Appendix

CASTLE – Causality Analysis based on STructure LEarning

The core causality module of CASTLE is based on Parent-and-Child (PC) discovery in Bayesian Network (BN) structure learning. Followings are the details of our method.

Definition 1. Parent-and-Child (PC): Given two features f_1 and f_2, if there is directed edge from f_1 to f_2, then f_1 is the parent of f_2, and f_2 is the child of f_1.

Using the example in Figure 1, ADR contains two parents, f_1 and f_2, and ADR is the child of f_1 and f_2. A parent and its child is not d-separated[1] from each other, given any subset of features, which provides the theoretical foundation for the PC-discovery algorithm. The 'd' in d-separation stands for directional. If two variables X and Y are d-separated in a directed graph, they are conditional independent in all probability distributions. Let F denote the full feature space and ADR denote the states of ADR; the PC-discovery problem can be formally defined as follows:

Definition 2. PC Discovery: Finding a subset , in which each variable is not d-separated from ADR given any subset of the variable set F.

In practice, conditional independence tests can be used to detect d-separate. For instance, the state that two features f_1 and f_2 are d-separated by a feature set F' is equivalent to checking

whether there exists a feature subset that satisfies $f_1 1 \mid f_1 2 \mid F'$. Here, $f_1 1 \mid f_1 2 \mid F'$ refers to that f_1 and f_2 are conditional independent of each other given F'. In this study, the states of the features and ADR are all discrete, and we used G^2 conditional independence test.[2] Generally speaking, our framework maintains two feature sets at all times, including the current PC candidate set PC(ADR) and untested feature set C(ADR). First, we initialize PC(ADR) as an empty set and C(ADR) as the full feature set. Then, our algorithm sequentially checks each feature in C(ADR), and each iteration consists of two steps, growth and pruning.

In the growth step, a feature $f \in C(ADR)$ is added to PC(ADR) if it is not conditional independent of ADR given any subset of the current PC(ADR), which is only a weaker necessity condition because it considers the subsets of PC(ADR), not the complete feature set F. Therefore, the growth step potentially renders some false candidates to PC(ADR). Thus in the pruning step, features in PC(ADR) is pruned if it is conditionally independent of ADR given the newly added feature *f*. This heuristic step is proposed here to improve efficiency by removing features that do not belong to PC(ADR) in the early stage. After sequentially checking the features, a refinement step is conducted to remove redundant features. The refinement step is to verify the correctness of all existing features in PC(ADR) by checking whether each feature $f \in PC(ADR)$ is conditionally independent of ADR given any subset of PC(ADR). Pseudo code of the full PC-discovery algorithm is in Figure S1.

However, in case of high dimensional feature space, PC-algorithm may produce many false positives. Considering a set of 20,000 features and a conditional independence threshold of 95%,

there will be 5%*20,000 features falsely discovered. The following corollary of PC provides a solution.

Corollary 1: If $f_1 \in \mathbf{PC}(f_2)$, then $f_2 \in \mathbf{PC}(f_1)$.

Proof: According to the definition of PC set, if $f_1 \in \mathbf{PC}(f_2)$, then there is a direct edge between f_1 and f_2, then f_2 is in the PC set of f_1.

Based on Corollary 1, a Robust PC-discovery algorithm is proposed to reduce the false discovery rate by first identifying PC(ADR), and then filtering each feature *f* in the PC(ADR)'s PC set using the condition: if ADR is not in the PC(*f*), then *f* is removed from PC(ADR). To note, the number of Robust PCs will always be a subset of the discovered PCs. Assuming a feature is falsely added to the PC set by PC-discovery with a 5% probability (Figure S1), the Robust PC-discovery algorithm can reduce the false discovery rate to 0.25% because a feature *f* is falsely added to the PC(ADR) if and only if *f* is falsely identified in the PC-discovery of ADR and ADR is falsely identified in the PC-discovery of *f*. As the discovery procedures for PC(ADR) and PC(*f*) are conducted independently, it is reasonable to assume that the false discovery probability is independent of each other. Thus in the best case, false discovery rate of the Robust PC is 5%*5%=0.25%. Pseudo code of the Robust PC-discovery algorithm is in Figure S2.

References

1. Geiger D, Verma T, Pearl J. Identifying independence in Bayesian Networks. Networks. 1990;**20**(1990):507-34.

2. Spirtes P, Glymour C, Scheines R. Causation, Prediction, and Search. Second ed: The MIT Press; 2001.

Function PCDiscovery(F, ADR)
//Input: F: the full feature set, ADR: the label of ADR
//Output: PC(ADR):The PC set of ADR

//Initialization:
PC(ADR)=Φ,C(ADR)=F

//Growth:
for each $f \in C(ADR)$
    if $\exists F' \subseteq PC(ADR)$ satisfies $f \perp ADR|F'$
        $PC(ADR) = PC(ADR) + \{f\}$
    endif
    $C(ADR) = C(ADR) - \{f\}$

    //Pruning:

    for
        if $f' \perp ADR|f$
            $PC(ADR) = PC(ADR) - \{f'\}$
        endif
    endfor
endfor

//Refinement:
for each $f \in PC(ADR)$
    if satisfies $f \perp ADR|F'$

Figure S1. PC Discovery Algorithm – Pseudo code

Function RobustPCDiscovery(F, ADR)
//Input: F: the full feature set, ADR: the label of ADR
//Output: PC(ADR):The Robust PC set of ADR

PC(ADR)= PCDiscovery(F,ADR);
for each $f \in PC(ADR)$
    PC(*f*)= PCDiscovery(F, *f*);
    if $ADR \notin PC(f)$
        $PC(ADR) = PC(ADR) - \{f\}$
    endif
endfor
Return PC(ADR)

Figure S2. Robust PC Discovery Algorithm – Pseudo code