Non-probabilistic Markov Categories for Causal Modeling in Machine Learning

Dhurim Cakiqi[1] Max A. Little[1,2]

[1]School of Computer Science,
University of Birmingham, UK

[2]Media Lab,
MIT,USA

Applied Category Theory, 2022

Outline

Outline

The limitation of machine learning

- Machine learning (ML) is very useful in modeling large data sets
- Typically machine learning is non-probabilistic
- This is a "causally-blind" approach to modeling
- We need a method which takes into account the possibility of spurious correlations in the data

The limitation of machine learning

- Machine learning (ML) is very useful in modeling large data sets
- Typically machine learning is non-probabilistic
- This is a "causally-blind" approach to modeling
- We need a method which takes into account the possibility of spurious correlations in the data

The limitation of machine learning

- Machine learning (ML) is very useful in modeling large data sets
- Typically machine learning is non-probabilistic
- This is a “causally-blind” approach to modeling
- We need a method which takes into account the possibility of spurious correlations in the data

The limitation of machine learning

- Machine learning (ML) is very useful in modeling large data sets
- Typically machine learning is non-probabilistic
- This is a “causally-blind” approach to modeling
- We need a method which takes into account the possibility of spurious correlations in the data

The limitation of machine learning

- Machine learning (ML) is very useful in modeling large data sets
- Typically machine learning is non-probabilistic
- This is a "causally-blind" approach to modeling
- We need a method which takes into account the possibility of spurious correlations in the data

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \rightarrow B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P\left(X_1, \ldots, X_n\right) = \prod_{n=1}^{N} P\left(X_n | \text{pa}\left(X_n\right)\right)$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P\left(Y | \text{do}\left(X\right)\right)$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P(X_1, \ldots, X_n) = \prod_{n=1}^{N} P(X_n | \mathrm{pa}(X_n))$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P(Y | \mathrm{do}(X))$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P(X_1, \ldots, X_n) = \prod_{n=1}^{N} P(X_n | \text{pa}(X_n))$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P(Y|\text{do}(X))$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P(X_1, \ldots, X_n) = \prod_{n=1}^{N} P(X_n | \mathrm{pa}(X_n))$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P(Y|\mathrm{do}(X))$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P(X_1, \ldots, X_n) = \prod_{n=1}^{N} P(X_n|\text{pa}(X_n))$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P(Y|\text{do}(X))$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P(X_1, \dots, X_n) = \prod_{n=1}^{N} P(X_n | \text{pa}(X_n))$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P(Y|\text{do}(X))$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

What is it?

- It is a probabilistic method in which to model the causal influence of events on another
- Typically, it is represented by a directed acyclic graph (DAG)
- $A \to B$ represents A having a causal influence on B
- Given an arbitrary DAG the probability distribution is given by

$$P\left(X_1, \ldots, X_n\right) = \prod_{n=1}^{N} P\left(X_n | \mathrm{pa}\left(X_n\right)\right)$$

- In causal modeling the do-operator is used to simulate experimental interventions by changing the structure of the DAG
- The interventional distribution $P\left(Y | \mathrm{do}\left(X\right)\right)$ is computed after replacing all incoming arrows into X with the constant $X = x$.

Causal modeling

Back-door adjustment formula

- Suppose we were interested in the direct causal effect of a drug on a patient's health, and there is a confounder in the age of the patient
- We can represent this in a DAG by,

$$H \leftarrow A \rightarrow D$$

$$D \rightarrow H$$

- In this case we make use of the *back-door adjustment* formula

$$P\left(H|\text{do}\left(D=d\right)\right)=\sum_{a\in\Omega_A}P\left(H|A=a,D=d\right)P\left(A=a\right).$$

Causal modeling

Back-door adjustment formula

- Suppose we were interested in the direct causal effect of a drug on a patient's health, and there is a confounder in the age of the patient
- We can represent this in a DAG by,

$$H \leftarrow A \rightarrow D$$

$$D \rightarrow H$$

- In this case we make use of the *back-door adjustment* formula

$$P\left(H|\text{do}\left(D=d\right)\right)=\sum_{a\in\Omega_A}P\left(H|A=a,D=d\right)P\left(A=a\right).$$

