Learning Causal Networks in Python

James Callan

1 Introduction

The aim of this report is to detail the implementation of various algorithms to learn causal networks in python.These algorithms have already been implemented in the statistical programming environment R. However, recently python has become more widely used in the field of statistical computation.

Python offers a number of advantages over R. Python is much more widely used in general than R so an implementation in python would be accessible to a wider array of developers.Python is also used in a variety of settings as it is a general purpose programming language. This would allow the software implementation to be integrated into many different systems, written in python, with relative ease.

2 Literature Review

2.1 Causal Networks

Causal Networks are graphical models which represent a set of variables, their conditional dependencies, and their causal relationships [1] as a Directed Acyclic Graph (DAG) or a Maximal Ancestral Graph (MAG). The nodes of the Graph represent the variables. The edges of the graph represent causality, the direction of the edge represents the direction of causality with parent nodes causing child nodes [1].

In a DAG all edges have a single direction, whereas in a MAG edges may be directed or bidirectional [2].

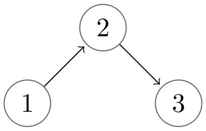

Figure 1 A DAG with 3 nodes and 2 edges

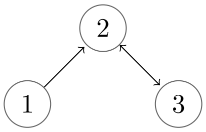

Figure 2 A MAG with 3 nodes and 2 edges

The learning of causal networks allows relationships between variables to be uncovered and presented in a simple and human readable fashion. This can provide useful information for further data analysis as the set of variables that cause another could be used to predict it's value.

2.2 Probability and Independence

In probability theory we can quantify our confidence in a particular event E occurring. This confidence, denoted as $P(E)$ is a real number between 1 and 0. $P(E) = 1$ representing a certainty and $P(E) = 0$ representing E being impossible.

In the case of random variables an event would be a variable X taking a particular value x_i, where x_i is a member of the set $x = \{x_1, x_2, ..., x_n\}$ of possible values X can take. The probability of this event is denoted as $P(X = x_i)$.

The probability distribution $P(X = x)$ or simply $P(X)$ defines the probabilities for every value which X can take.

Distributions of more than one variable can be described with joint distributions. For Variables X and Y the distribution $P(X, Y)$ describes the probability of both X and Y taking particular values simultaneously.

Joint distributions can also describe how the value of one variable can effect the value of an other. $P(X|Y)$ describes the probabilities of X taking particular values "given" a value that Y has taken. $P(X|Y)$ is defined as $P(X, Y)/P(Y)$.

If two variables are independent there is no correlation between the values they take. That is for two variables X and Y, the distribution $P(X|Y)$ would be the same for all values of Y. Therefore $P(X|Y) = P(X)$. If two variables are independent we can determine that there is no causal relationship between them.

Given the definition $P(X|Y) = P(X, Y)/P(Y)$ and $P(X|Y) = P(X)$ when X and Y are independent, it easy to see that $P(X, Y) = P(X)P(Y)$ if X and Y are independent.

Two variables X and Y can be considered independent conditioned on a third variable Z if there is no correlation between X and Y for given the value of Z.

In this case $P(X|Y, Z) = P(X|Z)$, as the value of Y has no impact on the value of X. The definition of conditional distributions shows $P(X|Y, Z) = \frac{P(X,Y,Z)}{P(Y,Z)}$, and $P(Y, Z) = P(Y|Z)P(Z)$. Using theses definitions and basic algebra $P(X, Y|Z) = P(X|Z)P(Y|Z)$ can be shown when X and Y are conditionally independent on Z.

2.3 D Separation

A path is any sequence of adjacent edges regardless of their directionality. A collider is node in a path which is both entered and left on edges which are directed toward the node. Unblocked refers to a path that does not traverse a collider [3].

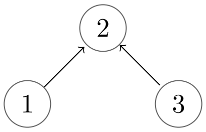

Figure 3 A graph containing a collider

If every path between nodes X and Y traverses a collider, nodes X and Y are unconditionally d-separated or d-separated conditioned on the empty set [3]. Two unconditionally d-separated nodes in a causal network are considered to be independent [4].

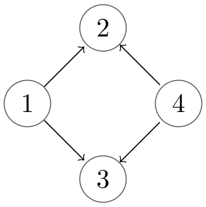

Figure 4

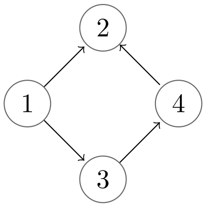

Figure 5

Two nodes X and Y are d-separated conditioned on set Z if:

1. There exists no unblocked path from X to Y that does not traverse any members of Z. [3]

2. There is no blocked path from X to Y in which all colliders are members of Z or have descendants in Z. [3]

Causal networks represent conditional independence with d-separation, variables X and Y are conditionally independent on set Z if nodes X and Y are d-separated conditioned on nodes in set Z [1].

