Hi! Setup for today that you'll need:

A clone of **github.com/pjbqb/dsip**

Follow the setup instructions in the README

1. Start a new conda environment:
   conda create -n causalnex python=3.7
   Do not try and do the install in the base environment, or use python 3.8. ***For Windows, python 3.6 may be safer****.*
2. Activate environment:
   conda activate causalnex
3. Use conda to install pygraphviz:
   conda install pygraphviz
   If you get an error that the package can't be found, try with conda-forge: conda install --channel "conda-forge" pygraphviz
4. Use conda to install Jupyter notebooks too:
   conda install notebook
5. Finally:
   pip install –r requirements.txt

CausalNex

Bayesian Networks

Thinking about causality in ML

Dr Paul Beaumont

Principal Data Scientist

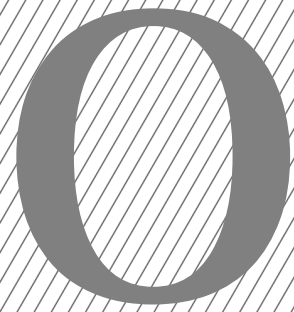
CausalNex

Agenda

● Talk / discussion ● Practical

0. Introduction to ML, causality & Bayesian Networks
1. Structure learning
2. Discretisation & probability fitting
3. Inference & interventions
4. Close

Introduction to causality & Bayesian Networks



The world of machine learning – in a nutshell

Predictive

Estimate the target for new observations

$$y = f(x)$$

Descriptive

Explain the effect that a change of certain inputs has on the target

$$y = f(x)$$

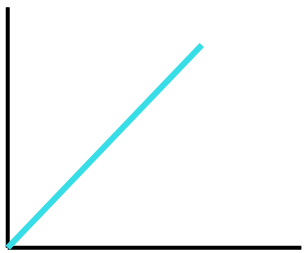
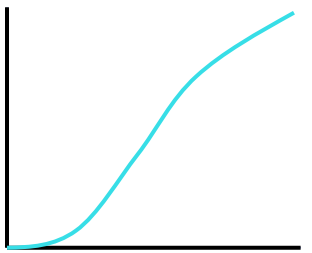
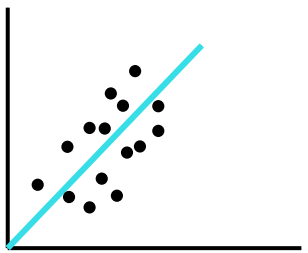
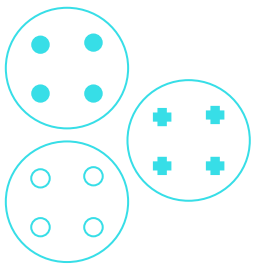
Prescriptive

Find the inputs that give optimal performance

f is known

$$y = f(x)$$

The truth is, there is no one way to explain different modelling techniques; they 'split' along multiple dimensions…

Y	**Supervised vs. Unsupervised**	?
$Y = f(x)$	**Univariate vs. Multivariate**	$Y = f(x_1, x_2, \ldots)$
	**Linear vs. Nonlinear**	
	**Regression vs. Classification**	

...the reality is complex and difficult to document

Classification

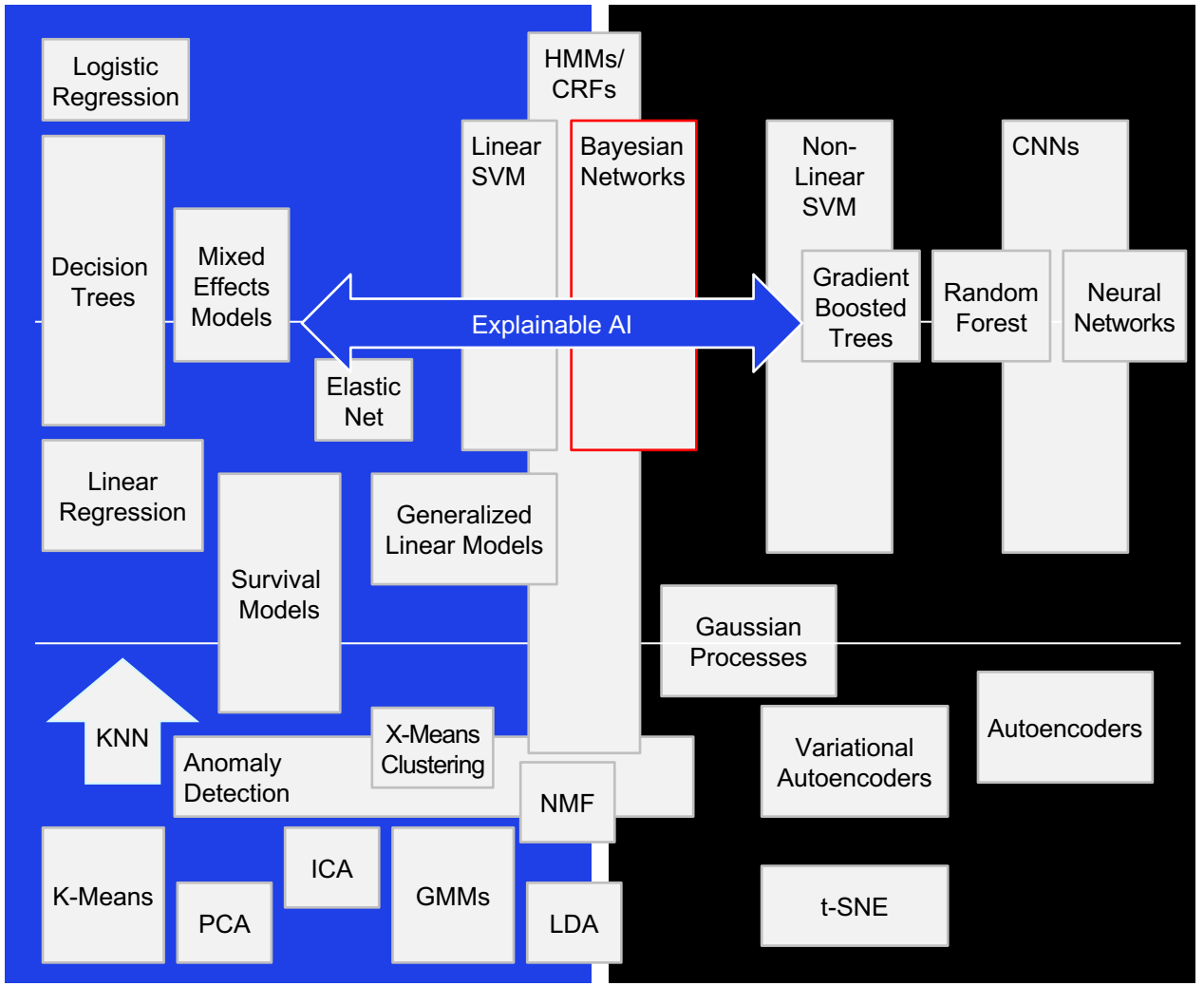
What group does this sample belong to?

Regression

What value would this sample take based on the data?

Unsupervised

What are the hidden / underlying patterns in my data?



Prescriptive

- Reinforcement Learning
- Q-Learning
- TD-Learning
- SARSA
- TRPO
- Genetic Optimization
- Non-Linear Optimization
- LP Optimization
- Greedy Algorithms
- Treatment-Effect Models
- Manual Decision Trees/Rules

Aside from the model choice, other important factors must be considered

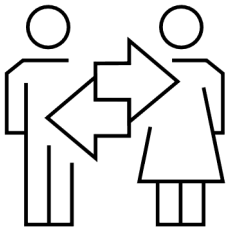
NOT EXHAUSTIVE

Deep dive

QuantumBlack does active research in all of these areas (and more)...

Algorithm fairness/bias

Methods for detection and correction of bias in models, ensuring models make fair predictions devoid of discrimination



Explainability

Facilitation of high performing machine learning models to be interpreted by humans, i.e., provide explanations for black box models

Causal inference

Models able to encode causal relationships, not just correlations, and allowing inclusion of input from domain experts

Ongoing model performance

Tool to enable tracking how models perform over time, supporting in identifying when to take corrective actions to sustain performance

...with many publications

DYNOTEARS: Structure Learning from Time-Series Data

Roxana Pamfil[1*], Nisara Sriwattanaworachai[1*], Shaan Desai[1], Philip Pilgerstorfer[1], Paul Beaumont[1], Konstantinos Georgatzis[1], Bryon Aragam[2]

[1]QuantumBlack, a McKinsey company causal@quantumblack.com

[2]University of Chicago bryon@chicagobooth.edu

*These authors contributed equally.

Abstract

We revisit the structure learning problem for dynamic Bayesian networks and propose a method that simultaneously estimates contemporaneous (*intra-slice*) and time-lagged (*inter-slice*) relationships between variables in a time-series. Our approach is score-based, and revolves around minimizing a penalized loss subject to an acyclicity constraint. To solve this problem, we leverage a recent algebraic result characterizing the acyclicity constraint as a smooth equality constraint. The resulting algorithm, which we call DYNOTEARS, outperforms other methods on simulated data, especially in high-dimensions as the number of variables increases. We also apply this algorithm on real datasets from two different domains, finance and molecular biology, and analyze the resulting output. Compared to state-of-the-art methods for learning dynamic Bayesian networks, our method is both scalable and accurate on real data. The simple formulation and competitive performance of our method make it suitable for a variety of problems where one seeks to learn connections between variables across time.

1 Introduction

Graphical models are a popular approach to understanding large datasets, and provide convenient, interpretable output that is needed in today's high stakes applications of machine learning and artificial intelligence. In particular, with the growing need for interpretable models and causal insights about an underlying process, directed acyclic graphs (DAGs) have shown promise in many applications. The edges in a DAG provide users with important clues about the relationship between variables in a system. When these edges are not known based on prior knowledge, it is necessary to resort to structure learning, namely, the problem of learning the edges in a graphical model from data. Broadly speaking, structure learning can be divided into static (i.e. equilibrium) and dynamic models, the latter of which explicitly model temporal dependencies. Static models make sense for independent and identically distributed data. Many applications, however, exhibit strong temporal fluctuations that we are interested in modeling explicitly. The problem of learning graphical structures from temporal data collected from dynamic systems has received significant attention from the machine learning (Koller and Friedman, 2009), econometrics (Lütkepohl, 2005), and neuroscience (Rajapakse and Zhou, 2007) communities.

In this paper, we revisit the problem of learning *dynamic Bayesian networks* (DBNs) (Dean and Kanazawa, 1989; Murphy, 2002) from data. DBNs have been used successfully in a variety of domains such as clinical disease prognosis (Van Gerven et al., 2008; Zandona et al., 2019), gene regulatory network (Linzner et al., 2019), facial and speech recognition (Meng et al., 2019; Nefian et al., 2002), neuroscience (Rajapakse and Zhou, 2007), among others. DBNs are the standard approach to modeling discrete-time temporal dynamics in directed graphical models. In econometrics, they are also known as *structural vector autoregressive* (SVAR) models (Demiralp and Hoover, 2003; Swanson and Granger, 1997).

We propose a simple, score-based approach for learning these models that scales gracefully to high-dimensional datasets. To accomplish this, we cast the problem as an optimization problem (i.e. score-based learning), and use standard second-order optimization schemes to solve the resulting program. Our approach is based

[†] Contributed during an internship at QuantumBlack. Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

Key decisions require explanatory models

Which medication will help a given patient?

What marketing campaign will be most effective?

