mathbf{Pa}_{\mathcal{H}'}(X) = \emptyset$, the reversal did not introduce a v-structure centered around X. The reversal did drop the v-structures $X \to C \leftarrow Y$ centered around C, $\forall Y \in \mathbf{Pa}_\mathcal{H}(C) \setminus X$, replacing them with serial connections $Y \to C \to X$. After the reversal, $\mathcal{H}'$ and $\mathcal{H}$ differ as follows: for all

$Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$, (1) $X \perp\!\!\!\perp_{\mathcal{H}} Y$ while $X \not\perp\!\!\!\perp_{\mathcal{H}'} Y$; and (2) $X \not\perp\!\!\!\perp_{\mathcal{H}} Y \mid C$ while $X \perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$. (1) does not affect classification equivalence of $\mathcal{H}$ and $\mathcal{H}'$ since we compute $P(c \mid \mathbf{x})$ by always conditioning on C, via $\propto P(c)P(\mathbf{x} \mid c)$. (2) can be remedied by adding to $\mathcal{H}'$ an arc $Y \rightarrow X$ for every $Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$. Now add one such arc $Y \rightarrow X$ to $\mathcal{H}'$. This introduced no cycles in $\mathcal{H}'$ because $\mathbf{Pa}_{\mathcal{H}'}(Y) = \emptyset$ and therefore there is no directed path from X to Y in $\mathcal{H}'$. Before adding $Y \rightarrow X$, the only parent of X in $\mathcal{H}'$ was C, $\mathbf{Pa}_{\mathcal{H}'}(X) = C$, and thus adding $Y \rightarrow X$ only introduced the v-structure $Y \rightarrow X \leftarrow C$. Since $Y \in \mathbf{Pa}_{\mathcal{H}'}(C)$, $Y \not\perp\!\!\!\perp_{\mathcal{H}'} C \mid X$ was already true, and the only effect of adding $Y \rightarrow X$ was to render $X \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$, which was our purpose (because $X \not\perp\!\!\!\perp_{\mathcal{H}} Y \mid C$). After the addition, $\mathbf{Pa}_{\mathcal{H}'}(X) = \{Y, C\}$. Now add to $\mathcal{H}'$ the arc $Q \rightarrow X$, for $Q \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus \{X, Y\}$. The only independence constraint in $\mathcal{H}'$ conditional to C that this addition modified is that now $Q \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$, because the two introduced v-structures, $Q \rightarrow X \leftarrow C$ and $Q \rightarrow X \leftarrow Y$, modified no such constraints. The reasoning regarding $Q \rightarrow X \rightarrow C$ in analogous to that for $Y \rightarrow X \rightarrow C$. Regarding $Q \rightarrow X \leftarrow Y$, $Q \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$ was already true due to the v-structure $Q \rightarrow C \leftarrow Y$ in $\mathcal{H}'$. For each next $Z \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus \{X, Y, Q\}$ added to $\mathcal{H}'$ it follows by induction that the only independence constraint conditional to C modified is to render $Z \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$. After adding to $\mathcal{H}'$ arcs $Y \rightarrow X$ for every $Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$, $Y \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$ holds and there are no independence constraints conditional to C holding in $\mathcal{H}'$ that do not also hold in $\mathcal{H}$. Thus, $\mathcal{H}'$ is classification-equivalent to $\mathcal{H}$. It is easy to see that it is also a C-DAG. ■

Proposition 5 *For each C-DAG $\mathcal{H}$ there is a classification-equivalent MC-DAG.*

Proof The proof is constructive. Start with $\mathcal{G} = \mathcal{H}$. At each step, produce a classification-equivalent C-DAG $\mathcal{G}'$ with one less parent of C than $\mathcal{G}$. Set $\mathcal{G} = \mathcal{G}'$. Repeated application will produce a classification-equivalent MC-DAG. ■

Any DAG $\mathcal{H}$ and its unique C-DAG $\mathcal{H}_C$ subgraph can be mapped to multiple equivalent MC-DAGs, obtained by choosing the arcs $X \rightarrow C$ which to reverse in a different order. An MC-DAG $\mathcal{G}$ is not a subgraph of $\mathcal{H}_C$ (unless $\mathbf{Pa}_{\mathcal{H}_C}(C) = \emptyset$ and $\mathcal{H}_C$ is an MC-DAG itself), as it contains the reversed arcs to $\mathbf{Pa}_{\mathcal{H}_C}(C)$ and arcs among $\mathbf{Pa}_{\mathcal{H}_C}(C)$ (shown in red in Figure 1). Note that $P_{\mathcal{H}_C}(c, \mathbf{x}) = P_{\mathcal{G}}(c, \mathbf{x})$ need not hold and thus the generative scores of $\mathcal{H}_C$ and $\mathcal{G}$ might differ. While an MC-DAG may have more parameters than a classification-equivalent C-DAG, due to the added arcs among $\mathbf{Pa}_{\mathcal{H}_C}(C)$, these parameters only make explicit the lack of independence among $\mathbf{Pa}_{\mathcal{H}_C}(C)$ conditional on C already present in $P_{\mathcal{H}_C}(c, \mathbf{x})$.