Causal modeling

Back-door adjustment formula

- Suppose we were interested in the direct causal effect of a drug on a patient's health, and there is a confounder in the age of the patient
- We can represent this in a DAG by,

$$H \leftarrow A \rightarrow D$$

$$D \rightarrow H$$

- In this case we make use of the *back-door adjustment* formula

$$P(H|\text{do}(D=d)) = \sum_{a\in\Omega_A} P(H|A=a, D=d)\, P(A=a).$$

Causal modeling

Back-door adjustment formula

- Suppose we were interested in the direct causal effect of a drug on a patient's health, and there is a confounder in the age of the patient
- We can represent this in a DAG by,

$$H \leftarrow A \rightarrow D$$

$$D \rightarrow H$$

- In this case we make use of the *back-door adjustment* formula

$$P\left(H|\text{do}\left(D=d\right)\right)=\sum_{a\in\Omega_A}P\left(H|A=a,D=d\right)P\left(A=a\right).$$

Causal modeling

Front-door adjustment formula

- Suppose we were interested in the mechanism by which smoking affects lung cancer, we assume that smoking, X, affects tar in the lungs, M, which affects cancer in lungs Y, in this model we also include an unobserved variable U
- We can represent the DAG as

$$Y \leftarrow U \rightarrow X$$

$$X \rightarrow M \rightarrow Y$$

- In this case we make use of the *front-door adjustment* formula

$$P\left(Y|do\left(X=x\right)\right)=\sum_{m\in\Omega_M}P\left(M=m|X=x\right)\sum_{x'\in\Omega_X}P\left(Y|M=m,X'=x'\right)P\left(X'=x'\right).$$

Causal modeling

Front-door adjustment formula

- Suppose we were interested in the mechanism by which smoking affects lung cancer, we assume that smoking, X, affects tar in the lungs, M, which affects cancer in lungs Y, in this model we also include an unobserved variable U
- We can represent the DAG as

$$Y \leftarrow U \rightarrow X$$

$$X \rightarrow M \rightarrow Y$$

- In this case we make use of the *front-door adjustment* formula

$$P(Y|do(X=x)) = \sum_{m\in\Omega_M} P(M=m|X=x) \sum_{x'\in\Omega_X} P\left(Y|M=m, X'=x'\right) P\left(X'=x'\right).$$

Causal modeling

Front-door adjustment formula

- Suppose we were interested in the mechanism by which smoking affects lung cancer, we assume that smoking, X, affects tar in the lungs, M, which affects cancer in lungs Y, in this model we also include an unobserved variable U
- We can represent the DAG as

$$Y \leftarrow U \rightarrow X$$

$$X \rightarrow M \rightarrow Y$$

- In this case we make use of the *front-door adjustment* formula

$$P\left(Y|do\left(X=x\right)\right)=\sum_{m\in\Omega_M}P\left(M=m|X=x\right)\sum_{x'\in\Omega_X}P\left(Y|M=m,X'=x'\right)P\left(X'=x'\right).$$

Causal modeling

Front-door adjustment formula

- Suppose we were interested in the mechanism by which smoking affects lung cancer, we assume that smoking, X, affects tar in the lungs, M, which affects cancer in lungs Y, in this model we also include an unobserved variable U
- We can represent the DAG as

$$Y \leftarrow U \rightarrow X$$

$$X \rightarrow M \rightarrow Y$$

- In this case we make use of the *front-door adjustment* formula

$$P\left(Y|do\left(X=x\right)\right)=\sum_{m\in\Omega_M}P\left(M=m|X=x\right)\sum_{x'\in\Omega_X}P\left(Y|M=m,X'=x'\right)P\left(X'=x'\right).$$

Outline

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1], +, \times, 0, 1)$

Example

(Min-plus semifield). $(\mathbb{R}^{+}, \min, +, \infty, 0)$

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1], +, \times, 0, 1)$

Example

(Min-plus semifield). $(\mathbb{R}^{+}, \min, +, \infty, 0)$

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1],+,\times,0,1)$

Example

(Min-plus semifield). $(\mathbb{R}^{+},\min,+,\infty,0)$

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1], +, \times, 0, 1)$

Example

(Min-plus semifield). $(\mathbb{R}^{+}, \min, +, \infty, 0)$

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1], +, \times, 0, 1)$

Example

(Min-plus semifield). $(\mathbb{R}^{+}, \min, +, \infty, 0)$

The need for semifields

- In order to normalize we need an algebra that has a multiplicative inverse and for our purposes additive inverses are not required
- Semifields are critical so that we can form a consistent set of rules in order to manipulate casual morphisms
- Semifields can be used to represent machine learning models, e.g. ML and deep learning (DL) algorithms.