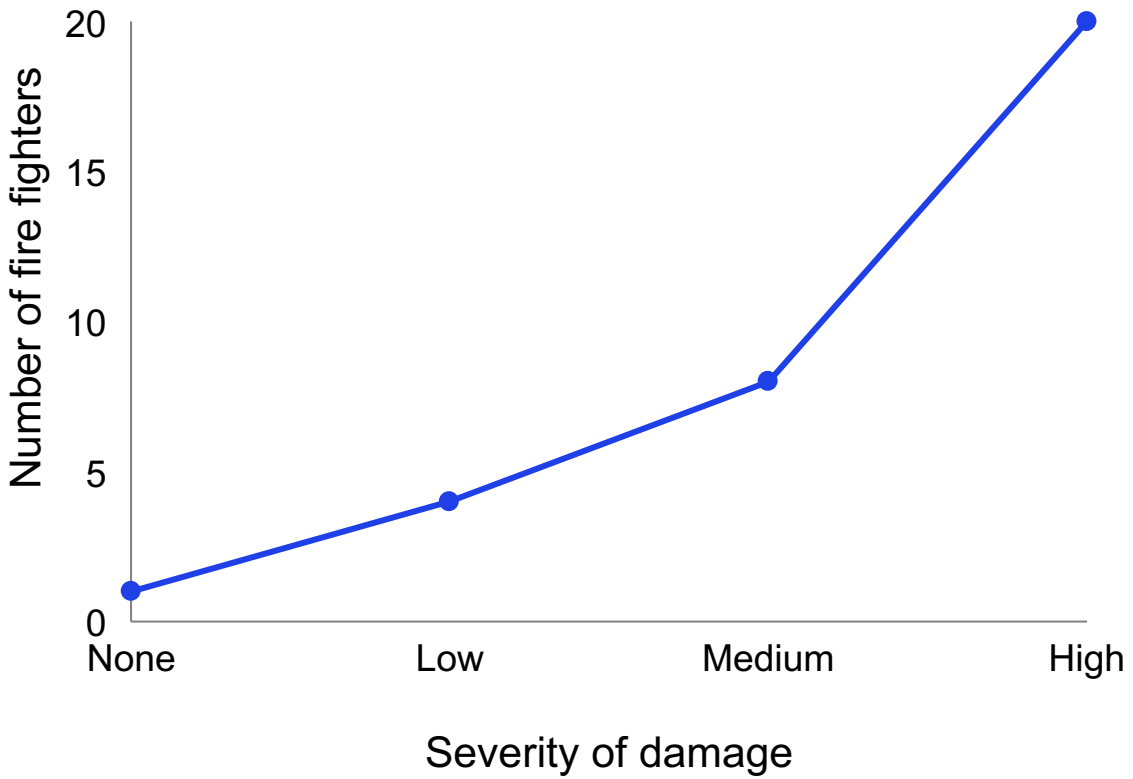
How can a pharmaceutical company reduce non-conformities during their drug manufacturing process?

What changes can a vehicle manufacturer make to their new product development process to reduce lead time?

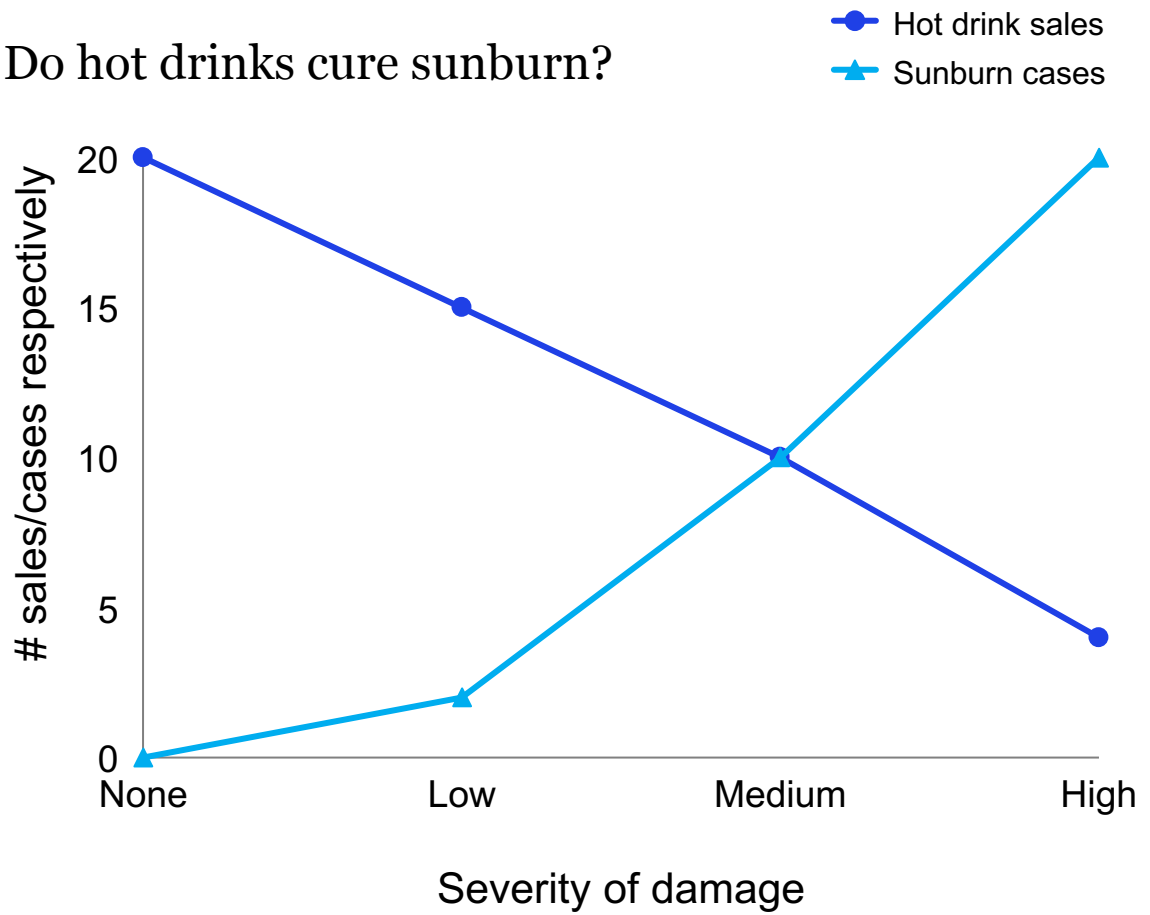
How can an company deploy resources to better serve customers?

We'd expect explanatory models to make causal sense before trusting a model

Should we get rid of fire fighters?



Do hot drinks cure sunburn?



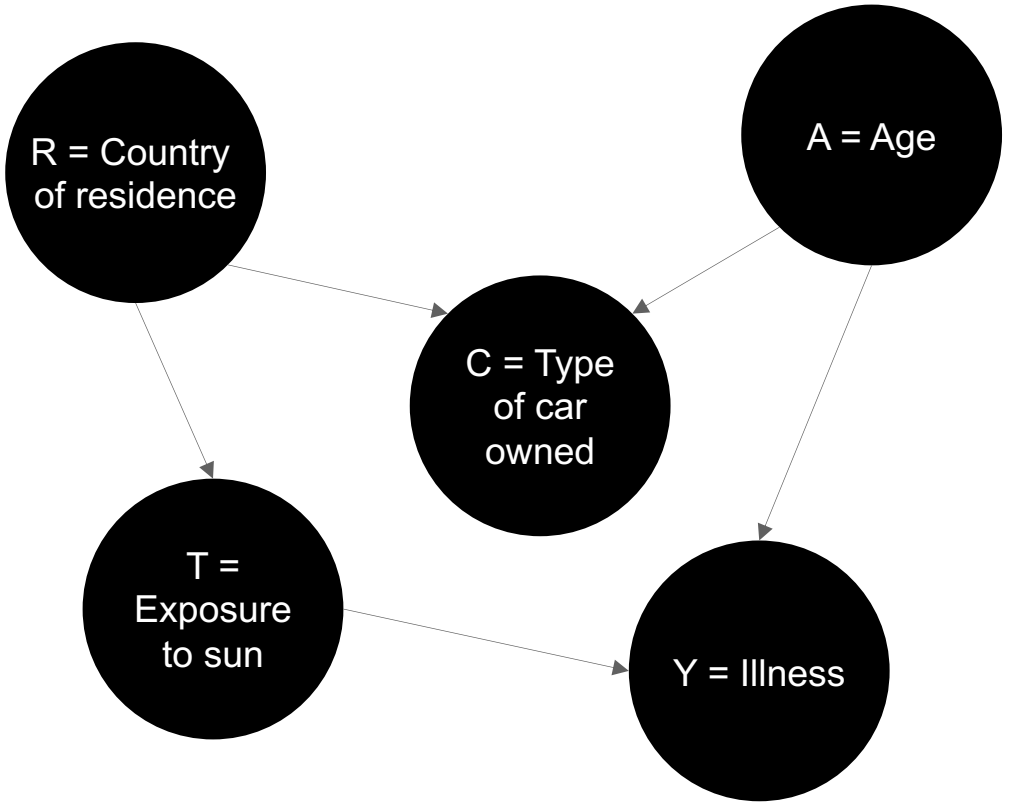
Q1: What is going on here? Is there a causal link?

Correlation is not causation…

Tackling these problems with classical machine learning can be problematic

Goal:

Find the effect of sun exposure on the illness



$R = \varepsilon_1$

$A = \varepsilon_2$

$C = R + A + \varepsilon_3$

$T = R + \varepsilon_4$

$Y = T + 10A + \varepsilon_5$

$\varepsilon_1, \ldots, \varepsilon_5$ independent normal(0,1)

Q2: Which is the better model?

```
model = sm.OLS(train.y, train[['t']])
results = model.fit()
print(results.summary())
```

Test RMSE = 10

OLS Regression Results

Dep. Variable:	y	R-squared:	0.019
Model:	OLS	Adj. R-squared:	0.019
Method:	Least Squares	F-statistic:	2.766e+04
Date:	Wed, 13 Feb 2019	Prob (F-statistic):	0.00
Time:	07:25:25	Log-Likelihood:	-5.2235e+06
No. Observations:	1401801	AIC:	1.045e+07
Df Residuals:	1401800	BIC:	1.045e+07
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
t	0.9984	0.006	166.317	0.000	0.987	1.010

```
model = sm.OLS(train.y, train[['t', 'car']])
results = model.fit()
print(results.summary())
```

Test RMSE = 7.7

OLS Regression Results

Dep. Variable:	y	R-squared:	0.408
Model:	OLS	Adj. R-squared:	0.408
Method:	Least Squares	F-statistic:	4.835e+05
Date:	Wed, 13 Feb 2019	Prob (F-statistic):	0.00
Time:	07:27:29	Log-Likelihood:	-4.8695e+06
No. Observations:	1401801	AIC:	9.739e+06
Df Residuals:	1401799	BIC:	9.739e+06
Df Model:	2		
Covariance Type:	nonrobust		

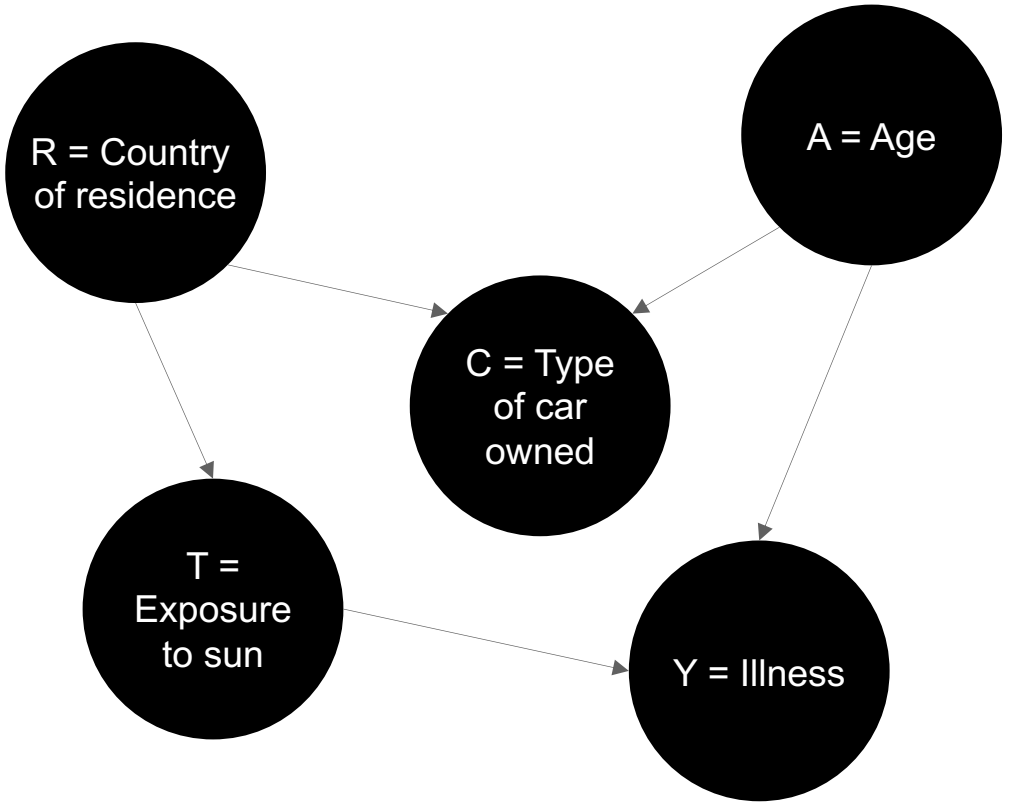
	coef	std err	t	P>|t|	[0.025	0.975]
t	-1.0036	0.005	-196.455	0.000	-1.014	-0.994
car	4.0024	0.004	959.740	0.000	3.994	4.011

Pearl (2014) Comment: Understanding Simpson's Paradox, The American Statistician. Simpson machine generator: http://dagitty.net/learn/simpson/

