熵的因果推理：图识别

黄进科

摘要

熵因果推理是一种全新的框架，它借助熵的理论来判断变量之间的因果关系。在[1]的研究中，作者提出了联合熵最小化算法，该算法能够识别出具有两个节点的因果图。在之后[2]的工作中，作者又提出了两个算法，Entropic Peeling, Entropic Enumeration，这两个算法基于之前提出的联合熵最小化算法，可以识别出具有两个以上节点的因果图。Entropic Peeling 算法利用联合熵最小化算法判断两个节点之间的父子关系，利用此关系结合变量独立性识别出因果图。Entropic Enumeration 利用'真实世界因果图的熵很小'这一猜想，先利用变量独立性构建骨架图 (无向图)，在骨架图的基础上枚举所有可能的因果图，选取熵最小的因果图作为最终结果。

本文将用 Python 对 Entropic Peeling 和 Entropic Enumeration 进行实现，并在 bnlearn 数据集[1]上测试，同时将测试结果与传统的因果图识别算法 PC,GES 比较。

关键词：因果推理，熵，Entropic Peeling，Entropic Enumeration

1 引言

因果推理可以基于已有的观测数据，发现变量之间的因果关系，从而在错综复杂的关系中找到问题的根源，支持人们做出正确的决策。具体到机器学习方面，有时模型无法辨别一些虚假的相关性与真实的因果关系，笼统地将所有关系用于训练，这样的训练结果往往是不稳定且不可解释的。用因果推理去指导机器学习可以提高结果的可解释性和可靠性。

2 相关工作

传统的图识别算法有两大类。第一类是基于条件独立性约束的，代表的算法有 PC 算法，IC 算法。第二类是基于打分机制的算法，比如 GES 算法，fGES 算法。本文需要与 PC 算法和 GES 算法进行比较，所以对两者进行进一步介绍。PC 算法利用变量条件独立性生成因果图，由于不同的因果图可能具有相同的变量独立性，因此基于变量独立性原理的 PC 算法某些情况下无法正确识别出边的方向 (即两个变量之间有因果关系，但无法确定谁是因谁是果)，故 PC 算法生成的因果图是一个部分有向图 (PDAG)。GES 算法基于贪心思想，使用打分函数 BDeu，每次选取分值最高的操作 (插入或删除边)，期望通过每次寻找局部最优的操作，最终得到分值较高的因果图。