2.4 Faithfulness

By testing for d-separation of variables X and Y given conditioning set Z on a graph we can see if $X \perp\!\!\!\perp Y|Z$ [4]. However, the reverse is only true if the distribution is faithful to the graph. That is, if a distribution is faithful to a graph then all independence relationships are represented on the graph and only these relationships are present [5]. Assuming faithfulness allows the topology of a graph to be learned by testing pairs of variables for independence given various conditioning sets.

2.5 Discriminating paths

For the FCI and RFCI algorithms the notion of discriminating paths is needed. A path π = (A, .., X,Y,Z) is discriminating a discriminating path for Y if:

1. π must include at least 3 edges
2. Y is a non-endpoint on π and is adjacent to Z on π.
3. A is not adjacent to Z and every other node is a collider on π and a parent of Z. [6]

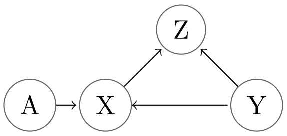

Figure 6
(A,X,Y,Z) is a discriminating path on Y

2.6 Tests for Independence

When trying to find causal relations between variables tests for independence and conditional independence must be performed. These tests allow us to find correlation between variables and also find out if the correlation between two variables is actually caused by another.

One such test is the χ^2 test, this test gives the probability that some data was drawn from a particular distribution.

If the variables X and Y are independent then $p(X,Y) = p(X)p(Y)$. Both of these distributions can be calculated from data. The χ^2 test can then be used to find the probability that they are equal.

To estimate $p(X,Y)$ the counts of all pairs of values taken by the variables can be found. They can then divided by the count of all data points. These counts can be stored in a table with the values of X as columns, values of Y as rows (or visa versa) and entries as observed counts, for ease of access. This type of table is known as a contingency table.

X	Y
1	0
1	1
0	1
1	1
0	0
1	1
1	0

		X	
		0	1
Y	0	1	2
	1	1	3

To estimate the distribution $p(X)$, the count of points in which X takes a value over the total number of points for all values of X. The counts can be found with the sum of the columns of the contingency table. The same can be done for $p(Y)$ with the rows of the table.

The χ^2 statistic can then be calculated with:

$$\chi^2 = \sum \frac{(p(X)p(Y)-p(X,Y))^2}{p(X)p(Y)}$$

The χ^2 value can then be passed to a cumulative density function and a p-value returned. The function used is dependent upon the degrees of freedom. The degrees of freedom are equal to the $(no.rows - 1) * (no.columns - 1)$. The p-value represents how likely it is that the two variables are independent.

The test is similar for conditional independence. The chi^2 statistic now depends on conditional distributions. If $X \perp\!\!\!\perp Y|Z$ $p(X,Y|Z) = p(X|Z)p(Y|Z)$. Therefore:

$$\chi^2 = \sum \frac{(p(X|Z)p(Y|Z)-p(X,Y|Z))^2}{p(X|Z)p(Y|Z)}$$

These values can all be found by calculating a contingency table for $X, YandZ$.

This statistic can then be used in the same way as before to calculate a p-value. However, the degrees of freedom are multiplied by the possible values of Z -1.

2.7 Algorithms For Learning Causal Networks

2.7.1 Constraint Based Methods

There have been a number of algorithms developed to learn graphical causal representations from data. Over the course of this project, I plan to implement: The PC algorithm [7], the Fast Causal Inference (FCI) algorithm [6], and the

Really Fast Causal Inference (RFCI) Algorithm [6].

Rather than learning an individual DAG the PC algorithm learns a set of DAGs which could represent the causal relationships found in the data. In cases where the direction of an edge is unclear no direction is specified. The output of the algorithm is a partially directed acyclic graph which represents the set DAGs with all possible orientations of undirected edges [7].

The FCI and RFCI algorithms learn partial ancestral graphs (PAGs) which represent the set of MAGs that fit the data [6]. Unlike in the PC algorithm if the direction of an edge can not be determined it is assumed that there is some set of latent variables influencing the nodes being tested. Each end of an edge can be one of three things in a PAG, an arrow in every MAG ($>$), a tail in every MAG ($-$) or an arrow in at least one MAG and a tail in at least one MAG (o), ends of an edge denoted by $*$ can be any of the specified types. This allows for a richer display of information than a partial DAG [6].