Tackling these problems with classical machine learning can be problematic

Goal:

Find the effect of sun exposure on the illness



$R = \varepsilon_1$

$A = \varepsilon_2$

$C = R + A + \varepsilon_3$

$T = R + \varepsilon_4$

$Y = T + 10A + \varepsilon_5$

$\varepsilon_1, \ldots, \varepsilon_5$ independent normal(0,1)

Q2: Which is the better model?

```
model = sm.OLS(train.y, train[['t']])
results = model.fit()
print(results.summary())
```

Test RMSE = 10

OLS Regression Results

Dep. Variable:	y	R-squared:	0.019
Model:	OLS	Adj. R-squared:	0.019
Method:	Least Squares	F-statistic:	2.766e+04
Date:	Wed, 13 Feb 2019	Prob (F-statistic):	0.00
Time:	07:25:25	Log-Likelihood:	-5.2235e+06
No. Observations:	1401801	AIC:	1.045e+07
Df Residuals:	1401800	BIC:	1.045e+07
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
t	0.9984	0.006	166.317	0.000	0.987	1.010

```
model = sm.OLS(train.y, train[['t', 'car']])
results = model.fit()
print(results.summary())
```

Test RMSE = 7.7

OLS Regression Results

Dep. Variable:	y	R-squared:	0.408
Model:	OLS	Adj. R-squared:	0.408
Method:	Least Squares	F-statistic:	4.835e+05
Date:	Wed, 13 Feb 2019	Prob (F-statistic):	0.00
Time:	07:27:29	Log-Likelihood:	-4.8695e+06
No. Observations:	1401801	AIC:	9.739e+06
Df Residuals:	1401799	BIC:	9.739e+06
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
t	-1.0036	0.005	-196.455	0.000	-1.014	-0.994
car	4.0024	0.004	959.740	0.000	3.994	4.011

Pearl (2014) Comment: Understanding Simpson's Paradox, The American Statistician. Simpson machine generator: http://dagitty.net/learn/simpson/

Running experiments is often not feasible and we may not observe all results we're interested in

% Data that exists

Experimental data

e.g. Randomised Control Trials

<1%

+ Randomly assign treatment to individuals:

  $Y \perp T$

+ Unconfounded by design
+ Answers to all questions of interest by design ("on policy")
− Often small data set
− Limited generalizability, risk if participants are not representative of population
− Unethical in many cases

Observational data

e.g. Real world data

>99%

+ Often large and rich data set
− Most common case because:
  - Did not think of the question when data was created
  - Financial and reputational risk
  - Budget and time constraints
  - Potential problem with hidden confounding
− May not include all cases or results we're interested in ("off policy")

There are 2 key challenges we need to solve when working with observational data

Today's tutorial

1. Confounding

A **confounder** is a variable that influences both the treatment and the target

Confounding can **limit identifiability** of the causal effect

---

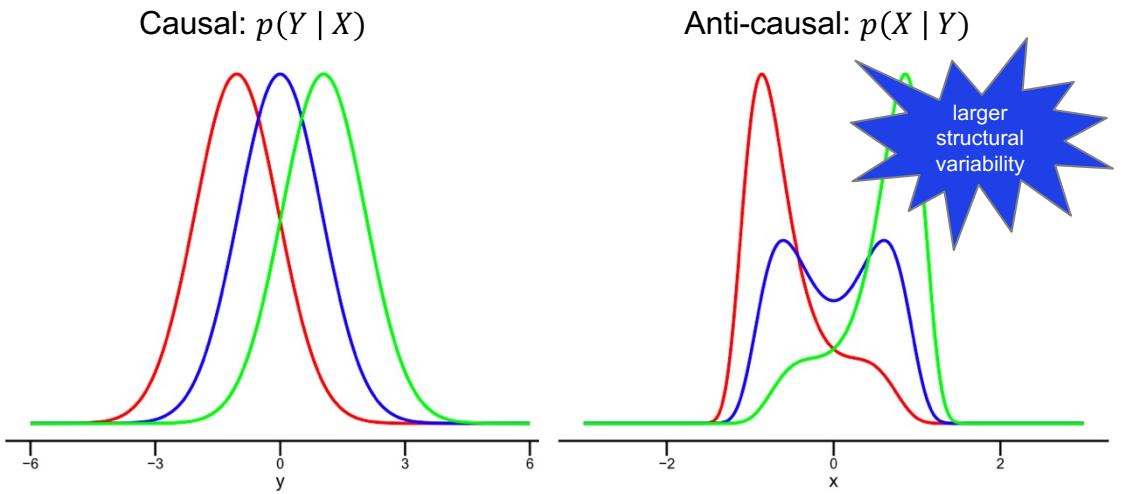
Suitable fixes

Match populations using **propensity score matching**

Obtain confidence intervals using **Causal Forests, BART**

Model relationships between all variables and encode subject matter expertise using **Bayesian Networks** or **structural equations**

2. Finding the causal direction



Eg, if $X \to Y$, $y|\, x \sim N(x^3 + \mathrm{x}, \sigma^2)$

"Causal Inference via Kernel Deviance Measures (KCDC)" postulates that sometimes a causal direction can be determined from distributions of the data.

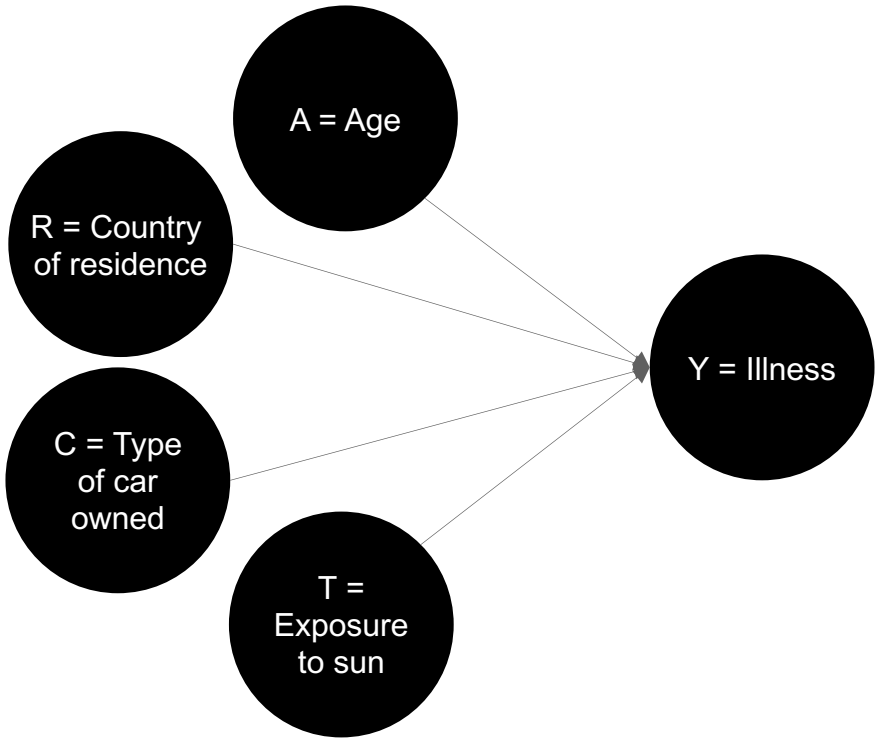
We can better identify the right intervention with graphical models

Establishing causal relationship is critical for us to recommend the right interventions

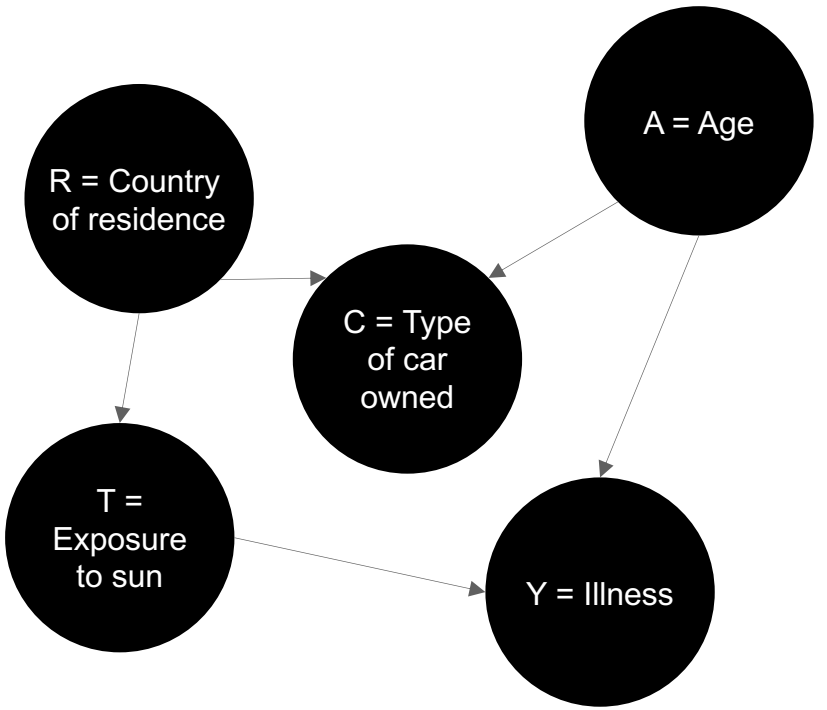
Traditional models, like linear regressions, have simplistic assumptions

We use graph models such as Bayesian Networks which can be more intuitive and allows domain expertise encoded with data to form a better understanding of relationships