3 本文方法

在[2]中，作者提出 Entropic Peeling 和 Entropic Enumeration 两个算法, 算法大致过程如下。

3.1 Entropic Peeling

1. 利用联合熵最小化算法和变量的条件独立性，构建出一个拓扑序列。
2. 对拓扑序列的变量再次进行条件独立性判断，构造出因果图。

[1]https://www.bnlearn.com/bnrepository/

3.2 Entropic Enumeration

1. 利用变量条件独立性，构造出骨架图 (无向图)。

2. 在骨架图的基础上，枚举所有的有向图情况。假如骨架图有 e 条边，则相应的有向图有 2^e 种情况。

3. 计算每个有向图的熵值，取熵值最小的有向图作为最终的因果图。

4 复现细节

4.1 Entropic Peeling

原论文[2]中作者给出了伪代码，我对该伪代码的两处逻辑进行了修改，用注释做了标记。

Procedure 1 Entropic Peeling

Input: dataset DF
Output: directed acyclic graph DAG
$R \leftarrow$ {1,...,|V|} {set of remaining nodes}
$I \leftarrow \emptyset$ {set of pairs found to be conditionally independent}
$T \leftarrow$ [] {list of nodes in topological order}
while $|R| > 0$ **do**
  $N \leftarrow \emptyset$ {set of nodes discovered as non-sources}
  $C \leftarrow$ {1,...,|V|} $\setminus$ R {condition on previous sources}
  for all $(X_i, X_j) \in \{R \times R\}$ **do**
    if $X_i \notin N$ **and** $X_j \notin N$ **and** $(X_i, X_j) \notin I$ **then**
      if CI($X_i, X_j|C$) **then**
        $I \leftarrow I \cup (X_i, X_j)$
      else if Oracle($X_i, Xj|C$) orients $X_i \rightarrow X_j$ **then**
        $N \leftarrow N \cup \{X_j\}$ {X_j is not a source}
      else
        $N \leftarrow N \cup \{X_i\}$ {Xi is not a source}
      end
    end
  end
  $S \leftarrow R \setminus N$ {the remaining nodes that are a source}
  $R \leftarrow R \setminus S$ {remove sources from remaining nodes}
  for all $X_i \in S$ **do**
    append X_i to T
  end
end
$skeleton$ = getSkeleton(DF) `// added by Jinke`
for *all* $(i, j) \in \{1, ..., |V|\}^2$ where i < j **do**
  `// modified by Jinke`
  if CI($T(i), T(j)|\{T(1), ..., T(j-1)\}\backslash T(i)) == False$ **and** edge($T(i), T(j)$) exists in *skeleton* **then**
    orient $T(i) \rightarrow T(j)$
  else
    no edge between $T(i)$ and $T(j)$
  end
end

我一开始根据作者给的伪代码进行实现，但是识别准确率很差，所以对伪代码的逻辑进行了修改。

为什么原来的伪代码的实现效果不好？

个人认为，因为 Oracle() 借助的是联合熵最小化算法[1]来判定两个节点的父子关系，我测试过，联合熵最小化代码 (也是自己实现的) 判断的正确率大约只有 50%。一旦 Oracle() 判断错误，就会导致某

个叶子节点率先加入拓扑序列。而先加入拓扑序列的叶子节点，在构建有向边的时候，由于只判断变量独立性，就会导致该叶子节点与拓扑序列后面的变量节点建立大量实际不存在的有向边。

针对上述问题，我修改了原代码两地方。修改 1：根据变量的条件独立性创建了骨架图 (同 PC 算法的第一步一样)。修改 2：在最终构建因果图的时候，当两个节点之间条件相关时还需判断两个节点在骨架图中是否存在边，若存在才建立映射关系，否则不建立边，从而避免一些错误边的建立。

测试证明，修改完的代码效果更好。

我想在此先说明一下，本文内容都是基于我个人的理解进行编写的，我是门外汉，很大概率不是作者的原代码不对，而是我的理解有误，所以本文内容仅供参考。

4.2 Entropic Enumeration

原论文[2]作者给出了思路，根据作者的思路，我的具体实现如下

Procedure 2 Entropic Enumeration

Input: dataset DF
Output: directed acyclic graph DAG
$skeleton$ = getSkeleton(DF)
$minimum\_entropy$ = INFINITE
$result\_dag$ = NULL
while (dag = enumerate_dag($skeleton$)) != NULL **do**
  if $minimum\_entropy$ > (e = getEntropy(dag)) **then**
    $minimum\_entropy = e$
    $result\_dag = dag$
  end
end
return $result\_dag$

getSkeleton 方法可以借鉴 PC 算法的代码实现。enumerate_dag 方法可以采用深搜的方式遍历。getEntropy 方法用于返回 dag 图的熵，dag 图的熵值等于 dag 图中所有的节点的熵值的总和，单个节点的熵值等于该节点与其父节点之间的联合熵值，联合熵值可以用联合熵最小化算法计算。

4.3 实验环境搭建

1. 从 github 上下载本文章关于 Entropic Peeling, Entropoc Enumeration 的项目代码实现[2]
2. 安装 Python: 3.9.13
3. 安装相关依赖包：pip install -r 代码目录/requirements.txt

4.4 项目目录结构说明

1. data 目录：存放待测因果图的.bif 文件[3]，程序将利用.bif 文件生成所需要的数据集
2. output 目录：存放输出的结果
3. Entropic_Peeling.py: 论文的算法实现
4. Entropic_Enumeration.py: 论文的算法实现
5. main.py: 主函数文件
6. utils.py: 工具函数文件