Since every C-DAG is an MC-DAG, but not vice-versa, the MC-DAG space is smaller. It is nonetheless sufficient, as for each C-DAG there is at least one classification-equivalent MC-DAG. We argue that it is also more suitable for greedy learning algorithms. Consider a forward search starting from an empty graph $\mathcal{H}' = (\mathbf{V} = \{X, Y, Z, C\}, \mathbf{E}_{\mathcal{G}'} = \emptyset)$, using penalized log-likelihood to learn from $\mathcal{D}$, a limited-N sample taken from $\mathcal{B} = (\mathcal{G}', \boldsymbol{\theta})$, with $\mathcal{G}' = (\{X, Y, Z, C\}, \mathbf{E}_{\mathcal{G}'})$ shown in Figure 1. After two iterations, the search may have reached the state with $\mathbf{E}_{\mathcal{H}'} = \{X \rightarrow C, C \leftarrow Y\}$. Then, its only option to account for $X \not\perp\!\!\!\perp_{\mathcal{G}'} Z \mid C$ is to add Z as a parent of C in $\mathcal{H}'$. This, however, renders $Z \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$, although $Z \perp\!\!\!\perp_{\mathcal{G}'} Y \mid C$, producing a model more complex that $\mathcal{G}'$. Alternatively, the complexity added by having $Z \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$ would lead the algorithm to halt, adding no arc between C and Y, or to add Y as a child of C, either way introducing independence constraint $Y \perp\!\!\!\perp Z \mid C$ missing from $\mathcal{G}'$. Proceeding in a MC-DAG space, however, the same