Linear regression



Bayesian network



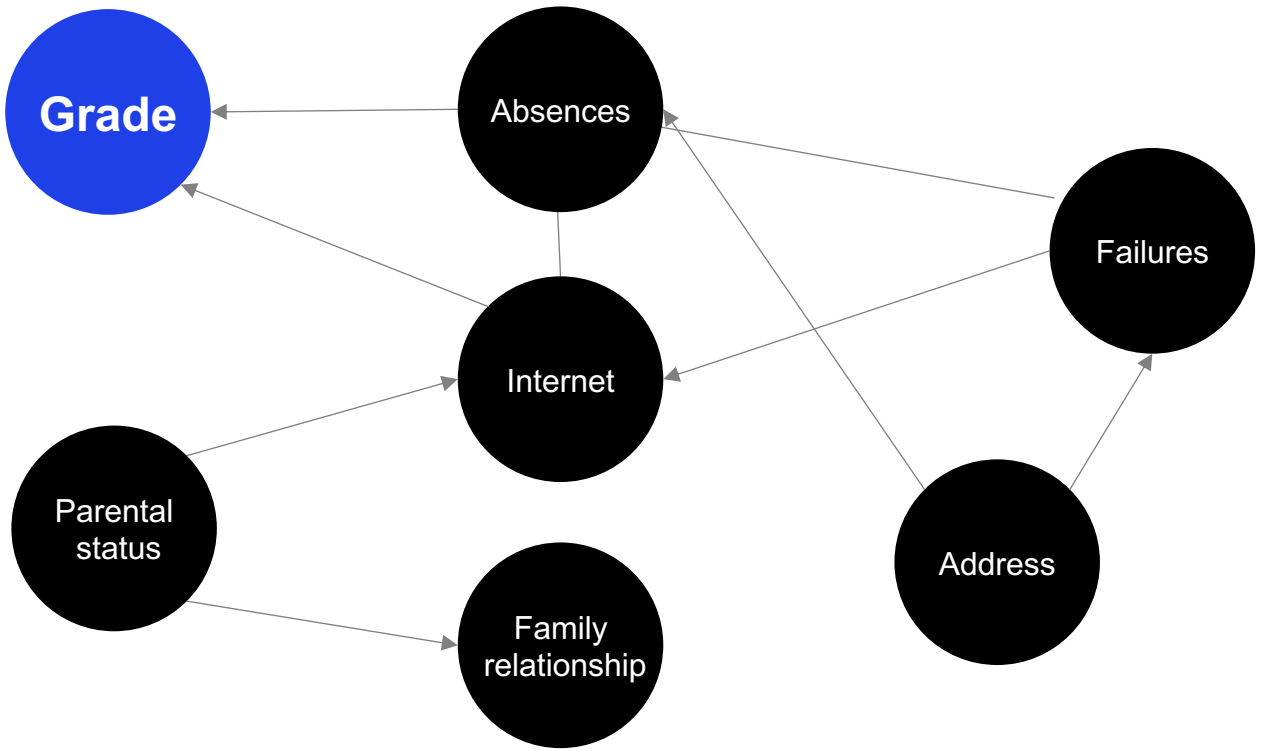
Bayesian networks

Use a **directed acyclic graph** to capture interdependencies between variables

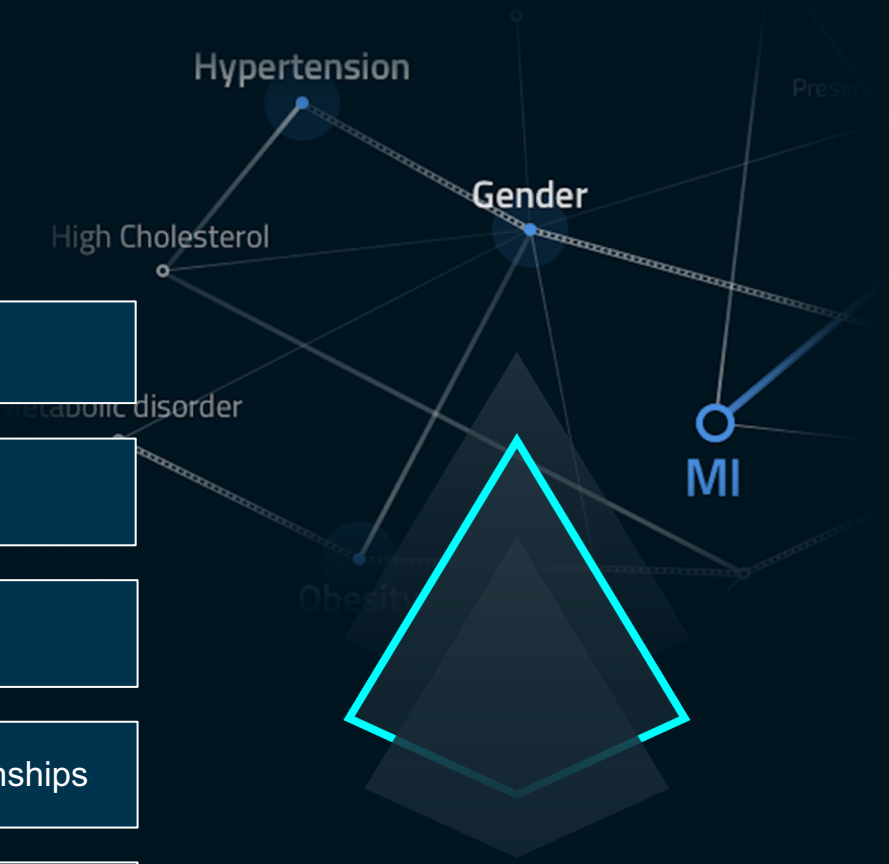
Nodes represent variables

Edges represent relationships between variables

- A → B = "B depends on A", or more precisely, *"The value of a node is independent of the rest of the variables in the graph given its parents."*



CausalNex is an **open-source** Python library that leverages **Bayesian Networks**



Structure learning – Learn relationships in data with NOTEARS, a state-of-the-art algorithm

Embed domain expertise – Enable experts to add and remove inaccurate relationships

Graph visualisation – Use extensions of NetworkX to help communicate results

Likelihood estimation – Estimate the probability distribution of variables based on their relationships

Counterfactual analysis – Infer what happens to *target Y* when *feature X* is changed

CausalNex

Powered by QuantumBlack

Status:

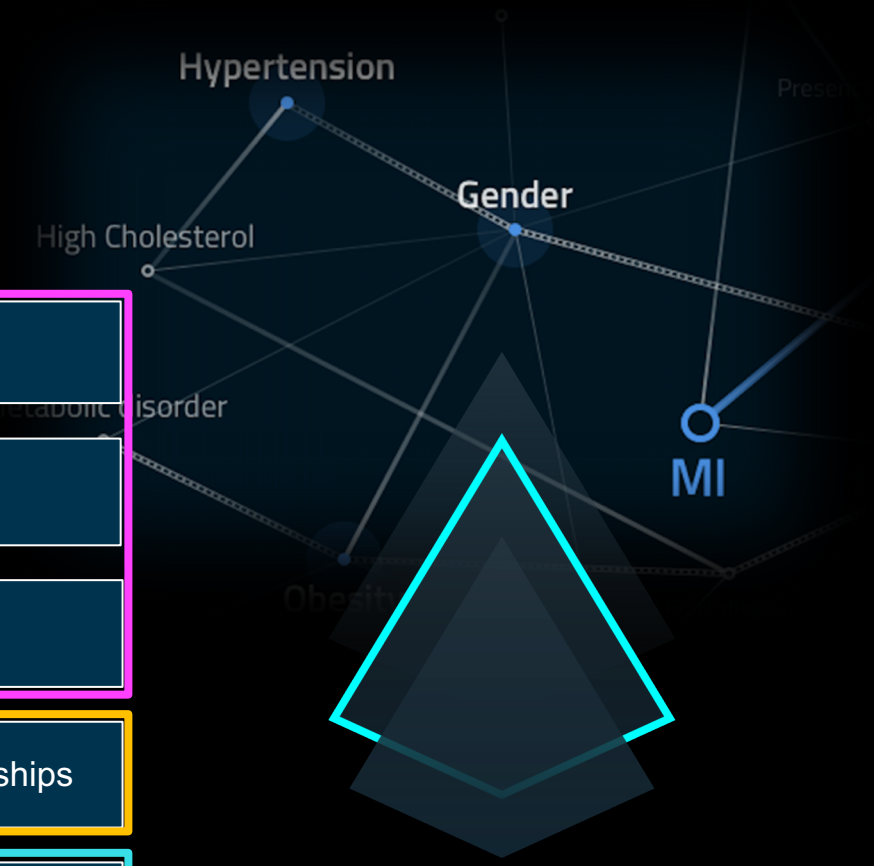
QuatumBlack Labs

PyPI

Read The Docs

`pip install causalnex`

CausalNex is an **open-source** Python library that leverages **Bayesian Networks**



Structure learning – Learn relationships in data with NOTEARS, a state-of-the-art algorithm

Embed domain expertise – Enable experts to add and remove inaccurate relationships

Graph visualisation – Use extensions of NetworkX to help communicate results

Likelihood estimation – Estimate the probability distribution of variables based on their relationships

Counterfactual analysis – Infer what happens to *target Y* when *feature X* is changed

CausalNex

Powered by QuantumBlack

Status:

QuatumBlack Labs

PyPI

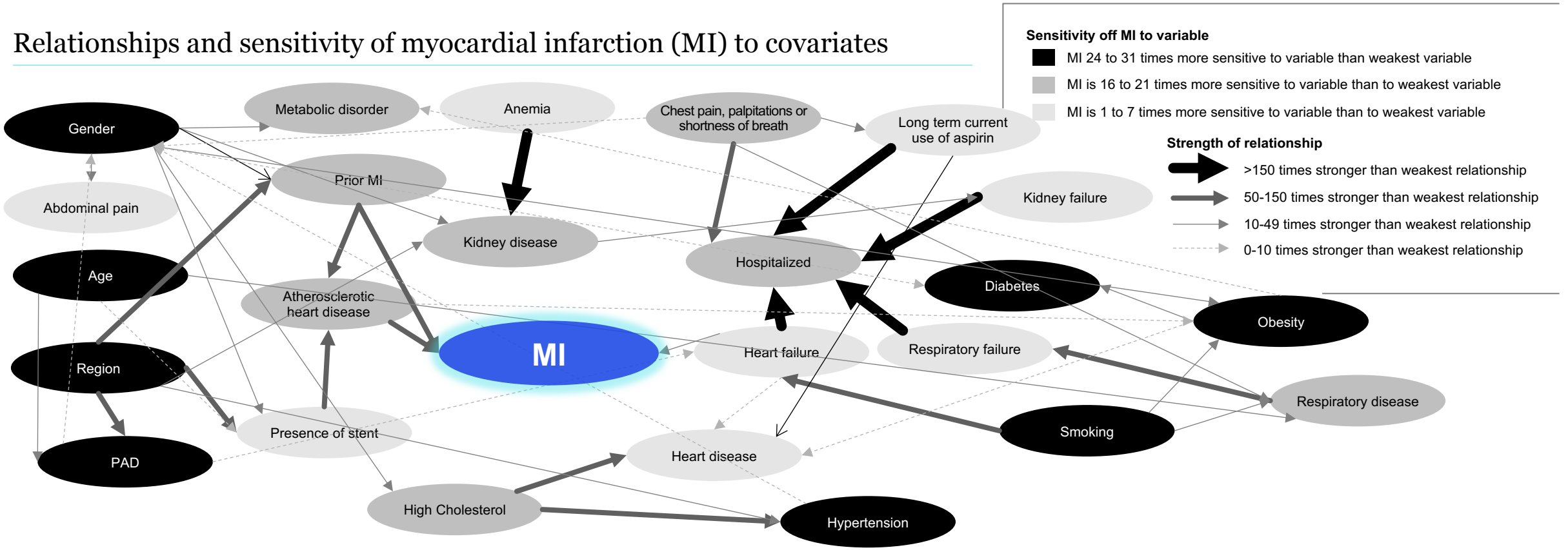
Read The Docs

`pip install causalnex`

Causal models can be used to support decision making in important domains such as healthcare

ILLUSTRATIVE

Relationships and sensitivity of myocardial infarction (MI) to covariates



The network structure is generated from both data and domain knowledge.

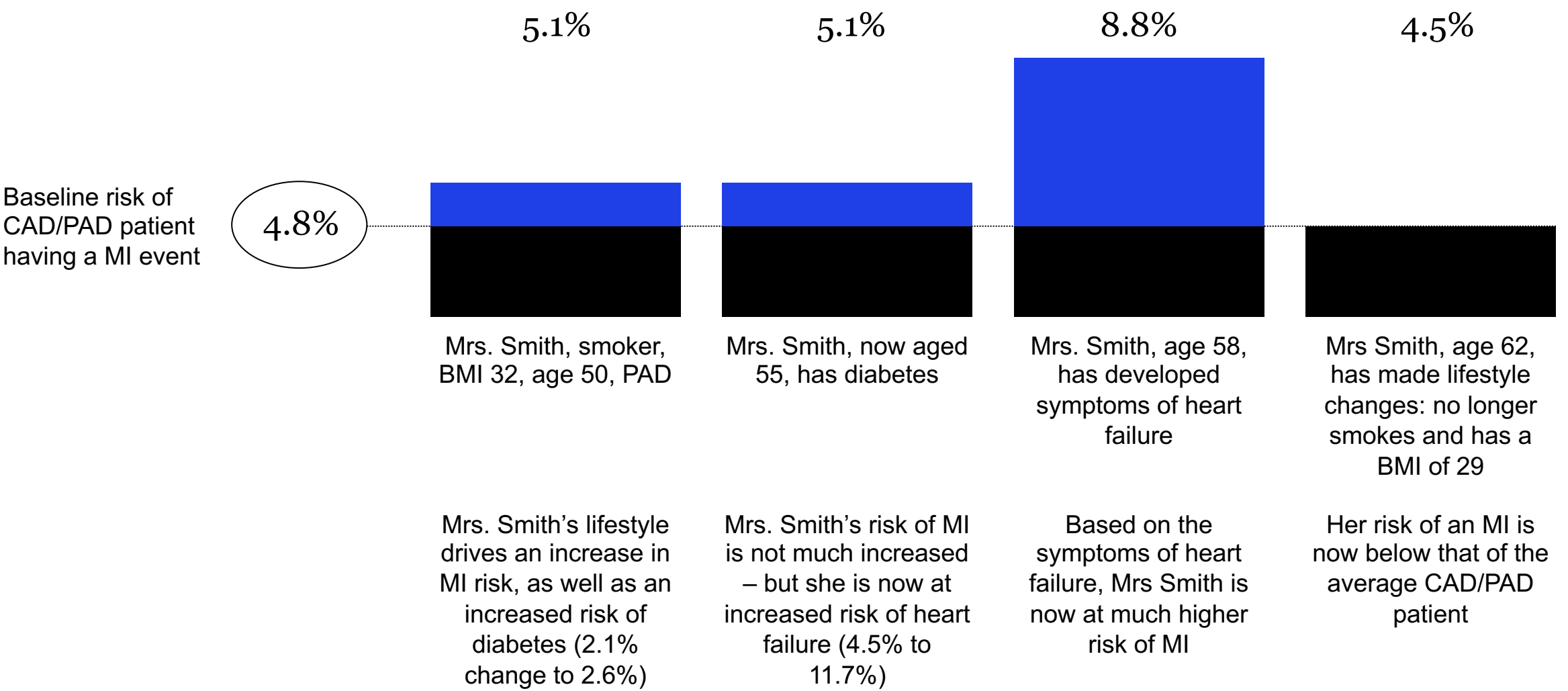
Incorporating domain expertise ensures the model represents a domain expert's view of causal relationships

Quantifying the relationship between patient demographics, comorbidities, and cardiovascular events can be used to identify key drivers of patient risk

