Learning latent variable models from data

Goutham Rajendran

THE UNIVERSITY OF CHICAGO

Background

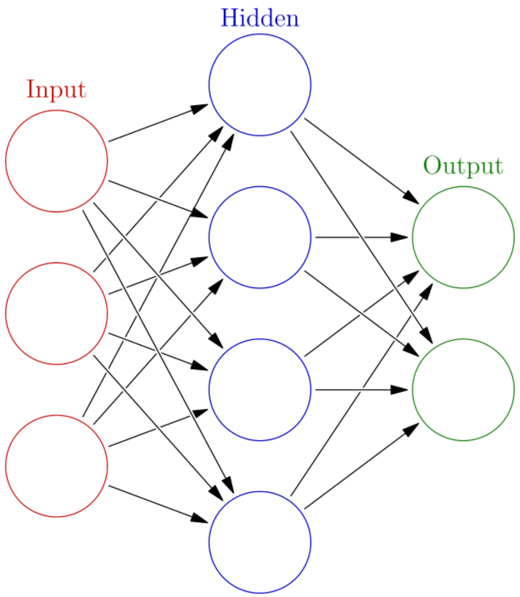
- A large body of work in Machine learning has been to fit models to data
- Two important subfields:
  - Robust ML: Can we learn when the data is extremely noisy?
  - Interpretable ML: Can we build a model that's easy to understand?

Robust Machine Learning

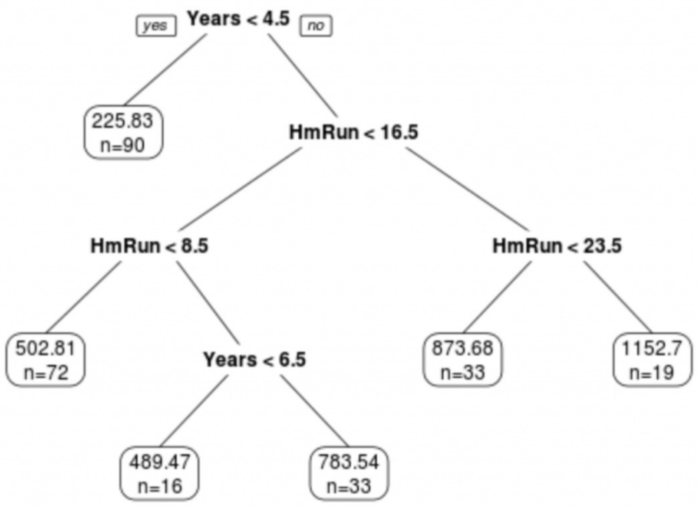
- Standard models usually account for mild noise
- What if there is a large fraction of random or even adversarial noise?
- Applications in finance, biology, economics, etc.
- Some of my projects (authors in alphabetical order):
  - Learning communities in large networks [JP**R**TX, Foundations of Computer Science 2021]
  - Sherrington-Kirkpatrick model [GJJP**R**, Foundations of Computer Science 2020]
  - Sparse principal components analysis [P**R**, Under submission 2022]

Interpretable Machine Learning

- Which model would you prefer?



Src: wiki



Src: statology

- An important branch of interpretable ML: Causal inference

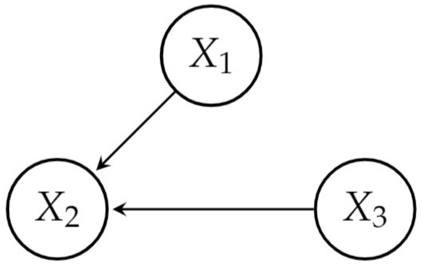
Causal inference

- Given variables or features, can we identify if some cause others?
- Useful to correctly intervene. For example, how will increasing price and/or changing material quality affect product sales?
- Applications in ML, physics, medicine, genetics, etc.
- Some of my projects:
  - Causal structure learning in polynomial time [**R**KGA, NeurIPS 2021]
  - Learning latent variable causal models [K**R**RA, NeurIPS 2021] (This talk)

Bayesian network diagrams

- Graphical models compactly represent causal relationships
- A very simple example:

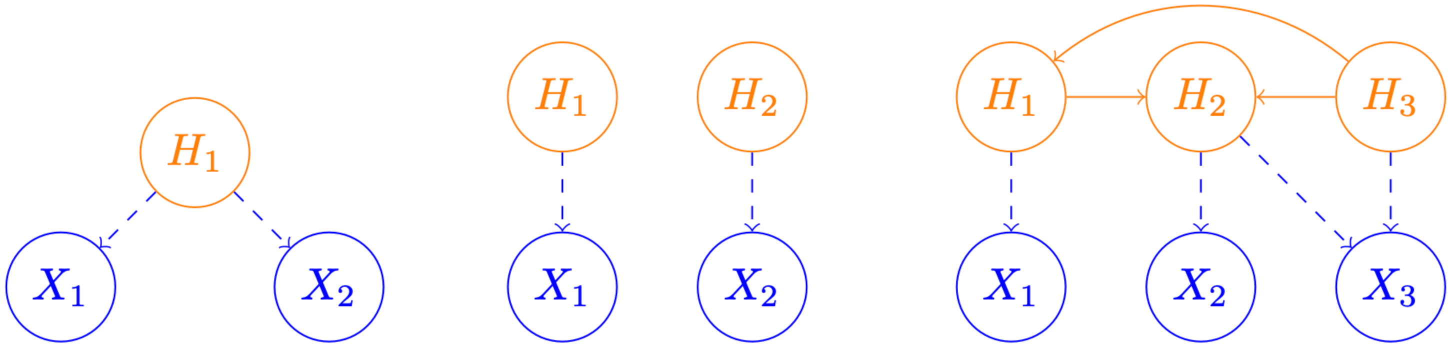
$$
\begin{array}{c}
\begin{matrix} X_1 & \quad X_2 & \quad X_3 \end{matrix} \\
\begin{bmatrix}
0.89 & -3.17 & 1.65 \\
-1.34 & -4.59 & 0.64 \\
2.11 & 5.05 & -0.28 \\
-0.23 & 0.56 & -0.34 \\
0.29 & -4.40 & 1.66 \\
 & \dots &
\end{bmatrix}
\end{array}
\longleftrightarrow
$$



- Learning such cause-effect relationships is a very hard problem
- But such models are robust to outliers, noise, etc., and help us intervene

Latent variable models

- Some variables are hidden (called latent) but they exist
- Any ML model in the real world must account for them



(a) A single hidden state.

(b) Two independent hidden states.

(c) Three dependent hidden states.

- Here, blue nodes are observed and red are hidden
- Our grand goal: Can we learn the entire causal model?

Why care?

- More powerful models explain data better and help us reason about things, e.g. epidemiology
- They let us build more intelligent systems capable of human level reasoning, e.g. robotics
- They let us generate realistic fake datapoints which lets us train against bad actors
  - Example 1: adversarial examples in self-driving cars
  - Example 2: VAEs (which are just latent variable models!) generate fake human faces which can be used for other downstream tasks such as to train GANs

Our work – An example

- This is a Gaussian mixture model and we observe the projections
- We recover the latent variable model on the right