Example

(Probability semifield). $([0,1], +, \times, 0, 1)$

Example

(Min-plus semifield). $(\mathbb{R}^+, \min, +, \infty, 0)$

Semifields

Definition

A semifield $(S, \oplus, \otimes, i_\oplus, i_\otimes)$, is a set, S, endowed with two binary operations $\oplus, \otimes$ that satisfy the following conditions;

1. $\oplus$ is associative and commutative on all $a, b \in S$ and has identity $i_\oplus$ i.e. $\forall a \in S, i_\oplus \oplus a = a \oplus i_\oplus = a$
2. $\otimes$ is associative and has identity $i_\otimes$ i.e. $\forall a \in S, i_\otimes \otimes a = a \otimes i_\otimes = a$, and for all $a \in S \setminus \{i_\oplus\}$ there exists an inverse a^{-1} such that $a \otimes a^{-1} = a^{-1} \otimes a = i_\otimes$
3. $\otimes$ is both left and right distributive with respect to $\oplus$ and for every $s \in S$, $i_\oplus$ annihilates s, i.e. $a \otimes i_\oplus = i_\oplus \otimes a = i_\oplus$.

Semifields

Definition

A semifield $(S, \oplus, \otimes, i_\oplus, i_\otimes)$, is a set, S, endowed with two binary operations $\oplus, \otimes$ that satisfy the following conditions;

1. $\oplus$ is associative and commutative on all $a, b \in S$ and has identity $i_\oplus$ i.e. $\forall a \in S, i_\oplus \oplus a = a \oplus i_\oplus = a$
2. $\otimes$ is associative and has identity $i_\otimes$ i.e. $\forall a \in S, i_\otimes \otimes a = a \otimes i_\otimes = a$, and for all $a \in S \setminus \{i_\oplus\}$ there exists an inverse a^{-1} such that $a \otimes a^{-1} = a^{-1} \otimes a = i_\otimes$
3. $\otimes$ is both left and right distributive with respect to $\oplus$ and for every $s \in S$, $i_\oplus$ annihilates s, i.e. $a \otimes i_\oplus = i_\oplus \otimes a = i_\oplus$.

Outline

Defining the category

Definition

A Markov category **C** is a symmetric monoidal category in which every object $X \in \mathbf{C}$ is equipped with a commutative comonoid structure given by a comultiplication $\mathrm{copy}_X : X \to X \times X$ and counit $\mathrm{del}_X : X \to I$, where $\times$ is the tensor product and I is the unit object. The comultiplication and counit are usually depicted in string diagrams, further they satisfy commutative comonoid equations, compatibility with the monoidal structure and the naturality of del.

Definition

A *semifield* Markov category **C** is a Markov category in which morphism composition and the monoidal product is defined by some underlying semifield $(S, \otimes, \oplus, i_\otimes, i_\oplus)$. A special case of this category is the *affine semifield* Markov category in which there is an associated terminal morphism for each object $X \in \mathbf{C}$.

Defining the category

Definition

A Markov category $\mathbf{C}$ is a symmetric monoidal category in which every object $X \in \mathbf{C}$ is equipped with a commutative comonoid structure given by a comultiplication $\text{copy}_X : X \to X \times X$ and counit $\text{del}_X : X \to I$, where $\times$ is the tensor product and I is the unit object. The comultiplication and counit are usually depicted in string diagrams, further they satisfy commutative comonoid equations, compatibility with the monoidal structure and the naturality of del.

Definition

A *semifield* Markov category $\mathbf{C}$ is a Markov category in which morphism composition and the monoidal product is defined by some underlying semifield $(S, \otimes, \oplus, i_\otimes, i_\oplus)$. A special case of this category is the *affine semifield* Markov category in which there is an associated terminal morphism for each object $X \in \mathbf{C}$.

Defining the category

Definition

A Markov category $\mathbf{C}$ is a symmetric monoidal category in which every object $X \in \mathbf{C}$ is equipped with a commutative comonoid structure given by a comultiplication $\mathrm{copy}_X : X \to X \times X$ and counit $\mathrm{del}_X : X \to I$, where $\times$ is the tensor product and I is the unit object. The comultiplication and counit are usually depicted in string diagrams, further they satisfy commutative comonoid equations, compatibility with the monoidal structure and the naturality of del.

Definition

A *semifield* Markov category $\mathbf{C}$ is a Markov category in which morphism composition and the monoidal product is defined by some underlying semifield $(S, \otimes, \oplus, i_\otimes, i_\oplus)$. A special case of this category is the *affine semifield* Markov category in which there is an associated terminal morphism for each object $X \in \mathbf{C}$.