Trained on Truven Claims data (2015 – 2017)
Structure learning uses DAGs with NO TEARS: Continuous Optimization for Structure Learning; Zheng et al.; NeurIPS 2018

With this approach we could better understand patient journeys

Risk of MI within 12 months[1]



SOURCE: Truven Claims Data (2105-2017)

Structure learning

1

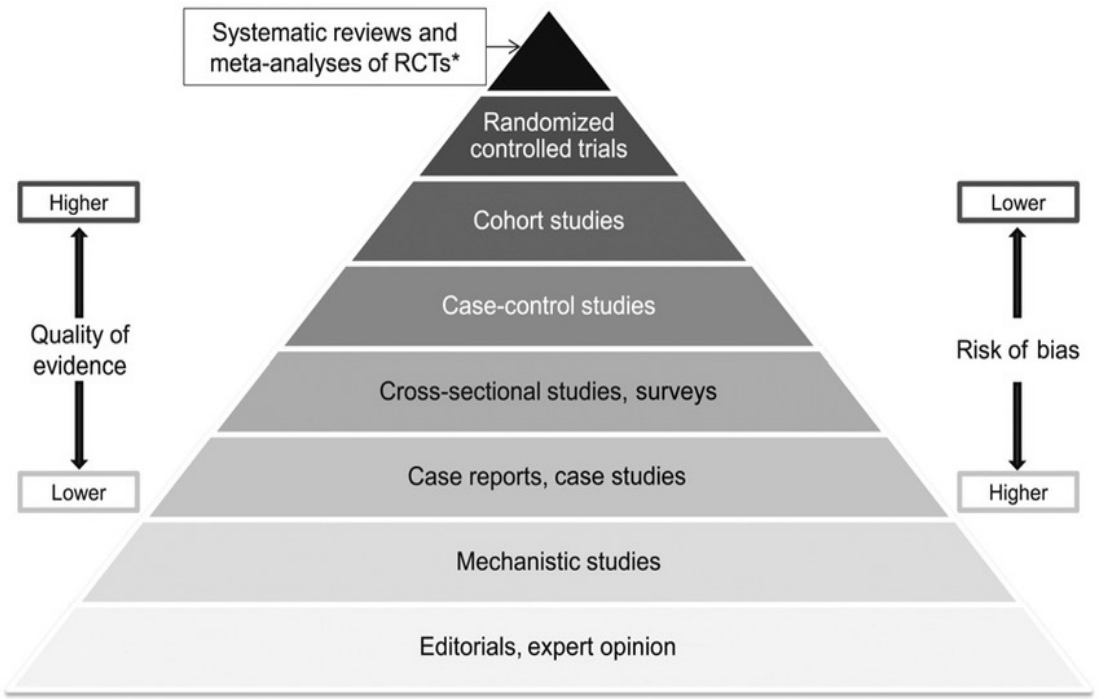
Defining the structure of a Bayesian network

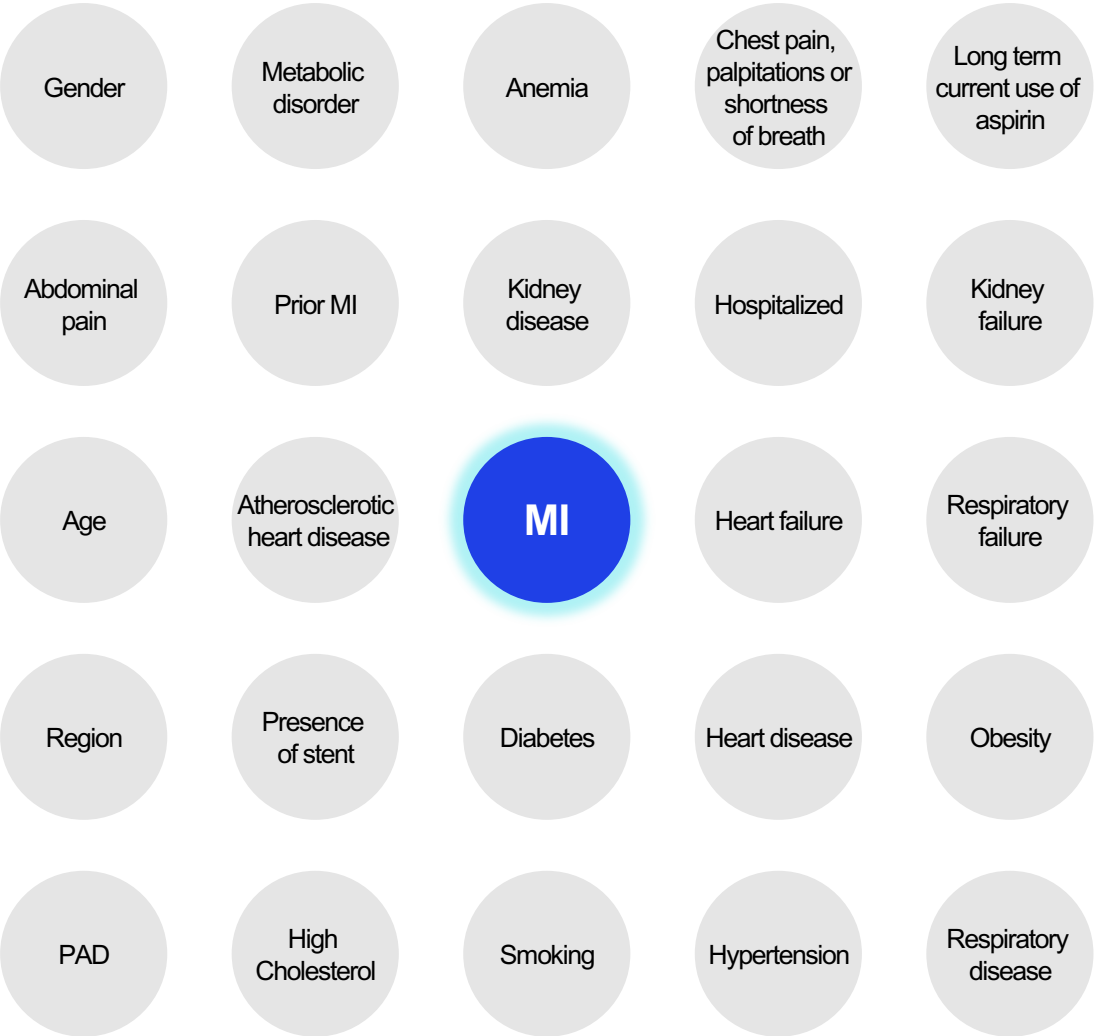
Domain Expertise

Present all variables to an expert

Ask them to tell us all relationships

Experts should consider evidence hierarchy





Yetley et. al. (2016) Options for basing Dietary Reference Intakes on chronic disease endpoints. AJoCN. 10.3945/ajcn.116.139097

Defining the structure of a Bayesian network

Domain Expertise

Present all variables to an expert

Ask them to tell us all relationships

Experts should consider evidence hierarchy

Challenges

How to deal with scale?

What if we miss some relationships?

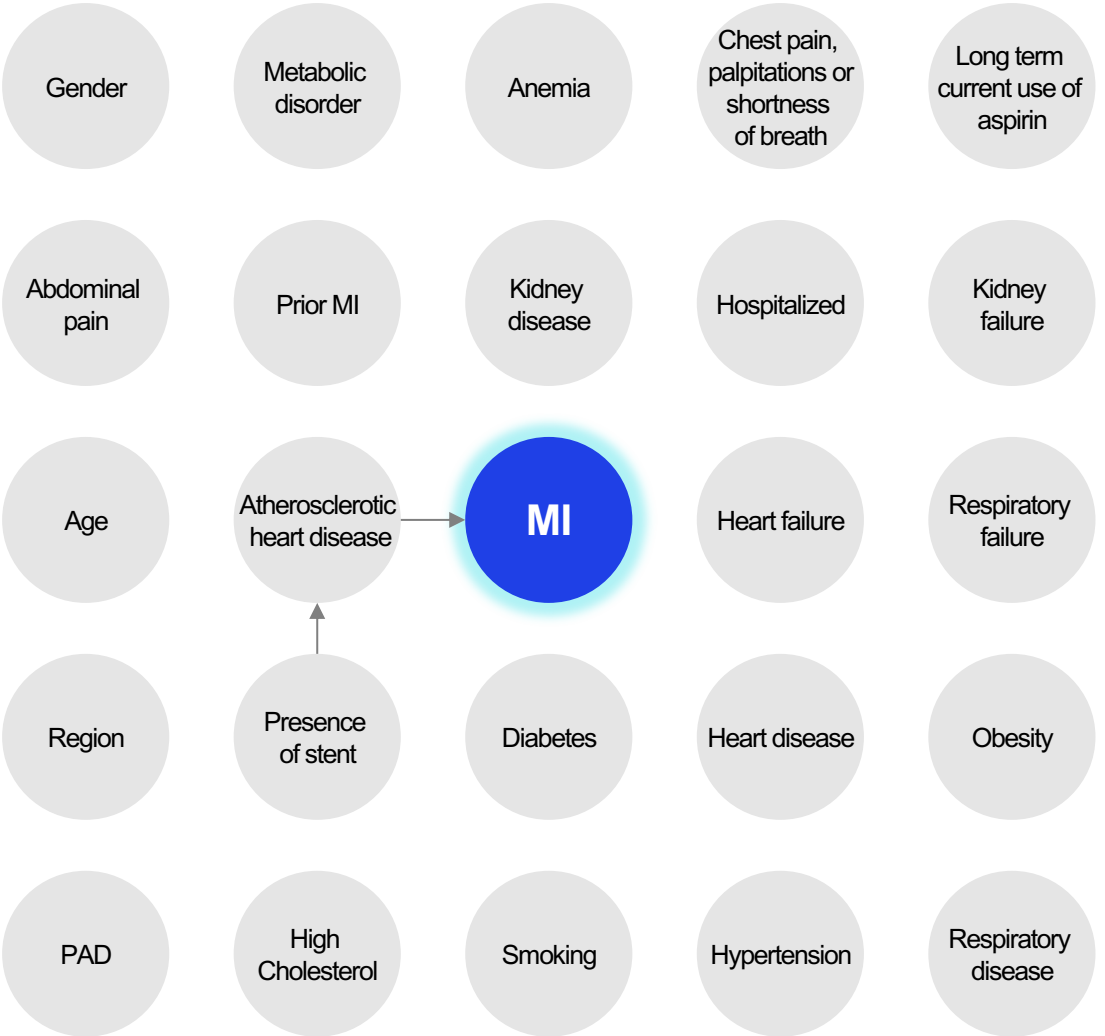
How to deal with higher-order causal effects?

We can use Machine Learning to help experts!

…

…

…



Structure learning

Evaluation

Search methods

Output type

	Evaluation	Search methods	Output type
Score-based	• Find graph that maximizes a specified score • Common scores: BIC, MDL, BDe, BDeu	• Greedy local search • Dynamic programming • Integer linear programming • Global continuous optimization	• *Directed acyclic graph* (DAG)
Constraint-based	• Start from complete graph, delete edges between nodes that are conditionally independent (CI), orient edges • Can choose CI criterion: Chi-squared test, G test	• Full search • Improved by testing CI in the right order	• Equivalence class of DAGs called *completed partially directed acyclic graph* (CPDAG)

Scanagatta et al. (2019). A survey on Bayesian network structure learning from data. Progress in Artificial Intelligence, 1-15.
Zhou (2011) Structure Learning of Probabilistic Graphical Models: A Comprehensive Survey. Arxiv: 1111.6925

Practical considerations for structure learning

NP-hard problem due to **large search space** and **combinatorial acyclicity constraint**

Data type: discrete, continuous, or mixed?

Time-varying data: do we need a dynamic Bayesian network?

Model complexity: linear model (needs less data) or nonlinear model (more flexible)?

Do we need to account for **confounders** or **missing data**?

CausalNex includes an implementation of **DAGs with NO TEARS,** a continuous optimization algorithm (Zheng et al.)

