Homework

Leo Casarsa

October 2015

1 Causal BayesDB

> Let's assume that the hypothesis class is some family of Bayesian Networks, $\mathcal{B}^*$. How to implement `SIMULATE`, `ESTIMATE` and `INFER`? Propose candidate implementations based on the talk.

We consider p random variables $X_1, \ldots, X_p$ whose distribution is Markov and faithful with respect to the underlying causal DAG. We assume that all random variable are observed, not dealing with the problem of hidden confounders in this first attempt. We denote $\mathcal{X} = \{X_1, \ldots, X_p\}$ to represent the set of random variables of interest.

The hypothesis class $\mathcal{B}^*$ can be formalized by a prior over Bayes networks, $P(B)$, which assigns positive probability mass only to the Bayes nets belonging to $\mathcal{B}^*$. Assume that this prior is given and that it can be factorized into a prior over graph structures and a prior on the parametrization of the Bayes nets given the structure, i.e., $P(B) = P(G, \Theta) = P(G)P(\Theta|G)$. Additionally, assume that the data likelihood induced by the parametrization $P(X|G, \Theta)$ is also given. We consider that it is possible to sample from and to evaluate the given distributions.

Let D be the observed data points, the first step will consist on updating the beliefs over the Bayes net, by assessing the posterior $P(B|D)$[1]. Assessing the posterior will be done through sampling methods, usually organized around Markov chain Monte Carlo procedures. In Venture, the following pseudo code corresponds to sampling a graph from the marginalized posterior $P(G|D)$,

```
(assume g (graph_prior hyperparams))
(assume theta (param_prior g))
(assume X (data_like theta g))
(observe X data)
(infer posterior)
(sample g)
```

where we assume that the models for the graph prior, for parametrization prior and for the data likelihood are provided, and we must only set appropriate

[1] Short notation for $P(B|x = D)$

values for the hyperparameters. The function `data_likelihood` returns a list of p likelihoods, one for each variable.[2]

1.1 ESTIMATE

The command `ESTIMATE` in BayesDB allows columns of a database to be selected, instead of rows. `ESTIMATE PAIRWISE` calls the specified pairwise function on each pair (of rows or columns) in the dataset. In the Strange Loop talk, `ESTIMATE` was used in the sense of `ESTIMATE PAIRWISE DEPENDENCE PROBABILITY`, therefore, we will focus on this functionality.

`ESTIMATE PAIRWISE DEPENDENCE PROBABILITY` estimates the probability that two variables X_i and X_j are dependent, for each pair $\{X_i, X_j\}$ in the dataset. In a Bayes net, two variables are dependent if, and only if,[3] X_i and X_j are not d-separated by the null set. A straightforward way of estimating the pairwise dependence probability between X_i and X_j is to sample graphs from the graph posterior $P(G|D)$ and to count the proportion of sampled graphs in which X_i and X_j are not d-separated.

For the following pseudocode we start from a model of the graph posterior, which was derived in the previous section, and we apply the function `d_connected`, which takes a DAG as input and returns a matrix M of binary values whose entry $M_{i,j}$ is 1 if, and only if, nodes X_i and X_j are not d-separated. We then sample n times from M, stack the samples in a 3D-tensor T, sum the sampled values along the third dimension of the tensor using the function `sum_three`, and divide each element by n using the function `pointwise_divide`.

```
(assume g_post (graph_posterior data))
(assume M (d_connected g_post)
(define T (empty))
(infer repeat(n
    do (pass
        (bind (collect M) (curry into T)))))
(define DP (pointwise_divide (sum_three T) n)))
```

The output `DP` is a symmetric matrix whose entries are the estimated dependence probabilities between pairs of variables.[4]

1.2 INFER

`INFER` fills in a missing value of the database with a point estimate over its predictive distribution $P(x'|D)$ given the observed data D, marginalized out

[2] The `data_likelihood` function samples from $P(X_i|\mathrm{pa}_G(X_i), \Theta)$, if the parents of X_i had already been sampled; otherwise, it samples from the parents of X_i (and memoize them).

[3] If the Markov and faithfulness conditions are fulfilled. Faithfulness can be eventually violated for particular choices of Θ. However, this should be an unlikely scenario.

[4] I realize that in Venture, using matrices and tensors might be a little more complicated, (being lists of lists of lists). Additionally, I have not given much thought on whether the functions `d_connected`, `sum_three` and `pointwise_divide` are straightforward to implement in Venture, but they should be easily implementable in C, Python or Matlab. However, I will not focus on these issues right now.

models, if the confidence of the point estimate is higher than a specified threshold, σ. The point estimate is often the mode, median, or mean, depending on the datatype.

Sampling from the predictive distribution can be thought as sampling from the data likelihood conditioned on the posterior models and then marginalizing over the models. In Venture, we first sample from the posterior of the graph and of the parametrization given the data, similar to section 1.