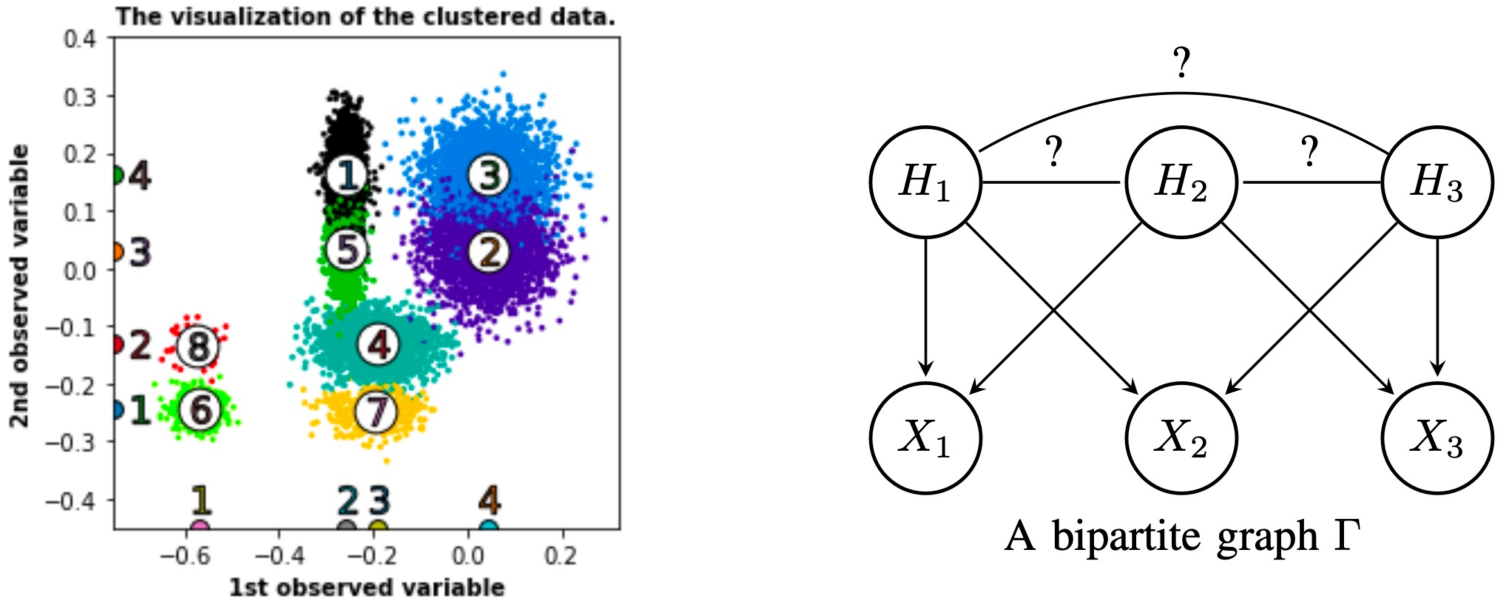


Figure 2: Example of a latent DAG and corresponding mixture distribution

Our work - Context

- In general raw data, such learning is impossible
- Indeed, there could be millions of models that explain the data
- In our work, we show that under specific assumptions, it’s possible to learn the model.
- In particular, our work is the first work that
  - Recovers causal relationships between latents
  - Works when the causal edges are not necessarily linear (The linear setting was partially solved in the topic modeling literature [AHJK, ICML 2013])

Our work - Assumptions

- Our main assumptions:
  - *Discrete latent variables* - latents represent states, not distributions
  - *Markovian property* - data is generated from a reasonable model
  - *Subset condition*: One latent variable should not encompass another
    - Closely related to anchor words assumption in topic modeling literature
  - *Nondegeneracy assumptions* – e.g. components cannot vanish or significantly overlap
  - *Existence of a mixture oracle* - Mixture models can be learnt, e.g. Gaussian mixtures can be learnt via EM algorithm, clustering, k-means, etc

Our work – Main algorithm

- Our main algorithm can be split into 2 parts
- Part 1: Identify latent variables' states and 1st layer of causal connections
- Part 2: Identify joint distribution of latent variables and 2nd layer of causal connections

- How do we measure accuracy?
- If we know ground truth: Structural Hamming distance, unoriented correct edges

Our work – Main algorithm part 1

- We first learn mixture models for small subsets of variables
  - Use various voting techniques to improve accuracy across subsets
- Identify the first layer of the model by “factorizing” the components
  - Reconstruct the discrete states
  - Identify causal relationships across latent and observed variables
  - Uses tensor decomposition – Jennrich’s algorithm was the main driver, Alternating Least Squares (ALS) was the failsafe

Our work – Main algorithm part 2

- Identify connections between latent variables
- Use it to reconstruct the joint distribution on the variables
- Finally, run Greedy Equivalence Search with the Discrete BIC score to learn causality

Algorithm 1: Learning $\mathbb{P}(H)$

Input:
- A bijective map $L : [k(X)] \to [k(X_1)] \times [k(X_2)] \times \ldots \times [k(X_n)]$;
- A bipartite graph Γ between X and H
- Values $\dim(H_i)$ for $i \in H$.
- Values $\mathbb{P}(Z = i)$ for $i \in [k(X)]$ (the probabilities of observing the mixture components)