Formulation

- NOTEARS[1] is a score-based method. The objective is to optimize some score F(W) subject to the weighted adjacency matrix W corresponding to a DAG.
- The authors propose an approach to convert the combinatorial optimization problem (left) into a **continuous** problem (right):

$$\begin{array}{ll} \min\limits_{W \in \mathbb{R}^{d\times d}} & F(W) \\ \text{subject to} & \mathsf{G}(W) \in \mathsf{DAGs} \end{array} \iff \begin{array}{ll} \min\limits_{W \in \mathbb{R}^{d\times d}} & F(W) \\ \text{subject to} & h(W) = 0 \end{array}$$

- The loss function F incorporates a **smooth loss** and an **L1 regularization** term that encourages sparsity.
- Graph is a DAG if and only if trace(W^k) = 0 for all *k*:
- The key breakthrough is a **novel acyclicity constraint**.
- Graph is a DAG if and only if trace(W^k) = 0 for all *k,* or equivalently:

$$\sum_{k=1} \sum_i^d \frac{\left(W^{2k}\right)_{ii}}{k!} = Trace\left(e^{(W \odot W)}\right) - d = 0$$

- The authors use a **least-squares loss** based on a linear model, but any smooth loss is compatible with the approach.

$$F(W) = \ell(W; \mathbf{X}) + \lambda \|W\|_1 = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}W\|_F^2 + \lambda \|W\|_1.$$

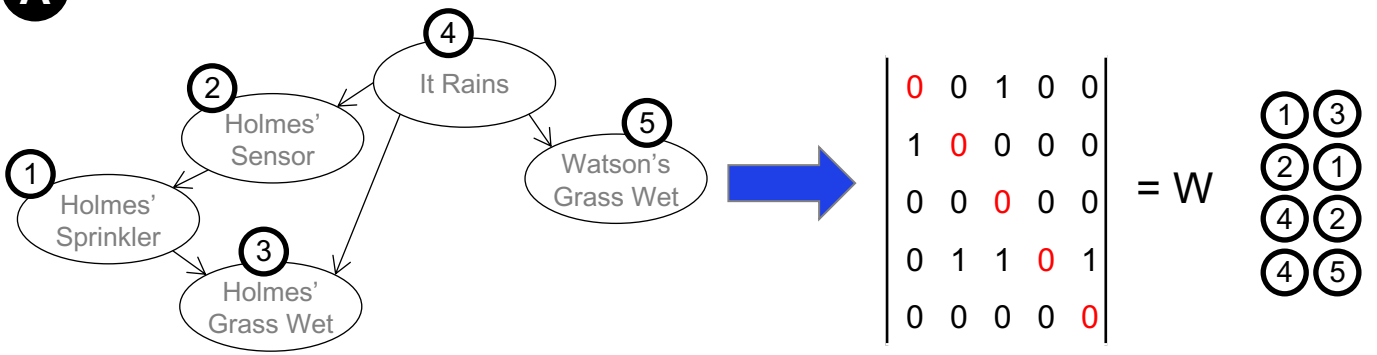
1 Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning

Zheng, Xun, et al. "DAGs with NO TEARS: Continuous optimization for structure learning." Advances in Neural Information Processing Systems. 2018.

CausalNex includes an implementation of DAGs with NO TEARS, a continuous optimization algorithm (Zheng et al.)

Previous techniques suffered because they needed to “check acyclicity holds” and this is a combinatorial optimization problem. The authors of *DAGs with NO TEARS (Zheng et al.)* convert this to a continuous test (that is faster and easier to incorporate into search algorithms), leveraging the properties of the adjacency matrix

A



B

$$\begin{vmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} = W^2$$

4 1
2 3

A The leading diagonal (or trace) of a DAG’s adjacency matrix, *W*, is all zeros.

B Raising *W* to a power, *k* will produce all possible paths *k* steps away. In a DAG, trace(W^k) = 0 for all *k*.

Trace(W^k) = 0 for all *k* is true iff:

$$\sum_{k=1} \sum_{i}^{d} \frac{(W^{2k})_{ii}}{k!} = trace\left(e^{(W \odot W)}\right) - d = 0 \ (< \epsilon)$$

https://arxiv.org/pdf/1803.01422.pdf

Structure learning does not guarantee causality

Structure learning algorithms make a best guess at direction – don't expect them to be correct

Always get experts to review the structure

Domain knowledge prior to structure learning

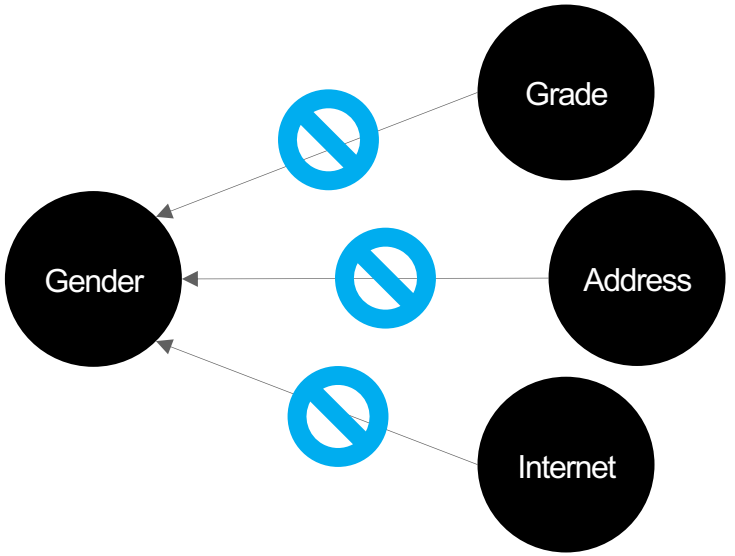
- Constrain search space via tabu / required edges

Domain knowledge after structure learning

- Add / remove / flip edges

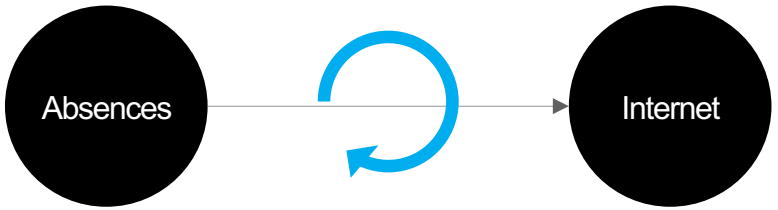
Prior to structure learning

Nothing should influence gender



After structure learning

This edge should be flipped



Practical 1

Let's explore the sensitivity of the structure learned to the parameters of the NOTEARS algorithm

Can you explore different configurations and find different results?

How do we validate our results anyway?

Can you set this up as a "parameter tuning" problem?

Discretisation & probability fitting

2

Each node has data that must be modelled by either a continuous or discrete distribution

Continuous

Often assumes Gaussian distribution

Discretized

Allows for data not normally distributed

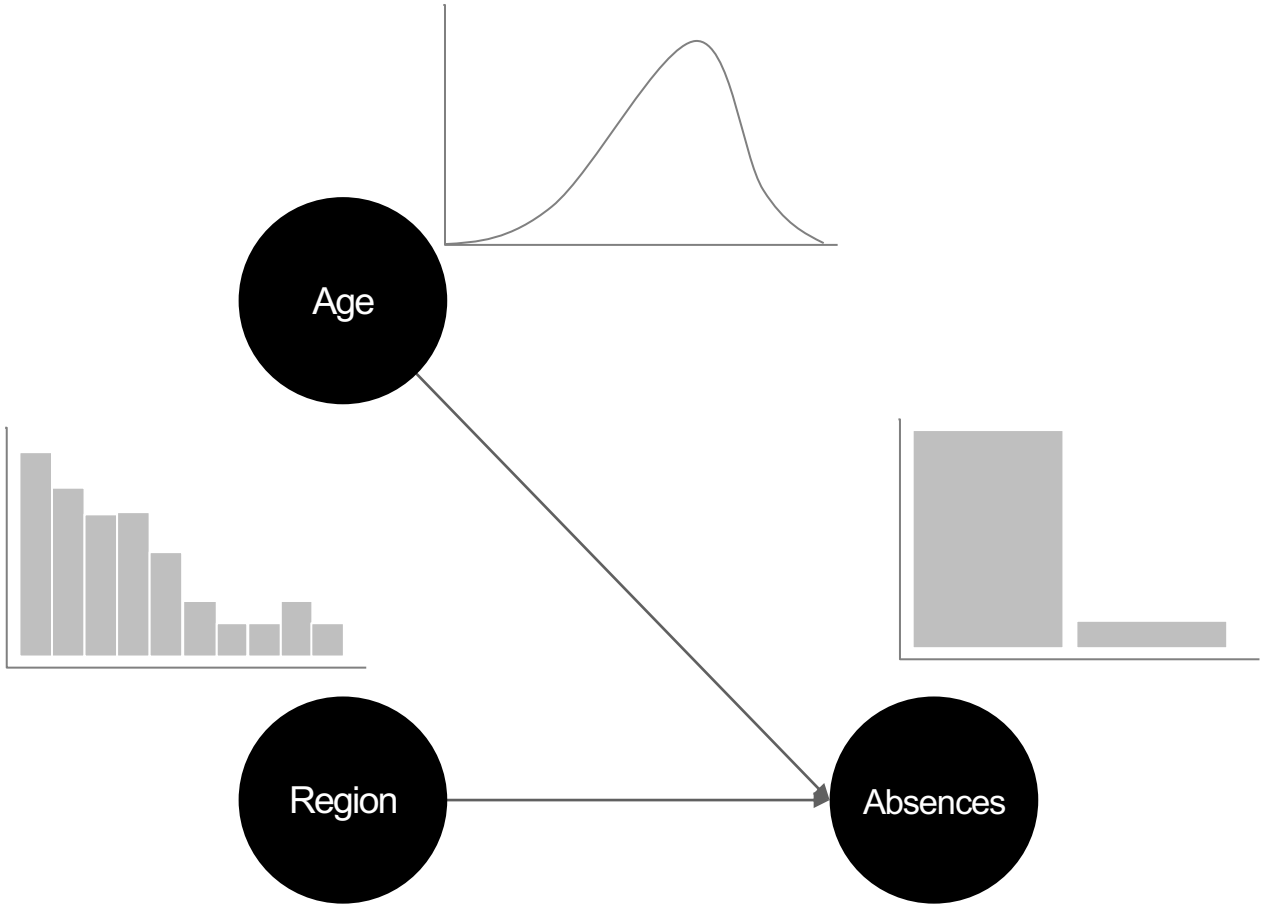
Flexible granularity, with continuous data bucketed as appropriate

Discrete or categorical data can also be grouped if no significant difference between classes