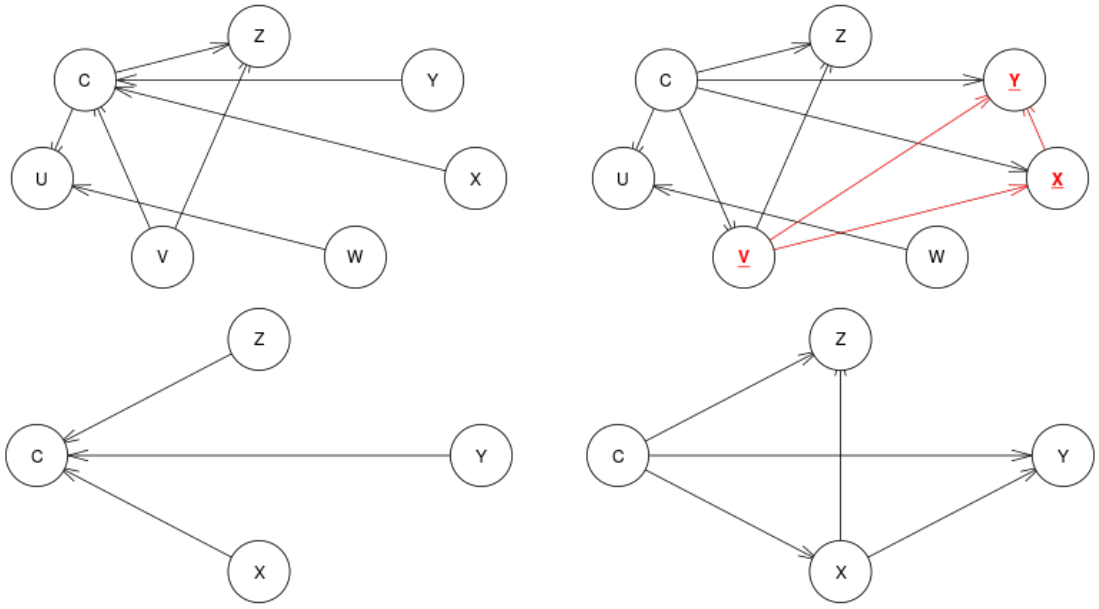

Figure 1: Above: A C-DAG $\mathcal{H}$ (left) and a classification-equivalent MC-DAG $\mathcal{G}$ (right). Only C and $\mathbf{Ch}_{\mathcal{H}}(C) = \{Z, U\}$ have parents in $\mathcal{H}$. In $\mathcal{G}$, all arcs incoming to C in $\mathcal{H}$ are reversed and there is an arc between every pair in $\mathbf{Pa}_{\mathcal{H}}(C)$ (shown in red). Only $\mathbf{Ch}_{\mathcal{G}}(C) = \{Z, Y, X, V, U\}$ have parents in $\mathcal{G}$. Below: An MC-DAG $\mathcal{G}'$ (right) and a classification-equivalent C-DAG $\mathcal{H}'$ (left) which lacks the constraint $Y \perp\!\!\!\perp Z \mid C$ present in $\mathcal{G}'$.

algorithm could have X and Y as children of C after two steps, and could recover the true structure in the following iterations.

4. Adapted GES algorithm for learning MC-DAGs

In this section we describe the adapted GES algorithm which visits only equivalence classes which contain MC-DAGs. We achieve this by means of additional operator validity conditions, discarding operators that produce CPDAGs outside this space. We begin by describing the properties of a CPDAG that has an MC-DAG consistent extension. We then provide the operator validity criteria.

4.1 CPDAGs for representing equivalence classes of MC-DAGs

The following conditions are needed for a CPDAG to have an MC-DAG as a consistent extension:

Proposition 6 *A CPDAG $\mathcal{P} = (\mathbf{V}, \mathbf{E}_{\mathcal{P}})$ has an MC-DAG as consistent extension if:*

1. $\forall X \to Y \in \mathbf{E}_{\mathcal{P}}$: $(Y \in \mathbf{Ch}(C))$ *or* $(X \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}, Y \in \{\mathbf{Nbr}_{\mathcal{P}}(C)\})$
2. $\forall X - Y \in \mathbf{E}_{\mathcal{P}}$: $X, Y \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$

Proof Assume that the conditions in Proposition 6 hold for a CPDAG $\mathcal{P}$. We need to find a $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$ such that every arc $X \to Y$ in $\mathcal{G}$ is such that $Y \in \mathbf{Ch}_{\mathcal{G}}(C)$. All arcs in $\mathcal{G}$ are either directed

or undirected in $\mathcal{P}$. Consider first an arc $X \to Y$ in $\mathcal{G}$ corresponding to an edge $X - Y$ in $\mathcal{P}$. By condition (2) in Proposition 6, $Y \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$. If $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$, the desired condition already holds in $\mathcal{G}$ for $X \to Y$. Otherwise, the arc $C - Y$ in $\mathcal{P}$ might be directed as $Y \to C$ in $\mathcal{G}$. $|\mathbf{Pa}_{\mathcal{G}}(C)| > 1$ may hold if every pair $Z, Q \in \mathbf{Pa}_{\mathcal{G}}(C)$ is adjacent in $\mathcal{G}$ and thus the $Z \to C \leftarrow Q$ are not v-structures. However, we will only prove the result for the case when $|\mathbf{Pa}_{\mathcal{G}}(C)| = 1$. With $|\mathbf{Pa}_{\mathcal{G}}(C)| = 1$, there is an equivalent DAG $\mathcal{G}'$ with $Y \in \mathbf{Ch}_{\mathcal{G}}(C)$, obtained by reversing $Y \to C$ in $\mathcal{G}$. $\mathcal{G}'$ has no cycles, because $\mathbf{Pa}_{\mathcal{G}}(C) = Y$ and thus there is no other path from Y to C in $\mathcal{G}'$. The reversal does not introduce any new v-structures into $\mathcal{G}'$ and thus does not modify its independence constraints. This is because reverting $C - Y$ as $C \to Y$ only adds v-structures $C \to Y \leftarrow Z$ for $Z \in \mathbf{Pa}_{\mathcal{G}'}(Y) \setminus \{C, \mathbf{Adj}_{\mathcal{G}}(C)\}$. Necessarily, $\mathbf{Pa}_{\mathcal{G}'}(Y) \subseteq \mathbf{Nbr}_{\mathcal{P}}(Y) \cup \mathbf{Pa}_{\mathcal{P}}(Y)$. For all $Z \in \mathbf{Nbr}_{\mathcal{P}}(Y)$ it follows from condition (2) in Proposition 6 that $Z \in \{$ C, $\mathbf{Ch}(C)$, $\mathbf{Nbr}(C)\}$. For all $Z \in \mathbf{Pa}_{\mathcal{P}}(Y)$, it follows from condition (1) that if $Z \notin \mathbf{Adj}_{\mathcal{P}}(C)$ then $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ and $C \to Y$ is already in the desired direction in $\mathcal{P}$. Otherwise, if $Z \in \mathbf{Adj}_{\mathcal{P}}(C)$ no v-structures are introduced by orienting $C - Y$ as $C \to Y$. Thus, for a $\mathcal{P}$ that satisfies conditions (1) and (2), there is a $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$ such that all undirected edges in $\mathcal{P}$ are directed into $\mathbf{Ch}_{\mathcal{G}}(C)$. Consider now the remaining case a directed arc $X \to Y$ in $\mathcal{P}$. By condition (1), if $X \notin \{C, \mathbf{Adj}_{\mathcal{P}}(C)\}$, then $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ and the condition is met. Otherwise, $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$. As dicussed for the case of an undirected arc $X - Y$, there is a $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$ such that $Y - C$ is directed as $C \to Y$. Thus, for a $\mathcal{P}$ complying with the conditions in Proposition 6 there is an MC-DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$. ■