Output: An $\dim(H_1) \times \ldots \times \dim(H_m)$ tensor such that $J \cong \mathbb{P}(H)$

`// Phase 1: use Lemma C.1 to compute the sets of components that correspond to a change in a single hidden variable`

arrows $= \{\}$
for $H_i \in H$ **do**
  $S = X \setminus ne_\Gamma(H_i)$
  for $c_1, c_2 \in [k(X)]$ **do**
    if $(L(c_2)_S == L(c_1)_S)$ *and* $c_1 \neq c_2$ **then**
      arrows$[H_i][c_1]$.append(c_2)

`// Phase 2: initialize` T `"along the edges"`

$A(0, \ldots 0) = 0, \quad T(0, \ldots 0) = \mathbb{P}(Z = 0)$
for $H_i \in H$ *and* $t \in \dim(H_i)$ **do**
  $A(0, \ldots, t, \ldots 0) = arrows[H_i][0][t]$ `// Note that an order does not matter`
  $J(0, \ldots, t, \ldots 0) = \mathbb{P}(Z = arrows[H_i][0][t])$

`// Phase 3: reconstruct all other entries of the tensor`

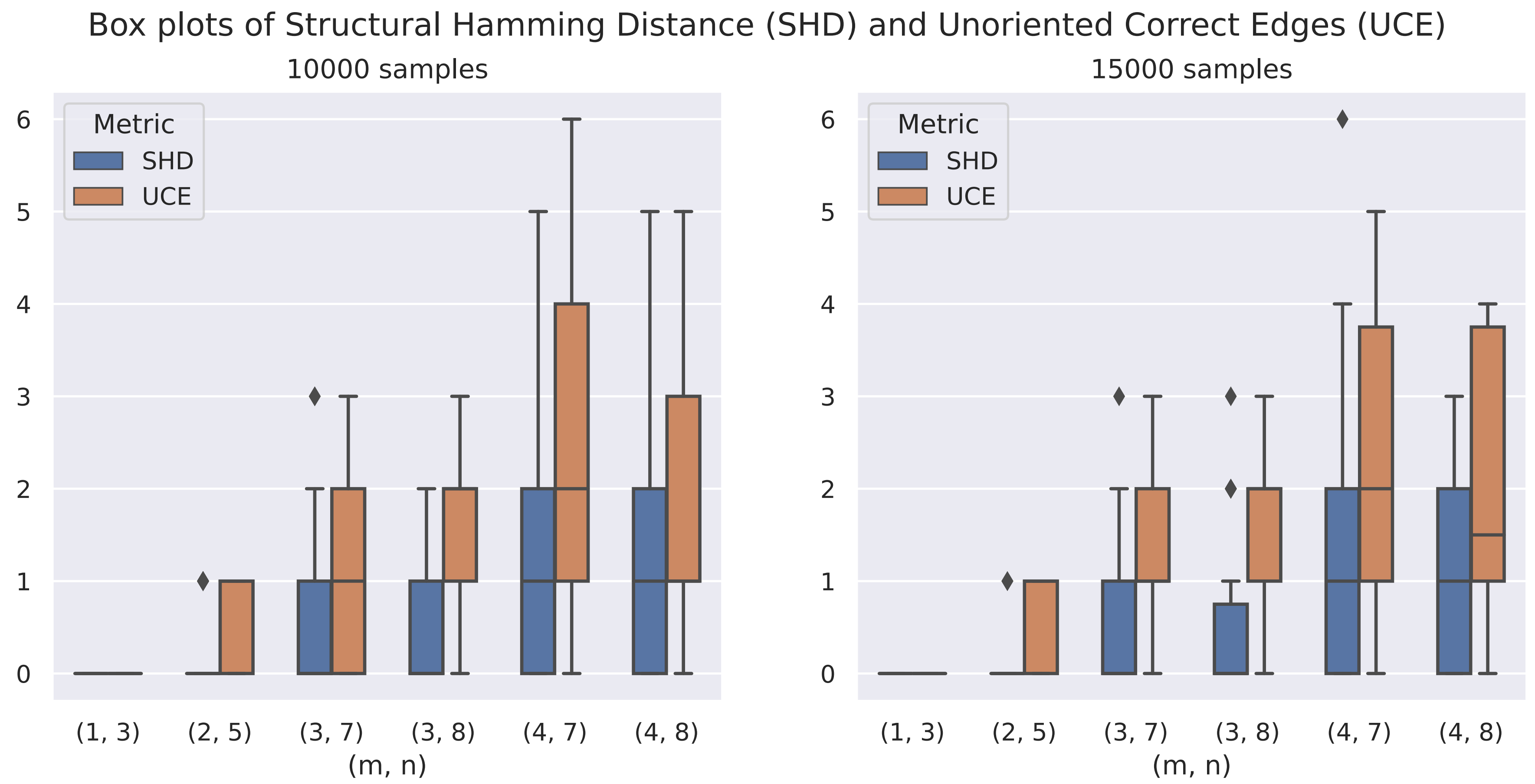
$r = 1$
while $r < m$ **do**
  for $ind \in \dim(H_1) \times \ldots \dim(H_r)$ **do**
    for $j = r + 1, \ldots, m$ *and* $t \in \dim(H_t)$ **do**
      Let i be the smallest index at which ind is non-zero.
      Let ind' be an index obtained from ind by changing j-th entry from 0 to t
      Let ind'' be obtained from ind' by changing i-th entry to 0.
      Let x be the unique entry in the intersection of arrows$[H_i][A(ind'')]$ and arrows$[H_t][A(ind)]$.
      $A(ind') = x$
      $J(ind') = \mathbb{P}(Z = x)$