Defining the category

Extra structure

Objects, morphisms and morphism composition

Objects in semifield Markov categories are arbitrary index sets X, Y, and morphisms are of the form $\mathbf{f} : X \to D(Y)$, where $D(Y) = S^Y$ i.e. the set of all maps $m : Y \to S$, D is as a monad. Morphism composition in the category denoted $\cdot$, on two morphisms $\mathbf{f}_1 : X \to D(Y)$ and $\mathbf{f}_2 : Y \to D(Z)$, is given as

$$(\mathbf{f}_1 \cdot \mathbf{f}_2)(z|x) = \bigoplus_{y \in Y} \mathbf{f}_2(z|y) \otimes \mathbf{f}_1(y|x).$$

The affine requirement

For two morphisms $\mathbf{f} : Z \to D(Y)$ and $\mathbf{1}_Y : Y \to D(1)$, composition of the two yields

$$\mathbf{1}_Y \cdot \mathbf{f} = \mathbf{1}_Z$$

where $\mathbf{1}_Z : Z \to D(1)$.

Defining the category

Extra structure

Objects, morphisms and morphism composition

Objects in semifield Markov categories are arbitrary index sets X, Y, and morphisms are of the form $\mathbf{f} : X \to D(Y)$, where $D(Y) = S^Y$ i.e. the set of all maps $m : Y \to S$, D is as a monad. Morphism composition in the category denoted $\cdot$, on two morphisms $\mathbf{f}_1 : X \to D(Y)$ and $\mathbf{f}_2 : Y \to D(Z)$, is given as

$$(\mathbf{f}_1 \cdot \mathbf{f}_2)(z|x) = \bigoplus_{y \in Y} \mathbf{f}_2(z|y) \otimes \mathbf{f}_1(y|x).$$

The affine requirement

For two morphisms $\mathbf{f} : Z \to D(Y)$ and $\mathbf{1}_Y : Y \to D(1)$, composition of the two yields

$$\mathbf{1}_Y \cdot \mathbf{f} = \mathbf{1}_Z$$

where $\mathbf{1}_Z : Z \to D(1)$.

Defining the category

Extra structure

Objects, morphisms and morphism composition

Objects in semifield Markov categories are arbitrary index sets X, Y, and morphisms are of the form $\mathbf{f} : X \to D(Y)$, where $D(Y) = S^Y$ i.e. the set of all maps $m : Y \to S$, D is as a monad. Morphism composition in the category denoted $\cdot$, on two morphisms $\mathbf{f}_1 : X \to D(Y)$ and $\mathbf{f}_2 : Y \to D(Z)$, is given as

$$(\mathbf{f}_1 \cdot \mathbf{f}_2)(z|x) = \bigoplus_{y \in Y} \mathbf{f}_2(z|y) \otimes \mathbf{f}_1(y|x).$$

The affine requirement

For two morphisms $\mathbf{f} : Z \to D(Y)$ and $\mathbf{1}_Y : Y \to D(1)$, composition of the two yields

$$\mathbf{1}_Y \cdot \mathbf{f} = \mathbf{1}_Z$$

where $\mathbf{1}_Z : Z \to D(1)$.

Marginalization, normalization, extraction and disintegration

Marginalization and normalization

Assume $\mathbf{C}$ is an affine semifield Markov over an arbitrary semifield $\mathbf{S} = (S, \oplus, \otimes, i_{\oplus}, i_{\otimes})$;

- *Marginalization*: For a morphism $\mathbf{f} : Z \to D(Y)$ and a terminal morphism $\mathbf{1}_Y : Y \to D(1)$, we have $\mathbf{1}_Y \cdot \mathbf{f} = \mathbf{1}_Z$ so that

$$\bigoplus_{y \in Y} [\mathbf{1}_Y(|y) \otimes \mathbf{f}(y|z)] = \mathbf{1}_Z(|z).$$

- *Normalization*: For a morphism $\mathbf{f} : X \to D(Y)$ we define a normalized morphism with a bar i.e. $\bar{\mathbf{f}}(y|x)$ as follows,

$$\bar{\mathbf{f}}(y|x) = \mathbf{f}(y|x) \otimes \left(\bigoplus_{y \in Y} \mathbf{f}(y|x) \right)^{-1},$$

where $\mathbf{f}(y|)$ denotes a morphism with no input and $\mathbf{f}(|y)$ is a morphism with no output.