4.2 Insert operators

For a current state $\mathcal{P}$, the Insert($X, Y, \mathbf{T}$) operator is:

Definition 7 (Chickering (2002b)) *Insert($X, Y, \mathbf{T}$)*
For non-adjacent nodes X and Y in $\mathcal{P}$, and for any subset $\mathbf{T}$ of the neighbors of Y that are not adjacent to X, the Insert($X, Y, \mathbf{T}$) operator modifies $\mathcal{P}$ by (1) inserting the directed edge $X \to Y$, and (2) for each $T \in \mathbf{T}$, directing the previously undirected edge between T and Y as $T \to Y$.

Chickering (2002b) defined validity conditions for Insert($X, Y, \mathbf{T}$) that can be checked on the CPDAG. The following additional conditions ensure that the operator can produce an MC-CPDAG.

Proposition 8 *Let $\mathcal{P}'$ be a PDAG obtained by applying a valid Insert($X, Y, \mathbf{T}$) to a CPDAG $\mathcal{P}$ that has a consistent extension $\mathcal{G}$ which is an MC-DAG. $\mathcal{P}'$ has a consistent extension $\mathcal{G}'$ which is an MC-DAG if $Y \in \{\mathbf{Ch}_{\mathcal{P}'}(C), \mathbf{Nbr}_{\mathcal{P}'}(C)\}$.*

Proof We need to find a DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$ that satisfies the conditions in Definition 3, that is, with each arc directed towards a child of the class variable. After applying Insert($X, Y, \mathbf{T}$), $\mathcal{P}'$ differs form $\mathcal{P}$ in that it contains the arc $X \to Y$ and that undirected edges $T - Y$ for $T \in \mathbf{T}$ have been oriented as $T \to Y$. These arcs will be identically oriented in every $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$. The remaining arcs in $\mathcal{G}'$ are either oriented as in $\mathcal{G}$, because they are compelled in $\mathcal{P}$, or can be oriented as in $\mathcal{G}$ because they are reversible in $\mathcal{P}$. Thus it suffices to show that, by proving $Y \in \mathbf{Ch}_{\mathcal{G}'}(C)$, the arcs $X \to Y$ and $T \to Y$, for every $T \in \mathbf{T}$, satisfy Definition 3. If $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ our condition is already satisfied because $Y \in \mathbf{Ch}_{\mathcal{G}'}(C)$ for any $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$. If $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$,

we need to prove that $C - Y$ can be oriented as $C \rightarrow Y$ in $\mathcal{G}'$ without modifying the independence constaints in $\mathcal{G}$. Orienting $C - Y$ as $C \rightarrow Y$ forms v-structures $C \rightarrow Y \leftarrow Z$ in $\mathcal{P}'$ for all $Z \in \{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C)$ as well as $C \rightarrow Y \leftarrow X$ if $X \notin \mathbf{Adj}_{\mathcal{P}'}(C)$. We first show that $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C) = \emptyset$. Namely, because $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$ and all $T \in \mathbf{T}$, by Definition 7, have incident undirected arcs, it follows, due to condition 2 in Proposition 6, that $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \subseteq \mathbf{Adj}_{\mathcal{P}'}(C)$ and thus $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C) = \emptyset$. If $X \notin \mathbf{Adj}_{\mathcal{P}'}(C)$, by condition (1) in Proposition 6, $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$, so $C \rightarrow Y$ is already properly oriented in $\mathcal{P}$. Therefore, enforcing in $\mathcal{G}$ the direction $C \rightarrow Y$ for $C - Y$ in $\mathcal{P}$ does not modify the independence constraints in $\mathcal{G}'$. Thus, for a $\mathcal{P}$ that matches the condition, the Insert$(X, Y, \mathbf{T})$ operator produces a CPDAG such that MC-DAG $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$. ■