return T

Our work – Experiments

- We built an end-to-end pipeline
- Ran synthetic experiments as proof of concept
  - Also validates our approach since real life data may work/fail for spurious reasons
- That means ground truth was available, so we report SHD/UCE metrics.
- Experimental setup:
  - m hidden variables, n observed variables
  - Gaussian mixtures with highly unbalanced clusters
  - Various design choices: k-means, agglomerative clustering, etc

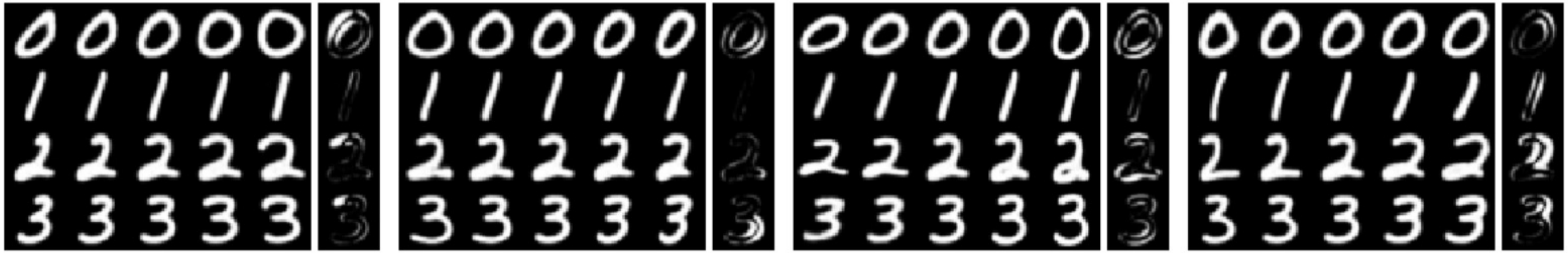
Our work - Experiments



Potential future directions

- (*Work in progress*) Use these ideas on vision datasets
  - Variational autoencoders
  - Nonlinear Independents Components Analysis

Prior work on this direction



(a) Variable 1: upper width (b) Variable 8: lower width (c) Variable 3: height (d) Variable 4: bend

src: [SRK, ICLR 2020]

- Rows are conditioned on digit
- Columns go from -2stddev to +2stddev

Potential future directions

- (*Work in progress*) Use these ideas on vision datasets
  - Variational autoencoders
  - Nonlinear Independents Components Analysis
- Part of our inspiration was work from topic modeling community
  - Applications to NLP?
- Modeling public opinion to better target intervention

Thank you