Marginalization, normalization, extraction and disintegration

Marginalization and normalization

Assume **C** is an affine semifield Markov over an arbitrary semifield $\mathbf{S} = (S, \oplus, \otimes, i_{\oplus}, i_{\otimes})$;

- *Marginalization*: For a morphism $\mathbf{f} : Z \to D(Y)$ and a terminal morphism $\mathbf{1}_Y : Y \to D(1)$, we have $\mathbf{1}_Y \cdot \mathbf{f} = \mathbf{1}_Z$ so that

$$\bigoplus_{y \in Y} [\mathbf{1}_Y(|y) \otimes \mathbf{f}(y|z)] = \mathbf{1}_Z(|z).$$

- *Normalization*: For a morphism $\mathbf{f} : X \to D(Y)$ we define a normalized morphism with a bar i.e. $\bar{\mathbf{f}}(y|x)$ as follows,

$$\bar{\mathbf{f}}(y|x) = \mathbf{f}(y|x) \otimes \left(\bigoplus_{y \in Y} \mathbf{f}(y|x) \right)^{-1},$$

where $\mathbf{f}(y|)$ denotes a morphism with no input and $\mathbf{f}(|y)$ is a morphism with no output.

Marginalization, normalization, extraction and disintegration

Marginalization and normalization

Assume **C** is an affine semifield Markov over an arbitrary semifield $\mathbf{S} = (S, \oplus, \otimes, i_\oplus, i_\otimes)$;

- *Marginalization*: For a morphism $\mathbf{f} : Z \to D(Y)$ and a terminal morphism $\mathbf{1}_Y : Y \to D(1)$, we have $\mathbf{1}_Y \cdot \mathbf{f} = 1_Z$ so that

$$\bigoplus_{y \in Y} [\mathbf{1}_Y(|y) \otimes \mathbf{f}(y|z)] = \mathbf{1}_Z(|z).$$

- *Normalization*: For a morphism $\mathbf{f} : X \to D(Y)$ we define a normalized morphism with a bar i.e. $\bar{\mathbf{f}}(y|x)$ as follows,

$$\bar{\mathbf{f}}(y|x) = \mathbf{f}(y|x) \otimes \left(\bigoplus_{y \in Y} \mathbf{f}(y|x) \right)^{-1},$$

where $\mathbf{f}(y|)$ denotes a morphism with no input and $\mathbf{f}(|y)$ is a morphism with no output.

Marginalization, normalization, extraction and disintegration

Marginalization and normalization

Assume **C** is an affine semifield Markov over an arbitrary semifield $\mathbf{S} = (S, \oplus, \otimes, i_{\oplus}, i_{\otimes})$;

- *Marginalization*: For a morphism $\mathbf{f} : Z \to D(Y)$ and a terminal morphism $\mathbf{1}_Y : Y \to D(1)$, we have $\mathbf{1}_Y \cdot \mathbf{f} = 1_Z$ so that

$$\bigoplus_{y \in Y} [\mathbf{1}_Y(|y) \otimes \mathbf{f}(y|z)] = \mathbf{1}_Z(|z).$$

- *Normalization*: For a morphism $\mathbf{f} : X \to D(Y)$ we define a normalized morphism with a bar i.e. $\bar{\mathbf{f}}(y|x)$ as follows,

$$\bar{\mathbf{f}}(y|x) = \mathbf{f}(y|x) \otimes \left(\bigoplus_{y \in Y} \mathbf{f}(y|x) \right)^{-1},$$

where $\mathbf{f}(y|)$ denotes a morphism with no input and $\mathbf{f}(|y)$ is a morphism with no output.

Marginalization, normalization, extraction and disintegration

Extraction and disintegration

Let $\times$ denote the monoidal product in which for two arbitrary morphisms $\mathbf{f} : X \to D(Y)$ and $\mathbf{g} : Y \to D(Z)$ we have $\mathbf{f} \times \mathbf{g} = (\mathbf{f} \times \mathbf{g})(x, y | u, v)$;

- *Extraction*: For a morphism $\mathbf{f} : X \to D(Y)$, extraction is defined as

$$\mathbf{f}(x, y | z) \mapsto \bar{\mathbf{f}}(y | x, z).$$

- *Disintegration*: For a morphism $\mathbf{f}_{12} : 1 \to D(X_1 \times X_2)$, disintegration is given as

$$\mathbf{f}_{12}(x_1, x_2 |) = \bar{\mathbf{f}}_{1|2}(x_1 | x_2) \otimes \mathbf{f}_2(x_2 |),$$

where we have morphisms $\bar{\mathbf{f}}_{1|2}(x_1 | x_2) : X_2 \to D(X_1)$ and $\mathbf{f}_2(x_2 |) : 1 \to D(X_2)$.