The PAG generated by the FCI algorithm is considered to be less accurate than the PAG generated by the RFCI algorithm. This is because the FCI algorithm performs more tests than the RFCI algorithm. These tests give us more confidence in the presence and the orientation of edges however this comes at the cost of speed.

2.7.2 Score Based Methods

The above algorithms are all constraint based. There is another class of algorithm which are score based. Rather than finding graphs using the rules that define the networks, score based methods generate random graphs and assign them a score based on how well they fit data.

An example of a scored based method is the graphical lasso algorithm.

2.8 Computing a skeleton

All three algorithms begin by calculating the skeleton of the graph. The skeleton consists of the edges that are present in the partial DAG or PAG however all edges are undirected. The separation set of two variables ($sepSet(x, y)$) contains the smallest set of variables that the pair are conditionally independent on. When calculating the skeleton the separation set of each pair of variables is also recorded as it is needed to orient the edges of the graph [6, 7].

The skeleton is calculated using a series of conditional independence tests. These conditional independence tests are dependent on the type of input data, therefore conditional independence tests can be defined independently of the algorithm. The conditional independence tests must be able to determine if two variables are indpendent based on some set of other variables from data.

To begin the skeleton computation the fully connected undirected graph of all variables is constructed and then conditional independence tests are performed to determine which edges to remove.

Adjacent pairs of variables are tested for independence conditioned upon set of variables adjacent to them. The size of these sets starts at 0 but is incremented after all adjacent pairs have been tested. This repeats until there are not enough adjacent variables to fill the conditioning set.

The edges which have not been removed form the skeleton of the graph which is needed for the PC, FCI and RFCI algorithms along with the recorded separation sets of removed edges [6, 7].

```
Algorithm to Estimate Skeleton
Input:    Data, Independence Test
Output:   Estimated Skeleton, Separation Sets of Variables

k = 0
N = list of variables in data
G = fully connected undirected graph with nodes = N

while there is some $S \in adj(X, G) \setminus Y$ where $|S| = k$ for some X in N and Y in adj(X,G)
  for all X in N:
    for all Y in adj(X, G):
      for all $S \in adj(X, G) \setminus Y$ where $|S| = k$:
        if $X \perp\!\!\!\perp Y \mid S$
          remove edge X,Y from G
          sepSet(X,Y) = S
          break
        end if
      end for
    end for
  end for
  k += 1
end while

return G, sepSet
```

2.9 The PC Algorithm

After generating the skeleton and separation sets of pairs of nodes, the pc algorithm attempts to orient the undirected edges of the graph.

It begins by searching for colliders. For each triple of vertices A, B, and C where A and B are adjacent, B and C are adjacent, but A and C are not adjacent orient the edges A-B-C to A→B←C if B is not in the separation set of the pair (A,C).

Once colliders are specified other edges can be oriented. This is done by repeating two steps until no more edges can be given a direction. The algorithm for edge orientation is described in.

```
PC Algorithm to Orient Edges
Input:   Skeleton of Causal Network, Separation sets of variables
Output:  Partial Directed Acyclic Graph

G = Skeleton of network
N = nodes of G

for all $i, j, k$ in N:
  if $k$ not in $adj(i, G)$ and $j$ in $sepSet(i, k)$:
    orient $i - j - k$ as $i \rightarrow j \leftarrow k$
  end if
end for

while no more edges can be oriented:
  for all $i, j, k$ in N:
    if $k$ not in $adj(i, G)$:
      orient $i \rightarrow j - k$ as $i \rightarrow j \rightarrow k$
    end if
    if $directedpath(i, j)$ in $G$:
      orient $i - j$ as $i \rightarrow j$
    end if
  end for
end while

return G
```

2.10 The FCI algorithm

The FCI algorithm starts in the same way as the PC algorithm using exactly the process of conditional independence tests to estimate a skeleton of the final graph. It also orients colliders in the same way, producing a PDAG.

Next, the final skeleton of the graph is obtained from the PDAG G. To do this The possible d-separators of every variable must be found in G. For variable X, the possible d separators set in nG ($PossibleDSep(X, G)$) consists of all variables Y for which there is some path $\pi = path(X, Y)$ in G, such that in every sub path A, B, C, B is a collider or A, B, C form a triangle. All adjacent variables (X, Y) are the tested for conditional independence, conditioned on every subset of each of the possible d separators set. If the test finds conditional independence the edge X, Y is removed and the conditioning set recorded as a separation set.