We Use Discretized

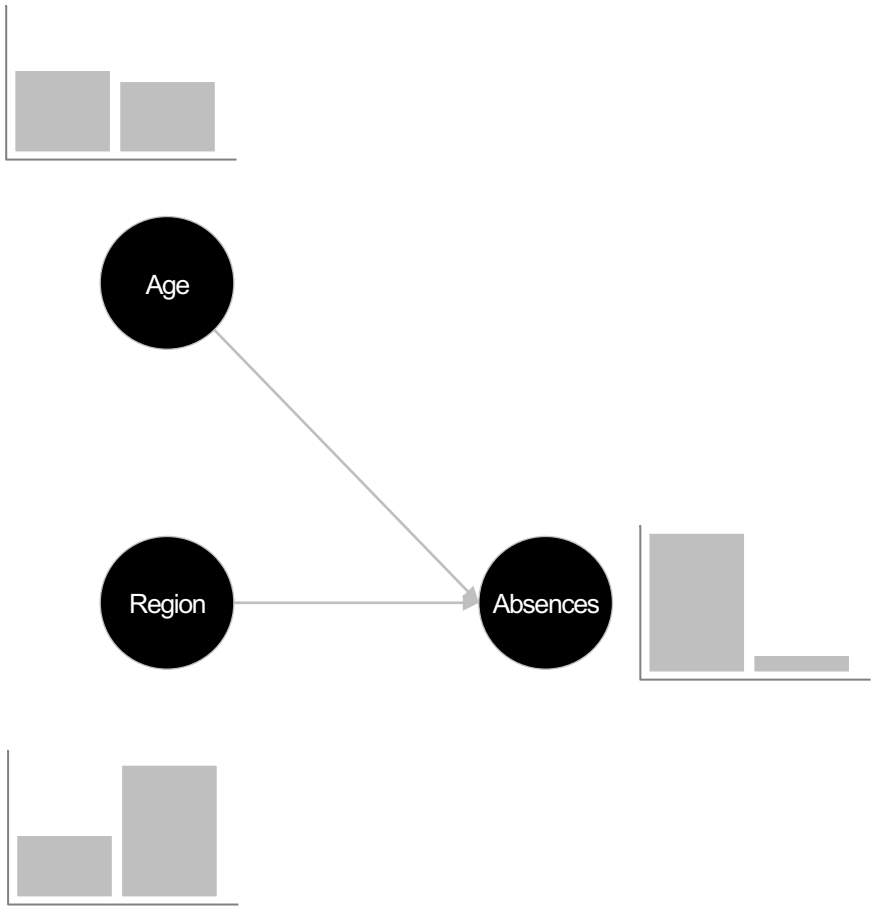
Real world data is not Gaussian

Flexibility and control over distributions



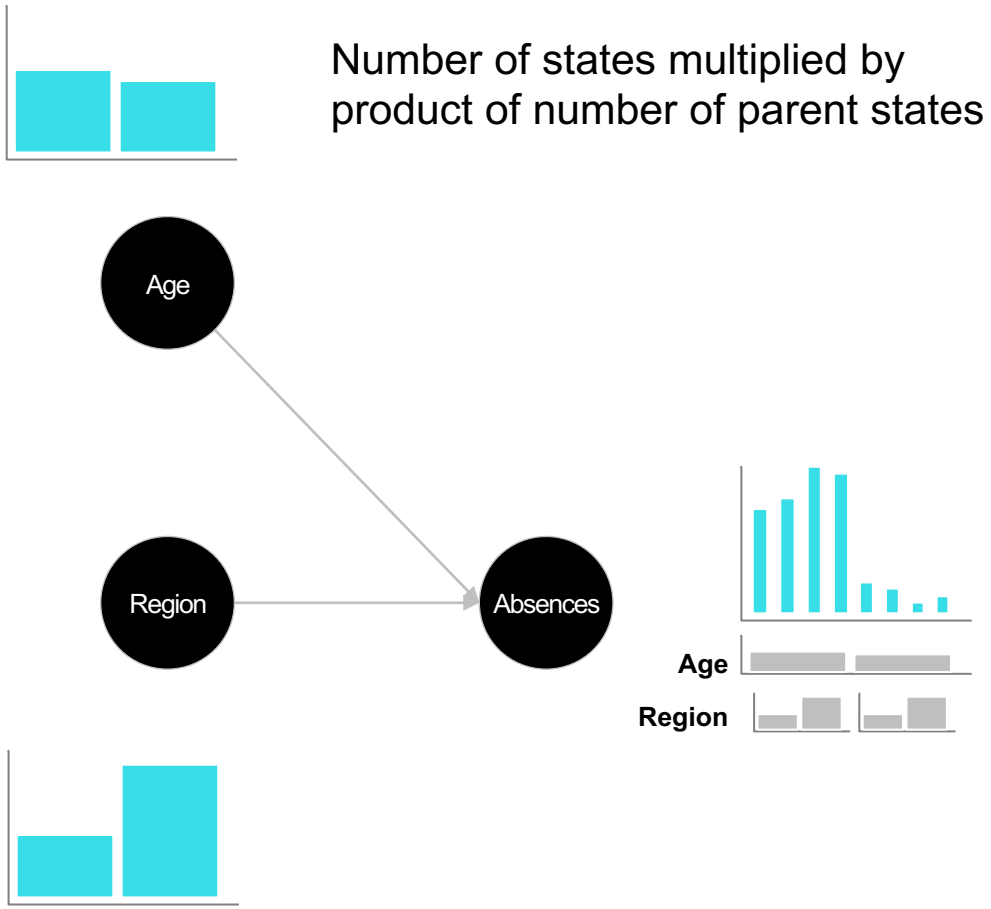
Behind every node in a Bayesian network is a conditional probability distribution (CPD)

Data distributions

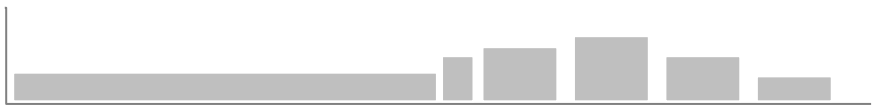


Conditional probability distributions

Number of states multiplied by product of number of parent states



Experts and data scientists should decide on an appropriate discretisation for each variable

Method	Example	When To Use	
			
Equal Width		Known linear mappings	days -> years
Percentile		Desire to make comparisons	top X% of earners
Outliers		Interested in boundary cases	unusually tall
Fixed		Replicate industry-standard buckets	% -> Grade

There are two main ways to learn discrete CPDs

Maximum likelihood estimation

Unable to encode prior beliefs

- Based on computing point estimates
- Assumes uniform prior distribution

Many Implementations

- Most stats / numeric / ML packages have a good implementation
- Many will return P(unobserved)=0

Bayesian estimation

Allows encoding of prior beliefs

- Based on computing PDFs
- Often expressed through pseudo-counts

Can provide better insights

- Allows interaction with experts when defining priors
- Implementations avoid P(unobserved)=0 issue

Bayesian networks can be evaluated using standard approaches for classification models

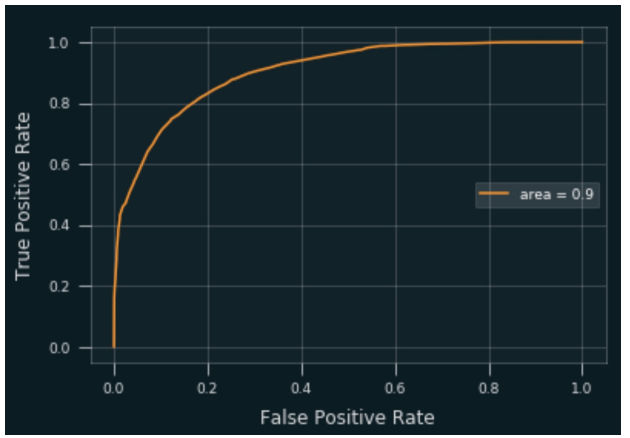
Prediction Test data example

- Learn CPDs using training set
- Predict any variable(s) from test set

	A	Low		High	
	B	Low	High	Low	High
Effect	False	0.9	0.4	0.8	0.3
	True	0.1	0.6	0.2	0.7

P(Effect=False | A=Low, B=High) = 0.4

P(Effect=True | A=Low, B=High) = 0.6



precision	recall	f1-score	support
0.846465	0.927764	0.885252	2478
0.669131	0.464698	0.548485	779
0.757798	0.696231	0.716869	3257
0.817010	0.817010	0.817010	3257
0.804051	0.817010	0.804705	3257

Practical 2

What happens when we don't have enough data to cover all options?

When will this get worse?

How can we try/help solve for this?

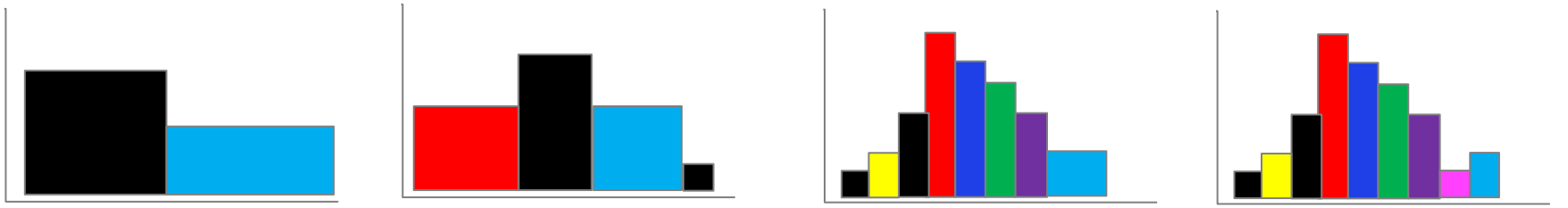
We can do some hacks to improve the model performance without increasing complexity

We can help our model pick up the structure of the model by discretizing features into useful buckets that explain variance in the target, (or, more generally, variance in other nodes a variable should impact, not just the target)

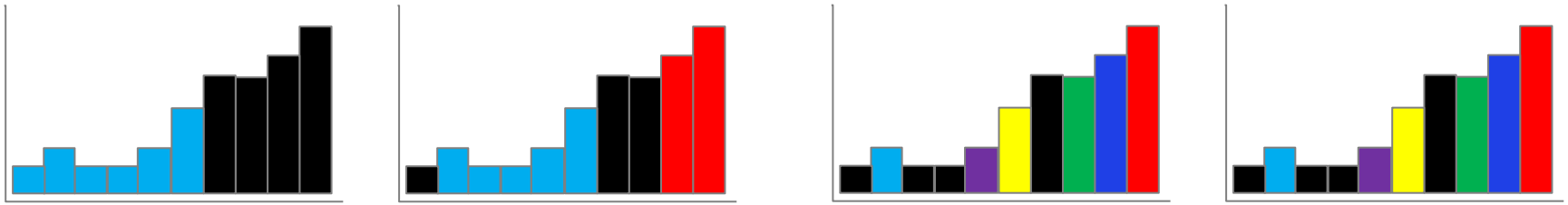
We can use the “MDLP algorithm” that calculates whether the entropy of the target distribution is affected sufficiently to justify a bucket split

Discretization continues whilst it helps ‘explain’ variance in the target

And stops when it no longer adds information



Discretization of feature: the algorithm identifies boundary points to split on and form distinct feature states



Distribution of target: more and more variance is explained as the feature is discretized further

Splitting the feature variable from  into  does not help predicting  and so the algorithm stops and uses 

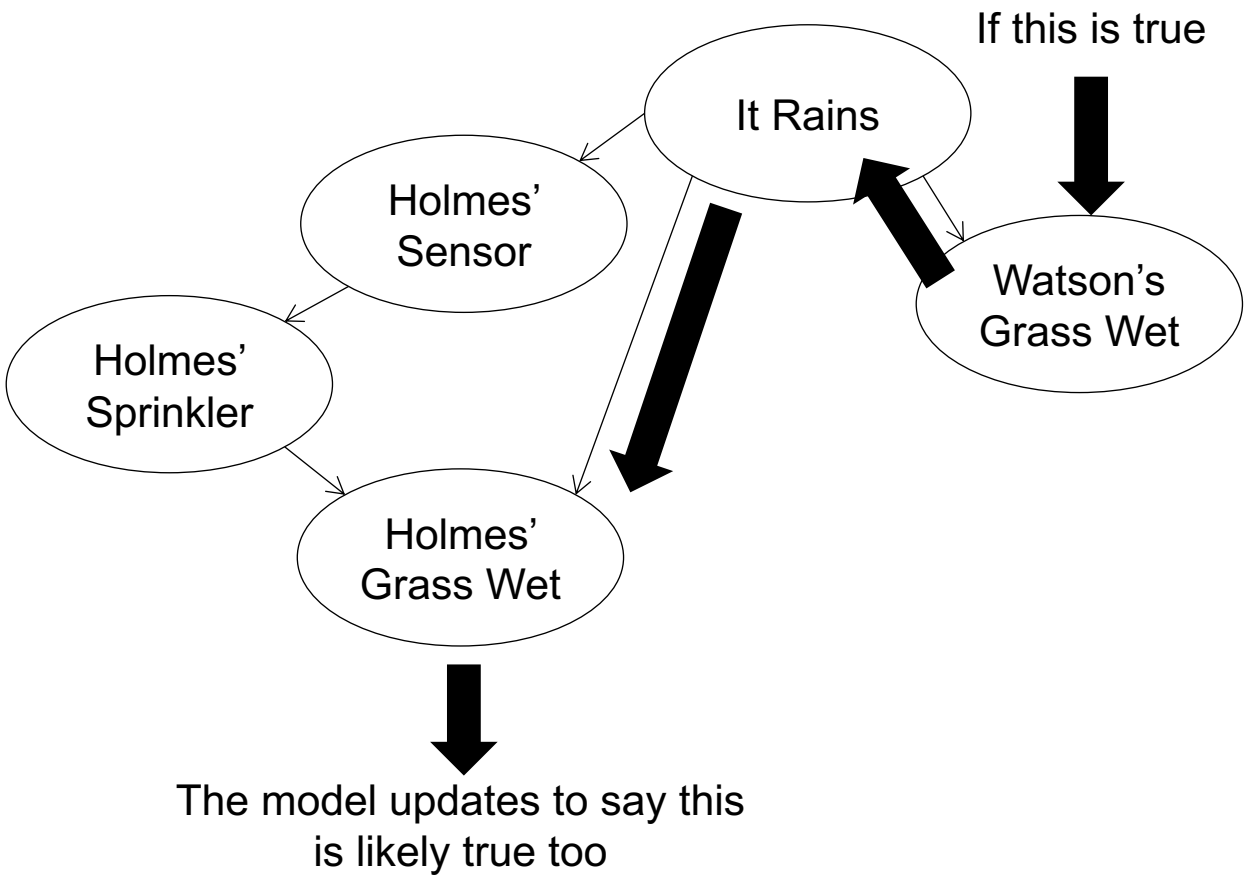
Inference & interventions

3

Learned models can be used to identify the 'most important' relationships between variables and update when given new evidence

How can we use Bayesian Networks?