4.3 Delete operators

For a current state $\mathcal{P}$, we apply the Delete$(X, Y, \mathbf{H})$ operator, followed by the Post-Delete(X, Y) if $X = C$. The operators are:

Definition 9 (Chickering (2002b)) *Delete$(X, Y, \mathbf{H})$*
For adjacent nodes X and Y in $\mathcal{P}$ connected either as $X - Y$ or $X \rightarrow Y$, and for any subset $\mathbf{H}$ of the neighbors of Y that are adjacent to X, the Delete$(X, Y, \mathbf{H})$ operator modifies $\mathcal{P}$ by deleting the edge between X and Y, and for each $H \in \mathbf{H}$, (1) directing the previously undirected edge between Y and H as $Y \rightarrow H$ and (2) directing any previously undirected edge between X and H as $X \rightarrow H$.

Definition 10 *Post-Delete(X, Y)*
Let $\mathcal{P}'$ be a PDAG obtained after applying a valid Delete$(X, Y, \mathbf{H})$ to $\mathcal{P}$. Then, if $X = C$, the Post-Delete(X, Y) operator deletes in $\mathcal{P}'$ all directed arcs $Z \rightarrow Y$, and orients all undirected edges $Z - Y$ as $Y \rightarrow Z$, for any $Z \in \mathbf{V}$.

The conditions given in Chickering (2002b) specify whether the Delete$(X, Y, \mathbf{H})$ operator is valid for a given X, Y, and $\mathbf{H}$. If $X = C$, the delete renders $Y \notin \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C)\}$. Thus, if it contains incoming or undirected edges into Y, $\mathcal{P}'$ does not satisfy the conditions in Proposition 6 that ensure there is an MC-DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$. Post-Delete$(X, Y)$ removes the violating arcs ensuring that $\mathcal{P}$ satisfies the conditions in Proposition 6.

Proposition 11 *Let PDAG $\mathcal{P}'$ be the result of applying a valid Delete$(X, Y, \mathbf{H})$, for $X = C$, followed by a Post-Delete(X, Y), to a PDAG $\mathcal{P}$. There exists a DAG $\mathcal{G}'$ that is a consistent extension of $\mathcal{P}'$ and is classification-equivalent to any MC-DAG $\mathcal{G}$ that is a consistent extension of $\mathcal{P}$.*

We only sketch the proof for brevity. Because Post-Delete(X, Y) ensured that $\mathcal{P}'$ matches the conditions in Proposition 6, it follows that is there is a DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$. Once Delete$(X, Y, \mathbf{H})$ rendered Y not adjacent to C, none of its incoming arcs affect classification-equivalence because they are not part of the minimal C-DAG subgraph of $\mathcal{P}$ (see Definition 2). Thus, we can delete them to obtain a PDAG with no parents for nodes that are no children of C. The undirected edges into Y would necesarily be oriented away from Y in a valid MC-DAG.

Proposition 12 *Let PDAG $\mathcal{P}'$ be the result of applying a valid Delete$(X, Y, \mathbf{H})$, followed by a valid Post-Delete(X, Y), to a CPDAG $\mathcal{P}$ that has a consistent extension $\mathcal{G}$ which is an MC-DAG. $\mathcal{P}'$ has a consistent extension $\mathcal{G}$ which is an MC-DAG if $C \notin \mathbf{H}$.*

Proof $\mathcal{G}'$ is an MC-DAG if all its arcs are directed towards a child of C. We need to ensure that: (1) If $Y \notin \{\mathbf{Ch}_{\mathcal{G}'}(C), \mathbf{Nbr}_{\mathcal{G}'}(C)\}$, then $\mathbf{Pa}_{\mathcal{G}'}(Y) = \emptyset$; (2) $H \in \mathbf{Ch}_{\mathcal{G}'}(C)$, for all $H \in \mathbf{H}$. $Y \notin \{\mathbf{Ch}_{\mathcal{G}'}(C), \mathbf{Nbr}_{\mathcal{G}'}(C)\}$ only happens if $X = C$. In that case, Post-Delete(X, Y) ensures that $\mathbf{Pa}_{\mathcal{G}'}(Y) = \emptyset$, satisfying the first condition. Regarding the second condition, $H \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$, for all $H \in \mathbf{H}$, by condition (2) in Proposition 6. If $C \neq H$, the second condition can be satisfied. Otherwise, $\mathcal{P}'$ will contain a v-structure $X \rightarrow C \leftarrow Y$. Thus for every $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$, $\mathbf{Pa}_{\mathcal{G}'}(C) \neq \emptyset$ and therefore such $\mathcal{G}'$ is not an MC-DAG. Thus, for $C \notin \mathbf{H}$, Delete$(X, Y, \mathbf{H})$ followed by Post-Delete(X, Y) produce a CPDAG that has an MC-DAG as a consistent extension. ■