[2]https://github.com/szu-advtech/AdvTech，具体项目路径是 2022/7-黄进科 指导老师-廖好

[3]https://www.bnlearn.com/bnrepository/

4.5 使用说明

1. 在 Windows 命令窗口 cmd 下，进入项目目录，执行 python main.py

2. 输出结果在项目的 output/images 下，以程序运行时间作为文件夹名称，各种算法的运行结果均以折线图的呈现方式存放在此目录中。

4.6 创新点

本文章实现了论文[2]中描述的 2 个因果图识别算法，以及论文[1]的联合熵最小化算法 (并非本文重点，故没有列出)。2 个算法的实现代码在小规模图上运行效果基本符合原论文要求。从而验证了熵的理论在识别因果图上是有一定作用的。

5 实验结果分析

检验识别结果准确率的指标是结构化汉明距离 SHD(Structural Hamming Distance)，SHD 等于一个有向图通过删除边，增加边，反转边转化为另一个有向图所需要的最少步骤。SHD 越小，表明 2 个图越接近。我们将用 4 种不同的算法 (本文章实现的 2 种熵算法，以及现有的 PC，GES 算法) 对样本进行学习，得到各自的因果图。将因果图和真实因果图进行比较，得到不同算法运行结果的 SHD，比较各个算法的识别效果。我们根据图的规模将测试的因果图分为 2 类：Small Networks 和 Medium Networks。测试结果如下。