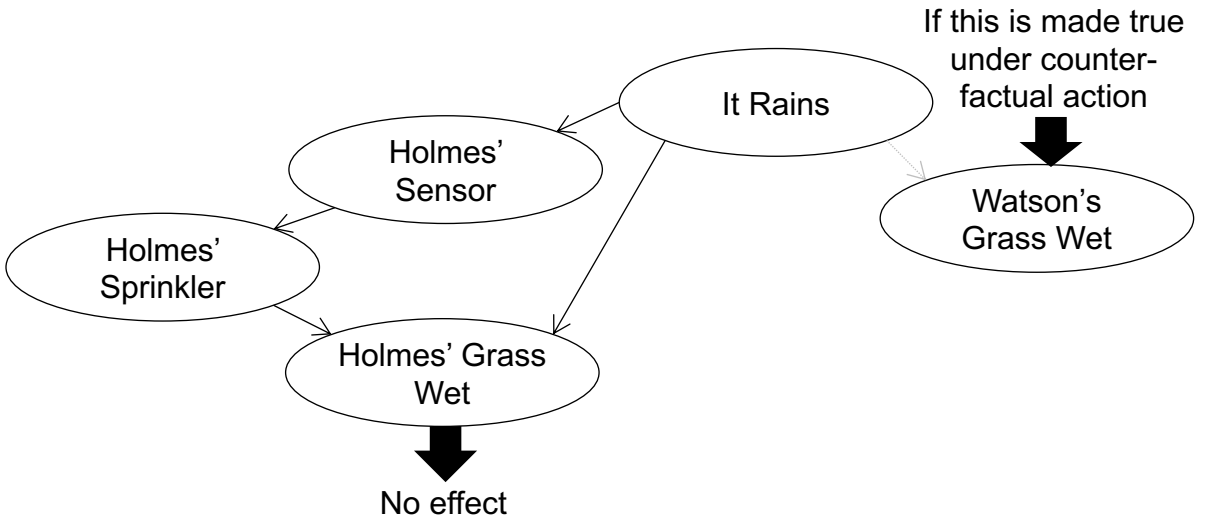
- The probabilities of variables in Bayesian Networks **update as observations are added to the model.** This is useful for inference, and for beginning predictive analytics
- Metrics can help us **understand the strength of relationships between variables** and identify key drivers of change. These will be nodes that are most valuable to target in interventions
- We can leverage the fact that variables interact with each other to **run advanced value-at-stake (counterfactual) analysis.** This assesses the combined effect of actions without making the naive assumption that any two actions are independent.

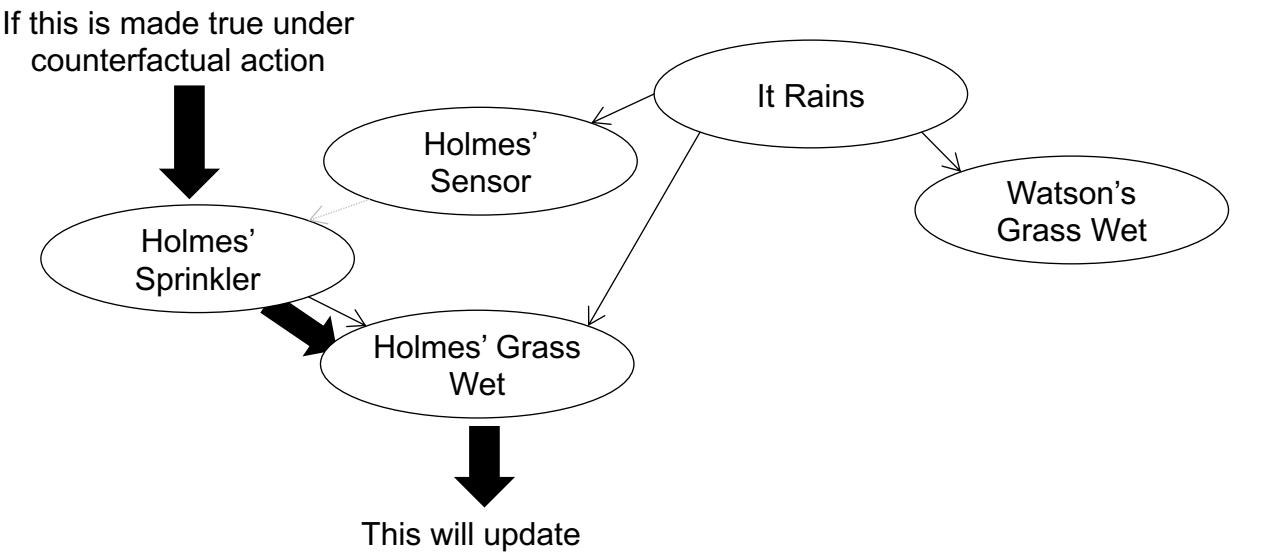


Counterfactual actions are different to inference, and lead to different results

'Conditioning' versus 'Doing'

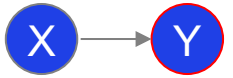
- An external intervention that makes Watson's grass wet (e.g. Watson's watering can) has no effect on whether it rained, which is different to conditioning and the outcome of the previous slide.
- Interventions effect variables' dependencies though: if Holmes' sprinkler is definitely set on, Holmes' Grass Wet marginal will change. In this instance counterfactual is same as conditioning.
- This is still not generally the same as conditioning, as probabilities don't propagate to update parent nodes. If Holmes sprinkler and it rains shared a common parent, then counterfactual wouldn't update it and thus would be different from conditioning





The differences between observational and interventional inference can be best viewed through SCM notation

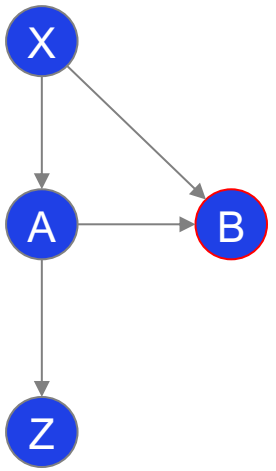
Original variable
Observational change
Interventional change
Updated distribution

For a causal model, where $X \rightarrow Y$ is denoted 

observing evidence asks the model to update likelihoods of variables throughout the model based on an observation.

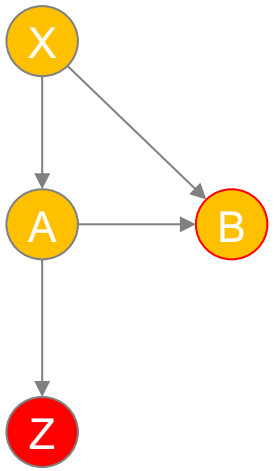
If Z is observed to have a specific value, what can we infer about the likely values *A, B* and *X* had at this point? The distributions of *A, B* and *X* all update given this observational information.

If we intervene on *Z*, the causal direction states that *A* (nor any other nodes) are impacted, and we "break" the link between *A* and *Z*. Distributions of *A, B* and *X* (in this instance) remain unchanged, because *Z* does not "cause" any of them, and so intervening on *Z* is folly.



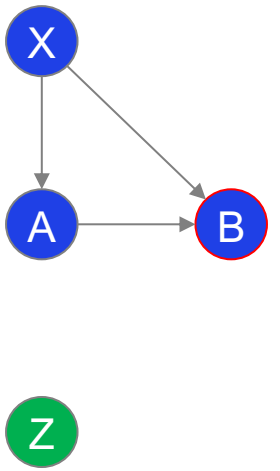
$P(B)$

X causes A and B, A causes B and Z. We are interested in B.



$P(B|Z = z)$

Conditioning on Z updates the network despite causal direction



$P(B|do(Z = z))$

Intervening on Z respects causal direction and has no impact

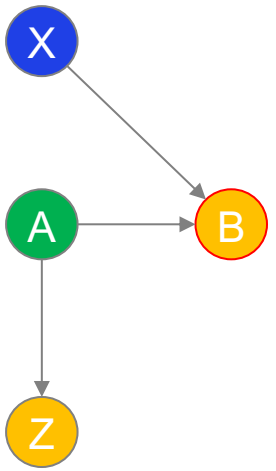
The differences between observational and interventional inference can be best viewed through SCM notation

Original variable
Observational change
Interventional change
Updated distribution

Intervening on A “breaks” its dependence from X (and doesn’t update the distribution of X, unlike conditioning on A). B and Z are both descendents of A, and their marginal probabilities would change due to the change in A.

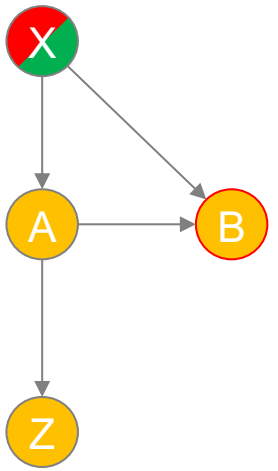
For nodes with no parents, intervening on them is identical to receiving observational update.

We can combine interventions with observational data. For observation in X and intervention on A, B would update to reflect changes in both A and X. A still stops depending on X because of the intervention, and Z updates based only on the intervention change to A, and not due to updates to X.



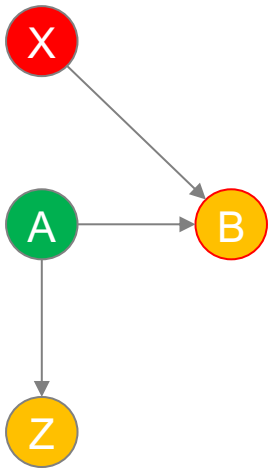
$P(B|do(A = a))$

Intervening on A updates dependents but is not the same as conditioning on A



$P(B|X = x) = P(B|do(X = x))$

Intervening on X is the same as conditioning in this instance



$P(B|X = x, do(A = a))$

We can intervene whilst adding observational evidence

Practical 3

Can you think of how we might find the "optimal" intervention to make?

Close

4

Bayesian Networks complement conventional modelling techniques and perhaps supersede their capabilities in some areas

Advantages

Bayesian Networks offer a graphical representation that is **reasonably interpretable and easily explainable**

Models can reflect both statistically significant information (learned from the data) and domain expertise simultaneously. Metrics can measure the significance of relationships and help identify the effect of specific actions

Relationships captured between variables in a Bayesian Network are more complex yet hopefully **more informative than a conventional model**

Counterfactual actions combine without naive independence assumptions

Considerations

This is **not a way of automatically perfectly identifying causal relationships**, but it can help a human explore this

Computational considerations limit the number of variables in a BN (max ~30)

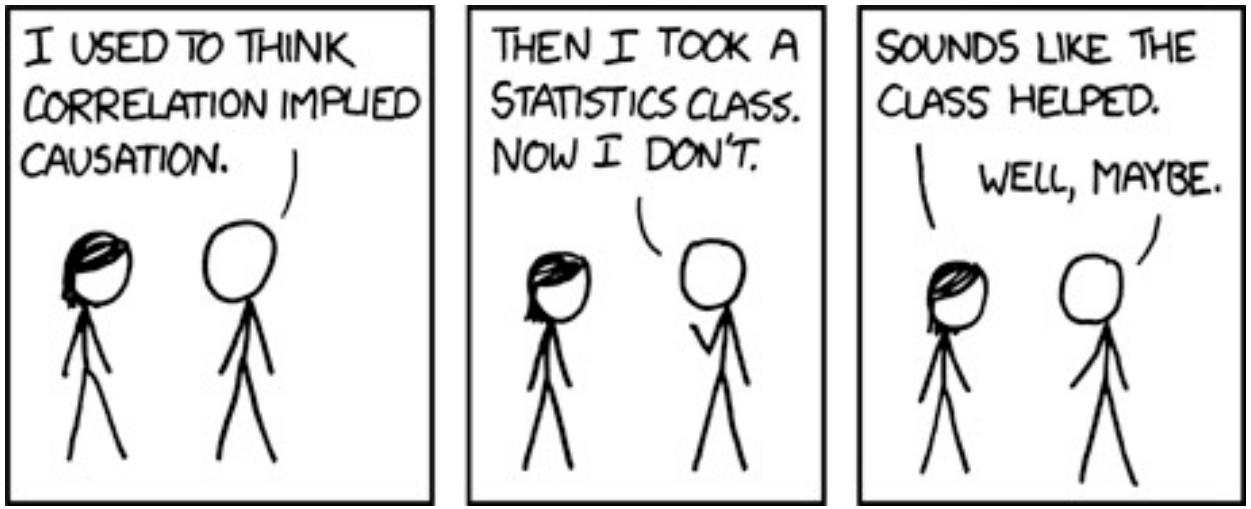
Takeaways

If we want to trust models for decisions, then we should expect them to make **causal sense**

Training on observational data is common, and the causal direction of relationships is not always clear

Methods exist to **help us identify possible causal relationships**, but domain experts can also help

Models that respect causality also exist and thanks to recent advances are now easier to learn and deploy



Comic: https://xkcd.com/552/

Any Questions?

Dr Paul Beaumont

paul.beaumont@quantumblack.com
linkedin.com/in/pbeaumont/