5. Local discriminative scoring for CPDAGs

Chickering (2002b) specified how to efficiently score Insert$(X, Y, \mathbf{T})$ and Delete$(X, Y, \mathbf{H})$ operators for decomposable scores, without converting the current state's CPDAG into DAGs. We cannot do this for discriminative scores such as conditional log-likelihood, as 1) they do not decompose over the network; and (2) we need a fully parameterized DAG to compute the underlying class-conditional probability $P(c \mid \mathbf{x})$.

With complete data we can, however, efficiently update the $P(c, \mathbf{x})$ of the current state $\mathcal{P}$ for each c and each $\mathbf{x}$ in our data set $\mathcal{D}$, without recomputing it from scratch, and then recompute the $P(c \mid \mathbf{x})$ and the discriminative score from the updated $P(c, \mathbf{x})$. This technique was described by Keogh and Pazzani (2002) for single arc insertion into a one-dependence Bayesian classifier. We adapt it for CPDAG Insert$(X, Y, \mathbf{T})$ and Delete$(X, Y, \mathbf{H})$ operators for MC-DAGs.

Let A be the $N \times r_c$ matrix holding the $P_{\mathcal{G}}(c, \mathbf{x})$ for each c and each $\mathbf{x}$ in $\mathcal{D}$, where N is the number of training instances in $\mathcal{D}$ and r_c the number of distinct class values, $\mathcal{G}$ is any DAG $\in \mathbf{cext}(\mathcal{P})$ and $\mathcal{P}$ is our current CPDAG. A valid Insert$(X, Y, \mathbf{T})$ that yields a CPDAG $\mathcal{P}'$ is such that there is a $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$, equal to a $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$ plus the arc $X \rightarrow Y$. Since $P_{\mathcal{G}}(c, \mathbf{x})$ is identical for any $\mathcal{G} \in \mathbf{cext}(\mathcal{P}')$, we get $P_{\mathcal{G}'}(c, \mathbf{x})$ by updating A to account for the new arc,

$$a'_{ij} = a_{ij} \times \frac{P_{\mathcal{G}'}(y^{(i)} \mid x^{(i)}, \mathbf{t}^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)}{P_{\mathcal{G}}(y^{(i)} \mid \mathbf{t}^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)},$$

where the i superscript denotes the values in the i-th instance of $\mathcal{D}$ and $\mathbf{pa}^*_{\mathcal{G}}(X)$ denotes $\mathbf{Pa}_{\mathcal{G}}(C) \setminus C$. The above assumes that $C \in \mathbf{Pa}_{\mathcal{G}}(Y)$, which is ensured by Proposition 8. Because we only have access to $\mathcal{P}$, and not to $\mathcal{G}$, we do not know $\mathbf{Pa}_{\mathcal{G}}(Y)$. Yet, there exists an adequate $\mathcal{G}$ such that its parents are fully determined by $\mathcal{P}$ and the insert operator: $\mathbf{Pa}_{\mathcal{G}}(Y) = \{\mathbf{T}, C, \mathbf{Pa}_{\mathcal{P}}(Y)\}$. Such $\mathcal{G}$ exists because we can orient all undirected edges $Y - N$, $N \in \mathbf{Nbr}_{\mathcal{P}}(Y) \setminus \mathbf{T} \setminus C$, as outgoing from Y, because the inser operator may have only compelled them as outgoing from Y, by adding $\mathbf{T}$ and C as parents of Y, and not the other way around. To compute the update, we do not need $\mathcal{G}$ and $\mathcal{G}'$, but only the probabilities $P_{\mathcal{G}}(y^{(i)} \mid x^{(i)}, \mathbf{t}^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)$ and $P_{\mathcal{G}'}(y^{(i)} \mid \mathbf{pa}^*_{\mathcal{G}'}(y)^{(i)}, c_j)$ which we can re-estimate from $\mathcal{D}$ each time.