5.1 Small Networks

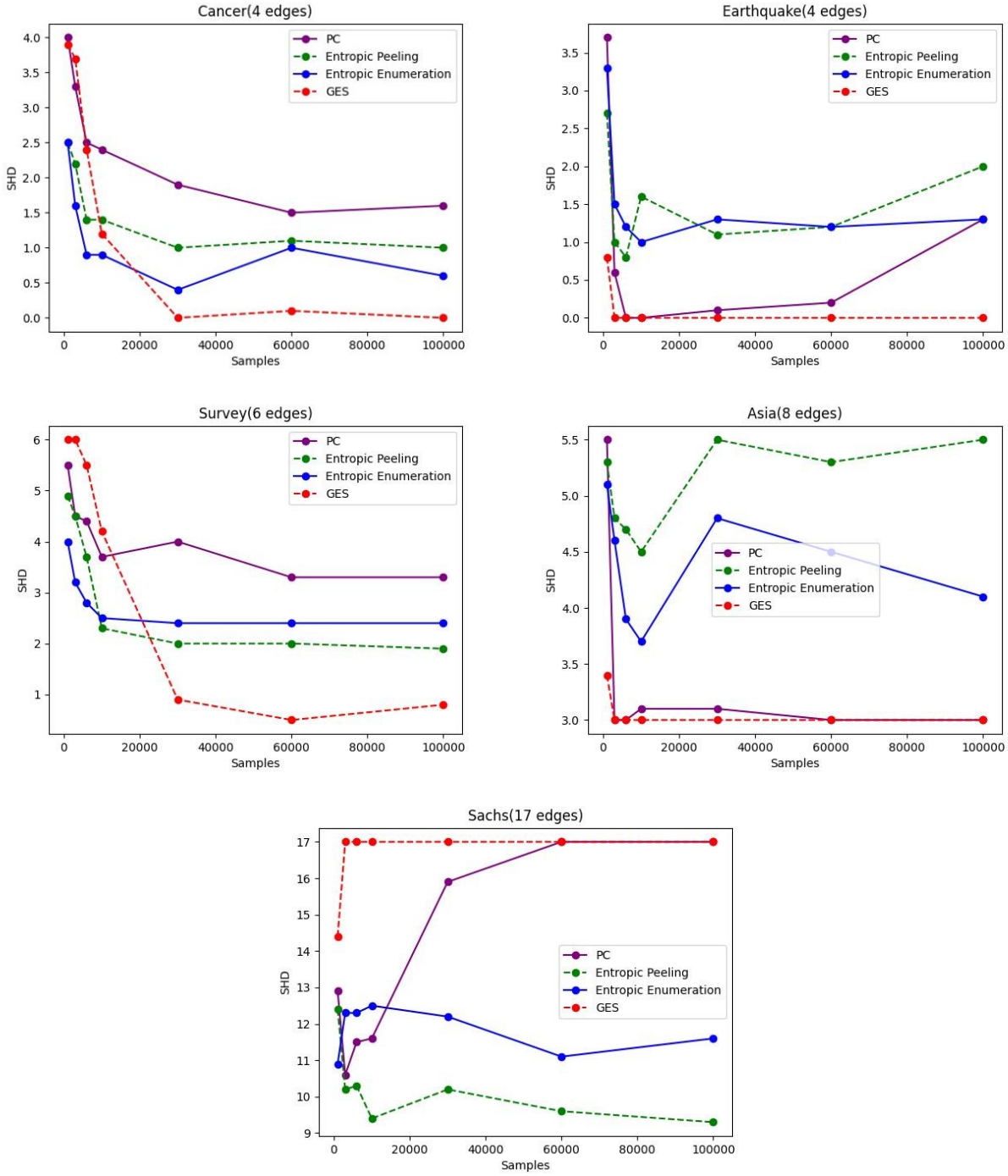

图 1: Small Networks Test

纵轴是 SHD，横轴是样本数量。

每张图中的每个数据点都是取 10 个样本数量相同，但数据不同的数据集测试结果的平均值。

从图 1我们可以观察到 2 点：1. 熵算法在多个数据集逊色于 GES，唯独在 Sachs 上优于 GES。2. 熵算法与 PC 算法效果差不多，但是在 Sachs 上，熵算法要明显优于 PC 算法。