The following pseudocode takes as input the confidence tolerance `sigma` and outputs a list of point estimates `E`. The procedure first uses function `missing_values` to identify missing values in the data points, then, for each missing value, it produces n samples from the predictive distribution. Point estimates for the missing values are produced using the user's estimator of choice (e.g., mean) and stored in list `E`. Finally, we compute the sample frequency of the point estimates and remove from our list `E` the values which appear less often as the given tolerance `sigma`.[5]

```
(define miss (missing_values data))
(assume g_post (graph_posterior data))
(assume theta_post (param_posterior data))
(assume X_pred (data_like theta_post g_post)))
(define pred_samples (empty))
(infer repeat(n
    do (pass
       (bind (collect X_pred) (curry into pred_samples)))))
(define E (estimator X_pred miss))
(define PE (frequency E X_pred))
(python: E[PE < sigma] = float('nan'))
```

The last line was written in Python syntax, in which is easier to do logical indexing.

1.3 SIMULATE

`SIMULATE` generates new data points according to the underlying models of the data, i.e., it produces samples from the conditional predictive distribution $P(x'|D, Q)$, where Q is a set of queries to condition from.

In Venture, `SIMULATE` behaves in a very similar way to `INFER`, with the difference that the data is sampled from the conditional predictive distribution. This is done through the function `cond_data_like`, which is similar to `data_like`, but accepts an extra argument `queries` as input, and samples from the data likelihood conditioned on the queries.

[5] This is the implementation assuming discrete data values. For continuous data values, we could do a similar procedure but computing a kernel density estimator instead of sample frequency.

```
(define miss (missing_values data))
(assume g_post (graph_posterior data))
(assume theta_post (param_posterior data))
(assume X_sim (cond_data_like theta_post g_post queries)))
(sample X_sim)
```

1.4 Ideas

Depending on the data likelihood, searching through the whole space of graphs is redundant. For instance, in the linear Gaussian case, one cannot distinguish between Markov equivalent graphs, therefore it would be better to restrict the hypothesis class to graphs from the same equivalent class, i.e., CPDAGs.

This is just a thought, but it seems that we can recover in this framework all the Structural Equation Model-based causal discovery methods (such as additive noise models)[5], by picking the appropriate data likelihood. This would allow a full Bayesian treatment of these methods, which, it seems, do not seem very flexible to incorporating further assumptions on the data. I was talking to Jonas Peters here at the lab and he told me that people apparently did not address even the nonlinear-gaussian models in a Bayesian setting yet.

2 Black Box

> Assume one has a black box that receives as inputs sets of variables, among which there can be interventions. The black box has an interface to the true data generator and can sample from, condition on any query and compute conditional mutual information on any subset of variables. How to formalize soft interventions in this scenario? Construct two or three Bayes Nets with variables which perform soft interventions.

We consider p random variables $X_1, \ldots, X_p$ and (at most) p intervention variables $I_1, \ldots, I_p$, whose distribution is Markov and faithful with respect to the underlying causal DAG G. We assume that all variables are observed, not dealing with the problem of hidden confounders in this first attempt. We denote $\mathcal{X} = \{X_1, \ldots, X_p\}$ the set of random variables of interest, $\mathcal{I} = \{I_1, \ldots, I_p\}$ the set of intervention variables, and $\mathcal{V} = \mathcal{X} \cup \mathcal{I}$ the set of observed variables. We have only access to $\mathcal{V}$, and the task is to distinguish $\mathcal{I}$ from $\mathcal{X}$.

In a causal Bayesian Network, an intervention is represented as an exogenous binary variable I_j with a single arrow in G, directed at the variable X_j it manipulates. If all intervention variables are *off*, say $I_j = 0$ for all $I_j \in \mathcal{I}$, we recover the passive observational distribution over $\mathcal{X}$, therefore $P(\mathcal{X}|\mathcal{I}_0) = P(\mathcal{X})$ [2]. If one of the intervention variables is *on*, say $I_j = 1$, there are two possible scenarios. In the first scenario the intervention is *hard*,[6] and X_j becomes independent of its causes, or parents in G,

$$P(X_j|\mathrm{pa}_G(X_j), I_j = 1) = P(X_j|I_j = 1). \tag{1}$$

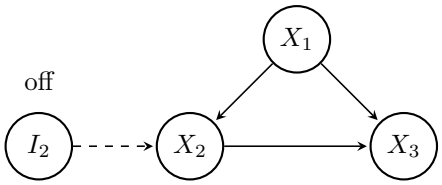
In the second scenario, the intervention is *soft*,[7] and the causal structure between X_j and its parents is not altered, but the dependency is,[8]

$$P(X_j|\mathrm{pa}_G(X_j)) \neq P(X_j|\mathrm{pa}_G(X_j), I_j = 1). \tag{2}$$

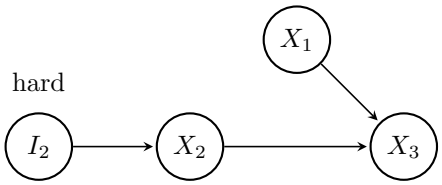
[6] Also known as "structural", "surgical", "ideal", or "independent".

[7] Also known as "parametric" or "dependent".