Marginalization, normalization, extraction and disintegration

Extraction and disintegration

Let $\times$ denote the monoidal product in which for two arbitrary morphisms $\mathbf{f} : X \to D(Y)$ and $\mathbf{g} : Y \to D(Z)$ we have $\mathbf{f} \times \mathbf{g} = (\mathbf{f} \times \mathbf{g})(x, y|u, v)$;

- *Extraction*: For a morphism $\mathbf{f} : X \to D(Y)$, extraction is defined as

$$\mathbf{f}(x, y|z) \mapsto \bar{\mathbf{f}}(y|x, z).$$

- *Disintegration*: For a morphism $\mathbf{f}_{12} : 1 \to D(X_1 \times X_2)$, disintegration is given as

$$\mathbf{f}_{12}(x_1, x_2|) = \bar{\mathbf{f}}_{1|2}(x_1|x_2) \otimes \mathbf{f}_2(x_2|),$$

where we have morphisms $\bar{\mathbf{f}}_{1|2}(x_1|x_2) : X_2 \to D(X_1)$ and $\mathbf{f}_2(x_2|) : 1 \to D(X_2)$.

Marginalization, normalization, extraction and disintegration

Extraction and disintegration

Let $\times$ denote the monoidal product in which for two arbitrary morphisms $\mathbf{f} : X \to D(Y)$ and $\mathbf{g} : Y \to D(Z)$ we have $\mathbf{f} \times \mathbf{g} = (\mathbf{f} \times \mathbf{g})(x, y|u, v)$;

- *Extraction*: For a morphism $\mathbf{f} : X \to D(Y)$, extraction is defined as

$$\mathbf{f}(x, y|z) \mapsto \bar{\mathbf{f}}(y|x, z).$$

- *Disintegration*: For a morphism $\mathbf{f}_{12} : 1 \to D(X_1 \times X_2)$, disintegration is given as

$$\mathbf{f}_{12}(x_1, x_2|) = \bar{\mathbf{f}}_{1|2}(x_1|x_2) \otimes \mathbf{f}_2(x_2|),$$

  where we have morphisms $\bar{\mathbf{f}}_{1|2}(x_1|x_2) : X_2 \to D(X_1)$ and $\mathbf{f}_2(x_2|) : 1 \to D(X_2)$.

Example of morphisms and compositions from machine learning

State normalization under the min-plus semifield

Example

Let $\mathbf{g}(y|\lambda) = -\ln(\lambda) + \lambda y$ for $y \in Y = \mathbb{R}^+$ and $\lambda > 0$, normalized $\mathbf{g}(y|\lambda) = \lambda y$. Assume a gamma prior, $\mathbf{f}(\lambda|) = -(\alpha - 1)(\ln(\lambda) + \ln\beta - \ln(\alpha - 1) + 1) + \beta\lambda$. Then

$$
\begin{aligned}
(\mathbf{g} \cdot \mathbf{f})(y|) &= \min_{\lambda \in \mathbb{R}^+} [\mathbf{g}(y|\lambda) + \mathbf{f}(\lambda|)] \\
&= \min_{\lambda \in \mathbb{R}^+} [\lambda(y - \beta) - (\alpha - 1)(\ln(\lambda) + \ln\beta - \ln(\alpha - 1) + 1)] \\
&= \ln\left(\frac{y + \beta}{\beta}\right)(\alpha - 1)
\end{aligned}
$$

which is normalized since $\ln\left(\frac{0+\beta}{\beta}\right)(\alpha - 1) = 0$. This is the negative log of the Lomax distribution, but we have obtained its log-density through minimization by differentiation, rather than integration.

Example of morphisms and compositions from machine learning

State normalization under the min-plus semifield

Example

Let $\mathbf{g}(y|\lambda) = -\ln(\lambda) + \lambda y$ for $y \in Y = \mathbb{R}^+$ and $\lambda > 0$, normalized $\mathbf{g}(y|\lambda) = \lambda y$. Assume a gamma prior, $\mathbf{f}(\lambda|) = -(\alpha - 1)(\ln(\lambda) + \ln\beta - \ln(\alpha - 1) + 1) + \beta\lambda$. Then

$$\begin{aligned}(\mathbf{g} \cdot \mathbf{f})(y|) &= \min_{\lambda \in \mathbb{R}^+} [\mathbf{g}(y|\lambda) + \mathbf{f}(\lambda|)] \\ &= \min_{\lambda \in \mathbb{R}^+} [\lambda(y - \beta) - (\alpha - 1)(\ln(\lambda) + \ln\beta - \ln(\alpha - 1) + 1)] \\ &= \ln\left(\frac{y + \beta}{\beta}\right)(\alpha - 1)\end{aligned}$$

which is normalized since $\ln\left(\frac{0+\beta}{\beta}\right)(\alpha - 1) = 0$. This is the negative log of the Lomax distribution, but we have obtained its log-density through minimization by differentiation, rather than integration.