The PDAG can then be converted to a PAG by setting all edges to $o - o$.

Next the colliders must be reoriented. This is done on a similar way however now edges can be bi-directional.

Once the final skeleton has been estimated, A series of orientation rules can be performed repeatedly until no more orientations can be found. Theses rules where originally described in. However these rules did not fully orient the $-$ tags of the graph [8]. 10 rules were found in [8] which allow full orientation of the graph. The complete set of rules is used in the R implementation and will also be used in the python implementation.

```
FCI Algorithm to Orient Edges
Input:    Skeleton of Causal Network, Separation sets of variables
Output:   Partial Ancestral Graph

G = Skeleton of network
N = nodes of G

for all $i, j, k$ in N:
  if $k$ not in $adj(i, G)$ and $j$ in $sepSet(i, k)$:
    orient $i - j - k$ as $i \rightarrow j \leftarrow k$
  end if
end for

for all i in N and j in adj(i,G):
  for all $T \subseteq PossibleDSep(i, G) \setminus \{i, j\}$:
    if $i \perp\!\!\!\perp j | T$:
      remove edge i,j from G
      $sepSet(i, j) = T$
      $sepSet(j, i) = T$
      break
    end if
  end for
  for all $T \subseteq PossibleDSep(j, G) \setminus \{i, j\}$:
        if $i \perp\!\!\!\perp j|T$:
            remove edge i,j from G
            $sepSet(i, j) = T$
            $sepSet(j, i) = T$
            break
        end if
    end for
end for

set all edges in G to o-o

for all $i, j, k$ in N:
    if $k$ not in $adj(i, G)$ and $j$ in $sepSet(i, k)$:
        orient $i * - * j * - * k$ as $i* \to j \leftarrow *k$
    end if
end for

while no more edges can be oriented:
    Apply orientation rules from [8]:
end while

return G
```

2.11 RFCI Algorithm

Like the other two algorithms the RFCI begins by estimating the skeleton of the final graph.

Next the colliders in the graph are found, this is done differently than in the two previous algorithms. We created and empty list L and for each triple (A,B,C) in the list of unshielded triples we check if both:

i $A \perp\!\!\!\perp B \mid ((sepset(A, B) \setminus \{C\}))$

ii $B \perp\!\!\!\perp C \mid ((sepset(A, B) \setminus \{C\}))$

where sepset(X,Y) is the separating set of Nodes X and Y. If both of these tests hold, add to list L. If only the first test holds, find the minimal separating set of A and B and record it as the separating set of A and B. Next, remove the edge A-B, add any newly create unshielded triples to M, and remove any destroyed triples from L and M. If only the second test passes do the same as above but for nodes B and C.

Finally the edges must be oriented. To begin edge orientation apply rules $R1-R3$ found in [8]. Next triangles (X,Y,Z) must be found with edges $Y o - *Z$,

$X <-*Y$, and $X -- > Z$. Once a triangle is found the shortest discriminating path from X to Z must be found.

If a path π exists take each pair of adjacent variables A and B on π and creator a counter $l = 0$. Next find a subset Y $\subseteq$ $(sepset(X, Y) \setminus \{A, B\})$ where $|Y| = l$. Then if $A \perp\!\!\!\perp B|Y \cup S$ let sepset(A,B) = Y, delete the edge $A * - * B$ from the graph and update newly formed unshielded triples orientation using the collider orientation described earlier. If the conditional independence test fails increment l and try again, testing with all subsets Y of size l, repeat this process until no set Y exists where —Y— = l or the edge $A * - * B$ is removed.

If no edges are removed from π by the above process then either: orient $Yo-*Z$ as $Y - - > Z$ if Y is in the sperating set of X and Z or orient $X < - * Yo - *Z$ as $X < - > Y < - > Z$ if not.

Finally orient as many edges using rules $R5 - R10$ from [8].

This whole process must be repeated until it no longer alters any orientations. [6]

References

[1] Thomas Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Machine Intelligence and Pattern Recognition*, volume 9, pages 69–76. Elsevier, 1990.

[2] Jiji Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9(Jul):1437–1474, 2008.

[3] Judea Pearl. Causality: models, reasoning, and inference. *Econometric Theory*, 19(675-685):46, 2003.

[4] Judea Pearl. Causal inference in statistics: An overview. *Statist. Surv.*, 3:96–146, 2009.

[5] Richard Scheines. An introduction to causal inference.

[6] Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

[7] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

[8] Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16):1873 – 1896, 2008.