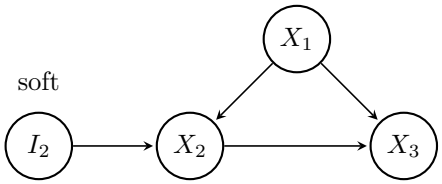
[8] Korb et al. [3] define two dimensions to characterize interventions, from dependent (soft) to independent (hard), and from deterministic to stochastic, i.e., that manipulates the target variable toward a specific value or a distribution of values, respectively. We will only deal with the two extremes of the first dimension (soft and hard), and will not assume anything about the second dimension here (we can treat the intervention as always stochastic).



$$P(X_1, X_2, X_3, I_2 = 0) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)$$



$$P(X_1, X_2, X_3, I_2 = \text{hard}) = P(X_1)P(X_2|I_2)P(X_3|X_1, X_2)$$



$$P(X_1, X_2, X_3, I_2 = \text{soft}) = P(X_1)P(X_2|X_1, I_2)P(X_3|X_1, X_2)$$

Figure 1: Examples of DAGs with interventions. The topmost DAG is equivalent to an observational DAG, since the intervention is off. The middle DAG has a hard intervention, which breaks the links between parents and the manipulated variable. A classical example of hard intervention are randomized experiments, in which X_2 will be sampled according to some distribution. The bottommost DAG presents a soft intervention, which may change the conditional independence of X_2 given its parents, while maintaining the causal structure intact. Notice the unshielded collider between variables $\{X_1, X_2, I_2\}$ An example of a soft intervention is to add a linear factor to X_2, e.g., $P(x_2|X_1, I_2) = P(x_2 - \Delta|X_1)$.

2.1 Identifying interventions

The first step in the detection of interventions is the identification of (binary)[9] single-arrow variables in G. If V_j is single-arrow and there is an arrow in G between $V_j, V_k \in \mathcal{V}$,

1. $V_j \not\perp\!\!\!\perp V_k$
2. V_j and $\mathcal{V} \setminus \{V_j, V_k\}$ are d-separated by V_k, and, by the Markov condition, $V_j \perp\!\!\!\perp \mathcal{V} \setminus \{V_j, V_k\} | V_k$.

Faithfulness guarantees that the variables that satisfy the previous conditional independence conditions are single-arrow. Since the black box has an interface for estimating conditional mutual information (MI) between pairs of variables, we can make a procedure for identifying single-arrow variables,

```
1: procedure SINGLE-ARROW IDENTIFICATION
2:     for i, j = 1, ..., p and i ≠ j do
3:         if MI(V_i, V_j) > 0 then
4:             for k = 1, ..., p and k ≠ i, j do
5:                 Compute MI(V_i, V_k|V_j)
6:             if MI(V_i, V_k|V_j) = 0 for all k ≠ i, j then
7:                 V_i is single-arrow with relation to V_j
```

Once all the single-arrow variables and their connections are detected, we must determine whether they are hard, soft or no interventions.

2.1.1 Hard Interventions

Since hard interventions block the links between the manipulated variable and its parents, identifying them is a straightforward comparison of mutual information conditioned on the value of the single-arrow variable. Concisely, we want to find the single-arrow V_i wrt. V_j, such that, for any $k = 1, \ldots, p$ with $k \neq i, j$, it is observed $\mathrm{MI}(V_i, V_k | V_j = 1) \neq \mathrm{MI}(V_i, V_k | V_j = 0)$, with exactly one of the conditioned mutual informations being zero.

2.1.2 Soft Interventions

Soft interventions only change the conditional distribution of a variable given its parents, therefore a different method will be necessary to identify them. We so far have not exploited the fact the intervention variables have arrows directed at the manipulated variable. Let again V_i be a single-arrow variable wrt. V_j, then if V_j has another parent V_k, the three variables form an unshielded collider. Constraint-based methods for causal discovery are able to identify unshielded colliders by observing the following conditional independence relations,

[9] The identification of binary variables is trivial and, while it does simplify the problem, it is easy to extend the following procedures to non-binary interventions.

1. $V_i \perp\!\!\!\perp V_k$, (or $\mathrm{MI}(V_i, V_j) = 0$)
2. $V_i \not\perp\!\!\!\perp V_k | V_j$ (or $\mathrm{MI}(V_i, V_k | V_j) > 0$).

Notice that our procedure will only identify interventions on variables that have another ancestor. Otherwise, it becomes a two-variable causal inference problem, which is only tractable with stronger assumptions.

References

[1] Jay Baxter. *BayesDB: Querying the Probable Implications of Tabular Data*. PhD thesis, Massachusetts Institute of Technology, 2014.

[2] Frederick Eberhardt and Richard Scheines. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007.

[3] Kevin B Korb, Lucas R Hope, Ann E Nicholson, and Karl Axnick. Varieties of causal intervention. In *PRICAI 2004: Trends in Artificial Intelligence*, pages 322–331. Springer, 2004.

[4] Vikash Mansinghka, Charles Kemp, Thomas Griffiths, and Joshua Tenenbaum. Structured priors for structure learning. *arXiv preprint arXiv:1206.6852*, 2012.

[5] Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *arXiv.org preprint*, arXiv:1412.3773 [cs.LG], December 2014. Submitted to Journal of Machine Learning Research.