For a Delete$(X, Y, \mathbf{H})$, the update is analogous: $a'_{ij} = a_{ij} \times \frac{P_{\mathcal{G}'}(y^{(i)} | \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)}{P_{\mathcal{G}}(y^{(i)} | x^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)}$. A Post-Delete$(X, Y)$ does not change $P(c \mid \mathbf{x})$ so this suffices. Thus, for each operator, we update $P(C, \mathbf{x})$ in $O(Nr_c)$ time, independently of the number of features n.

6. Experimental evaluation

We compared our method (MG, short for MC-DAG GES) to the method by Acid et al. (2005) (abbreviated with A) and a number of *k-dependence Bayesian classifiers*, that is, augmented naive Bayes models with up to k predictor parents per predictor. Namely, we used the naive Bayes (DB^0, the 0 standing for 0-dependence), a 1-dependence classifier learned with a hill-climbing search and a discriminative score (DB^1_{HC}; Keogh and Pazzani, 2002) and its 2-dependence generalization (DB^2_{HC}) implemented in the bnclassify R package (Mihaljević et al., 2018), and the optimal log-likelihood 1-dependence classifier (DB^1_{CL}; Friedman et al., 1997). For comparison with the corresponding k-DB, we also limited the number of parents per predictor for MG to 1 (MG^1), 2 (MG^2), or left it unbounded (MG). We considered both an empty graph and a naive Bayes as the initial state of the search (the latter indicated with the NB subscript, e.g., MG_{NB}).

We used data sets from the UCI repository (Dheeru and Karra Taniskidou, 2017), together with the `mofn-3-7-10` dataset by Kohavi and John (1997) (see Table 1). We discretized numeric variables with the Fayyad and Irani (1993) method and removed instances with missing values, except in `voting`, where we treated them as a separate value. For all MG and DB_{HC} variants we used 10-fold stratified cross-validation accuracy estimate as the scoring function, with the greedy search proceeding as long as the score did not degrade. For the DB^1_{CL}, we used log-likelihood as the score.

MG_{NB} strongly outperformed all methods on `car` and `nursery` (see Table 1), and all methods but A on `mofn-3-7-10`. Both unbounded variants of MG (i.e., MG_{NB} and MG) were outperformed only on the mushroom data set, by DB^1_{CL}, the only non-greedy algorithm, and by A. Differences between the best unbounded MG and the remaining methods were not significant on the remaining data sets. The difference with DB^1_{CL} on mushroom is possibly due to the greedy nature of MG: DB^1_{CL} is able to find the optimal 1-DB with respect to log-likelihood, with 21 augmenting arcs, while MG_{NB} and MG added one and zero arcs, respectively, to their initial structures.

Traversing the space of equivalence classes, rather than that of DAGs, did not provide an advantage when bounding the number of predictor parents. Namely, MG^1_{NB} only outperformed DB^1_{HC} on `tictac`, with no additional significant differences between them, nor between MG^2_{NB} and DB^2_{HC}. MG_{NB} only visited slightly more distinct structures than DB_{HC}. For example, on a single run on `voting` MG^1_{NB} visited 968 states after ten iterations, versus 961 states by DB^1_{HC}. While the difference grows with the number of augmenting arcs included, it would still be negligible on `voting` with the maximal 16 of iterations.

MG_{NB} strongly outperformed MG on `car`, `nursery`, `tictac`, `crx`, `mofn-3-7-10`, and `kr-vs-kp` while MG was better on `iris` and `voting`. The poor performance of MG was due to it returning extremely sparse graphs: an empty graph for `mofn-3-7-10` and one with three arcs on `car`. On the other hand, MG was accurate on `voting` by using only four predictor variables, while MG_{NB} added 15 augmenting arcs and used all predictors.

7. Conclusion

We have specified the smallest DAG subspace that covers all possible class-conditional distributions. We presented an algorithm to traverse the equivalence classes in this space by adapting the greedy equivalence search algorithm. Finally, we specified how to compute the discriminative score of a

Table 1: Predictive accuracy estimated with 10-fold stratified cross-validation. N is the number of data instances, n the number of predictor variables, and r_c the number of distinct classes. A shows the accuracy for the method by Acid et al. (2005) reported in their paper.