Outline

Defining the semifield homomorphism

Suppose, we have two arbitrary semifields $\mathbf{S} = (S, \oplus, \otimes, i_{\oplus}, i_{\otimes})$ and $\mathbf{S}' = (S', \oplus', \otimes', i_{\oplus'}, i_{\otimes'})$.

Definition

A function $h : S \to S'$ is a semifield homomorphism if for every $s_1, s_2 \in S$, the following equations hold

$$h(s_1 \oplus s_2) = h(s_1) \oplus' h(s_2)$$
$$h(s_1 \otimes s_2) = h(s_1) \otimes' h(s_2)$$
$$h(i_{\oplus}) = i_{\oplus'}$$
$$h(i_{\otimes}) = i_{\otimes'}$$

such that $\otimes'$ distributes over $\oplus'$.

Defining the semifield homomorphism

Suppose, we have two arbitrary semifields $\mathbf{S} = (S, \oplus, \otimes, i_\oplus, i_\otimes)$ and $\mathbf{S}' = (S', \oplus', \otimes', i_{\oplus'}, i_{\otimes'})$.

Definition

A function $h : S \to S'$ is a semifield homomorphism if for every $s_1, s_2 \in S$, the following equations hold

$$h(s_1 \oplus s_2) = h(s_1) \oplus' h(s_2)$$
$$h(s_1 \otimes s_2) = h(s_1) \otimes' h(s_2)$$
$$h(i_\oplus) = i_{\oplus'}$$
$$h(i_\otimes) = i_{\otimes'}$$

such that $\otimes'$ distributes over $\oplus'$.

Defining the semifield homomorphism

Suppose, we have two arbitrary semifields $\mathbf{S} = (S, \oplus, \otimes, i_\oplus, i_\otimes)$ and $\mathbf{S}' = (S', \oplus', \otimes', i_{\oplus'}, i_{\otimes'})$.

Definition

A function $h : S \to S'$ is a semifield homomorphism if for every $s_1, s_2 \in S$, the following equations hold

$$\begin{aligned} h(s_1 \oplus s_2) &= h(s_1) \oplus' h(s_2) \\ h(s_1 \otimes s_2) &= h(s_1) \otimes' h(s_2) \\ h(i_\oplus) &= i_{\oplus'} \\ h(i_\otimes) &= i_{\otimes'} \end{aligned}$$

such that $\otimes'$ distributes over $\oplus'$.

Outline

Defining the semifield transport functor

Assume that $\mathbf{C}, \mathbf{C}'$ are affine semifield Markov categories over semifields $\mathbf{S} = (S, +, \times, 0, 1), \mathbf{S}' = (S', \oplus, \otimes, i_{\oplus}, i_{\otimes})$ respectively.

Theorem

Let $\mathbf{f} : X \to D(Y) \in \mathbf{C}$ and $\mathbf{f}' : X \to D(Y) \in \mathbf{C}'$. If there exists a semifield homomorphism $h : S \to S'$, then we can define a semifield transport functor $F : \mathbf{C} \to \mathbf{C}'$ is defined as the following;

1. *On objects $X \in \mathbf{C}$, $F(X) = X \in \mathbf{C}'$*
2. *On morphisms $\mathbf{f}, \mathbf{f}'$, $F(\mathbf{f})(y|x) = h(\mathbf{f}(y|x)) = \mathbf{f}'(y|x)$.*

Corollary

If there exists an isomorphism of semifields i.e. an inverse to the semifield homomorphism, then it is possible to construct an isomorphism of categories.

Defining the semifield transport functor

Assume that $\mathbf{C}, \mathbf{C}'$ are affine semifield Markov categories over semifields $\mathbf{S} = (S, +, \times, 0, 1)$, $\mathbf{S}' = (S', \oplus, \otimes, i_\oplus, i_\otimes)$ respectively.

Theorem

Let $\mathbf{f} : X \to D(Y) \in \mathbf{C}$ and $\mathbf{f}' : X \to D(Y) \in \mathbf{C}'$. If there exists a semifield homomorphism $h : S \to S'$, then we can define a semifield transport functor $F : \mathbf{C} \to \mathbf{C}'$ is defined as the following;

1. *On objects $X \in \mathbf{C}$, $F(X) = X \in \mathbf{C}'$*
2. *On morphisms $\mathbf{f}, \mathbf{f}'$,$F(\mathbf{f})(y|x) = h(\mathbf{f}(y|x)) = \mathbf{f}'(y|x)$.*

Corollary

If there exists an isomorphism of semifields i.e. an inverse to the semifield homomorphism, then it is possible to construct an isomorphism of categories.