5.2 Medium Networks

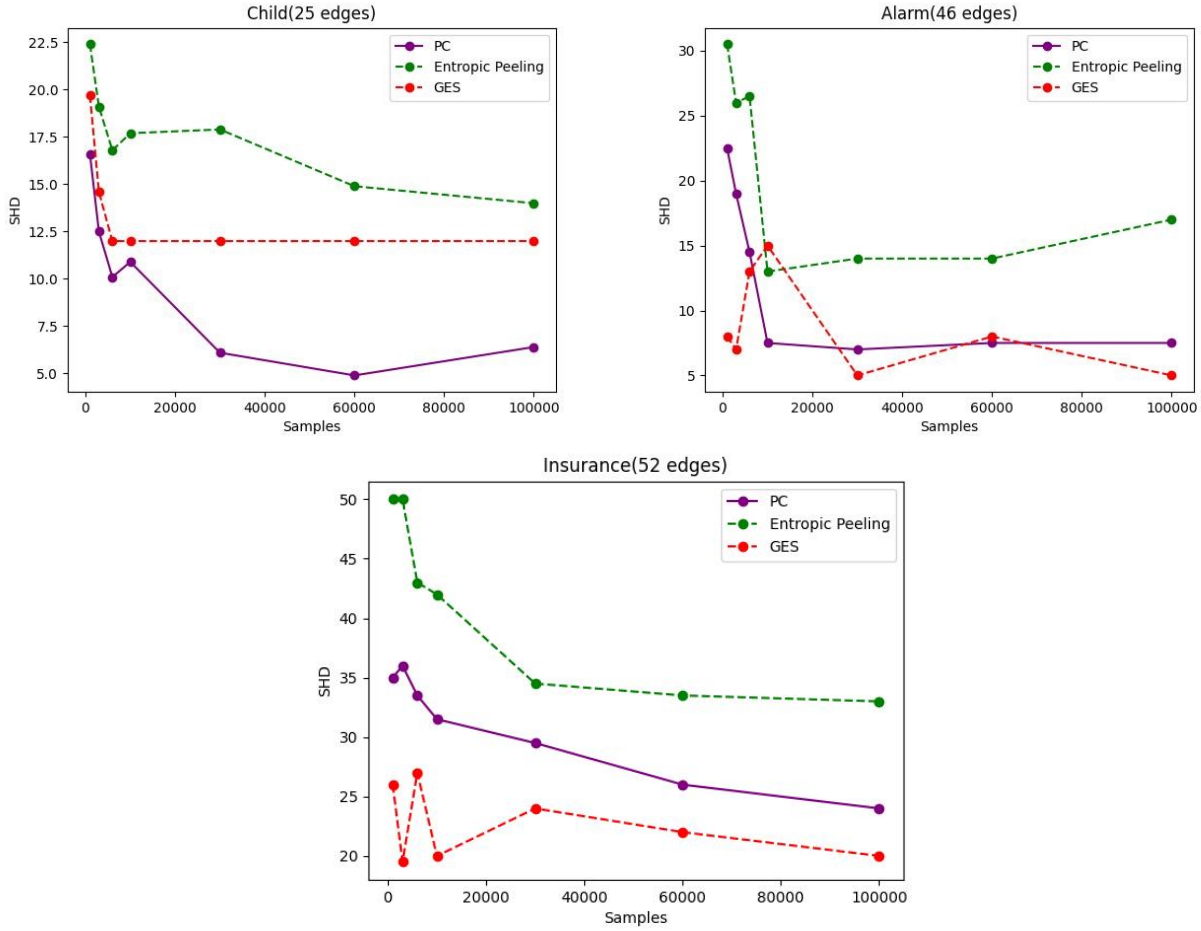

图 2: Medium Graphs Test

纵轴是 SHD，横轴是样本数量。

每张图中的每个数据点都是取 2 个样本数量相同，但数据不同的数据集测试结果的平均值。

从**图** 2可以观察到 2 点：1.Entropic Peeling 算法均逊色于其它算法，但是在 Child 和 Alarm 上的识别准确率均符合原论文的要求。2. 在 Insurance 上，熵算法 SHD 一直处于 30 以上，没能达到原论文 20∼25 的要求。另外，我实现的 Entropic Enumeration 复杂度过高，无法处理较大的图，故此处无法做比较。

6 总结与展望

本文对论文[2]的两个熵算法进行了实现，并在 bnlearn 提供的网络上对算法进行了测试。Entropic Peeling 在除了 Insurance 之外的其它网络，识别准确率都与原论文基本相符。Entropic Enumeration 在 Small Networks 上的识别准确率与原论文基本相符。

本文的算法实现也存在不足之处。对于 Entropic Peeling 的实现，算法在 Insurance 数据集上识别效果并不理想，SHD≈35，没有达到原论文 SHD≈25 的要求。对于 Entropic Enumeration 的实现，算法复杂度高，除了枚举所有 dag 图的复杂度 O(2^{edges}) 之外，还需要计算每个节点的熵值 (计算熵值需要对数据集进行多次遍历)，该实现目前只能针对小规模图 (大约 18 条边以下的骨架图) 进行判别。对于中等规模因果图 (19 条边以上的，比如 Child, Alarm, Insurance)，该算法基本无法运行，可能会栈溢出，内存溢出。故该算法有待改进优化。

我希望以后可以把 Entropic Enumeration 的算法优化一下，看看对大一点的图识别效果是不是真的很好。

再次说明，本文的算法的实现均是基于我个人对原论文[2]的理解，因为本人能力和水平十分有限，如果有理解不正确的地方，还请各位多多指正。

参考文献

[1] KOCAOGLU M, DIMAKIS A G, VISHWANATH S, et al. Entropic causal inference[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[2] COMPTON S, GREENEWALD K, KATZ D A, et al. Entropic causal inference: Graph identifiability[C]//International Conference on Machine Learning. 2022: 4311-4343.