	MG^1_{nb}	MG^2_{nb}	MG_{nb}	MG^1	MG^2	MG	DB^1_{hc}	DB^2_{hc}	DB^1_{cl}	DB^0	A	N	n	r_c
iris	0.93	0.94	0.91	0.9	0.91	0.94	0.94	0.93	0.93	0.94	**0.95**	150	4	3
car	0.95	0.94	**0.98**	0.86	0.86	0.86	0.94	0.95	0.94	0.86	0.93	1728	6	4
nursery	0.95	0.96	**0.97**	0.9	0.9	0.9	0.95	0.96	0.93	0.9	0.94	12960	8	5
breast	**0.98**	**0.98**	0.97	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.96	**0.98**	**0.98**	683	9	2
tictac	0.74	0.89	**0.90**	0.71	0.78	0.75	0.71	**0.90**	0.76	0.69		958	9	2
glass	0.74	**0.75**	0.74	0.74	0.74	0.74	**0.75**	0.72	0.74	0.74	0.73	214	9	6
mofn-3-7-10	0.94	0.94	**1.00**	0.86	0.86	0.86	0.93	0.93	0.93	0.86	**1.00**	1324	10	2
wine	**0.99**	**0.99**	0.98	**0.99**	**0.99**	**0.99**	0.98	**0.99**	0.98	**0.99**		178	13	3
crx	**0.87**	**0.87**	**0.87**	0.85	0.85	0.84	0.86	0.86	0.86	0.86	**0.87**	653	15	2
voting	0.91	0.9	0.91	**0.95**	0.94	**0.95**	0.91	0.9	0.94	0.9		435	16	2
tumor	**0.48**	0.46	0.47	**0.48**	**0.48**	**0.48**	0.47	0.45	0.38	**0.48**		132	17	18
lymphography	0.84	0.84	**0.86**	0.85	0.85	0.84	0.84	0.84	0.84	0.85	0.82	148	18	4
mushroom	0.98	0.98	0.98	0.97	0.97	0.97	0.99	0.98	**1.00**	0.97	**1.00**	5643	22	2
iono	0.92	0.92	0.91	0.92	0.92	0.92	0.92	0.92	**0.93**	0.92		351	34	2
dermatology	0.98	0.98	0.98	0.98	0.98	0.98	**0.99**	0.98	0.97	0.98		358	34	6
soybean	0.91	0.92	0.92	0.91	0.91	0.91	0.91	0.92	**0.93**	0.91	0.9	562	35	19
kr-vs-kp	0.96	0.97	**0.98**	0.88	0.88	0.88	0.96	0.97	0.92	0.88	0.97	3196	36	2
molecular	0.88	0.9	**0.92**	0.91	0.91	0.91	0.9	0.9	0.81	0.91		106	57	2

CPDAG search operator in a time that is independent of the number of variables and that does not require converting the CPDAG into a DAG.

Future work includes evaluating our algorithm on additional synthetic and real-world data sets, to better assess its merits. It is possible that a different search algorithm could take better advantage of equivalence classes when the nodes' in-degree is bounded, as for the k-dependence models. Adapting our algorithm to traverse the space of augmented naive Bayes models, rather than that of MC-DAGs, would amount to simple additional restrictions to operator validity conditions.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 785907 (HBP SGA2), the Spanish Ministry of Economy, Industry and Competitiveness through the Cajal Blue Brain (C080020-09; the Spanish partner of the EPFL Blue Brain initiative) and TIN2016-79684-P projects, the Regional Government of Madrid through the S2013/ICE-2845-CASI-CAM-CM project, and Fundación BBVA grants to Scientific Research Teams in Big Data 2016.

References

S. Acid and L. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

S. Acid, L. de Campos, and J. G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

C. Bielza and P. Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1), 2014.

D. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002a.

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

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan Kaufmann, 1993.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29: 131–163, 1997.

E. J. Keogh and M. J. Pazzani. Learning the structure of augmented Bayesian classifiers. *International Journal on Artificial Intelligence Tools*, 11(4):587–601, 2002.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1): 273–324, 1997.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, Cambridge, MA, USA, 2009.

B. Mihaljević, C. Bielza, and P. Larrañaga. *bnclassify: Learning Discrete Bayesian Network Classifiers from Data*, 2018. URL `https://CRAN.R-project.org/package=bnclassify`. R package version 0.4.0.

M. Minsky. Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, 49: 8–30, 1961.

M. Pazzani. Constructive induction of Cartesian product attributes. In *Proceedings of the Information, Statistics and Induction in Science Conference*, pages 66–77, 1996.

N. A. Zaidi, J. Cerquides, M. J. Carman, and G. I. Webb. Alleviating naive Bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 14:1947–1988, 2013.