Defining the semifield transport functor

Assume that $\mathbf{C}, \mathbf{C}'$ are affine semifield Markov categories over semifields $\mathbf{S} = (S, +, \times, 0, 1)$, $\mathbf{S}' = (S', \oplus, \otimes, i_\oplus, i_\otimes)$ respectively.

Theorem

Let $\mathbf{f} : X \to D(Y) \in \mathbf{C}$ and $\mathbf{f}' : X \to D(Y) \in \mathbf{C}'$. If there exists a semifield homomorphism $h : S \to S'$, then we can define a semifield transport functor $F : \mathbf{C} \to \mathbf{C}'$ is defined as the following;

1. *On objects $X \in \mathbf{C}$, $F(X) = X \in \mathbf{C}'$*
2. *On morphisms $\mathbf{f}, \mathbf{f}'$,$F(\mathbf{f})(y|x) = h(\mathbf{f}(y|x)) = \mathbf{f}'(y|x)$.*

Corollary

If there exists an isomorphism of semifields i.e. an inverse to the semifield homomorphism, then it is possible to construct an isomorphism of categories.

Defining the semifield transport functor

Assume that $\mathbf{C}, \mathbf{C}'$ are affine semifield Markov categories over semifields $\mathbf{S} = (S, +, \times, 0, 1)$, $\mathbf{S}' = (S', \oplus, \otimes, i_\oplus, i_\otimes)$ respectively.

Theorem

Let $\mathbf{f} : X \to D(Y) \in \mathbf{C}$ and $\mathbf{f}' : X \to D(Y) \in \mathbf{C}'$. If there exists a semifield homomorphism $h : S \to S'$, then we can define a semifield transport functor $F : \mathbf{C} \to \mathbf{C}'$ is defined as the following;

1. *On objects $X \in \mathbf{C}$, $F(X) = X \in \mathbf{C}'$*
2. *On morphisms $\mathbf{f}, \mathbf{f}'$,$F(\mathbf{f})(y|x) = h(\mathbf{f}(y|x)) = \mathbf{f}'(y|x)$.*

Corollary

If there exists an isomorphism of semifields i.e. an inverse to the semifield homomorphism, then it is possible to construct an isomorphism of categories.

Outline

An example transport functor

Information transform

Example

Let $h(s) = -\frac{1}{a}\ln(s)$ for $a > 0$, mapping $[0,1] \to \mathbb{R}^{+} \cup \infty$. In this case, $h^{-1}(s') = \exp(-as')$ and we have

$$s'_1 \otimes s'_2 = -\frac{1}{a}\ln\left(\exp\left(-as'_1\right) \times \exp\left(-as'_1\right)\right) = \frac{a}{a}\left(s'_1 + s'_2\right) = s'_1 + s'_2$$

$$s'_1 \oplus s'_2 = \to -\frac{1}{a}\ln\left(\exp\left(-as'_1\right) + \exp\left(-as'_2\right)\right) \to \min\left(s'_1, s'_2\right)$$

in the limit as $a \to \infty$. This obtains the semifield $S' = ([0,\infty], \min, +, \infty, 0)$, the *min-plus* semifield from the usual probability semifield, $([0,1], +, \times, 0, 1)$.

An example transport functor

Information transform

Example

Let $h(s) = -\frac{1}{a}\ln(s)$ for $a > 0$, mapping $[0,1] \to \mathbb{R}^+ \cup \infty$. In this case, $h^{-1}(s') = \exp(-as')$ and we have

$$s_1' \otimes s_2' \quad = \quad -\frac{1}{a}\ln\left(\exp\left(-as_1'\right) \times \exp\left(-as_1'\right)\right) = \frac{a}{a}\left(s_1' + s_2'\right) = s_1' + s_2'$$

$$s_1' \oplus s_2' \quad = \quad \to -\frac{1}{a}\ln\left(\exp\left(-as_1'\right) + \exp\left(-as_2'\right)\right) \to \min\left(s_1', s_2'\right)$$

in the limit as $a \to \infty$. This obtains the semifield $S' = ([0,\infty], \min, +, \infty, 0)$, the *min-plus* semifield from the usual probability semifield, $([0,1], +, \times, 0, 1)$.

Outline

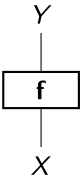
String diagram convention

- We will represent morphisms of the type $\mathbf{f} : X \to D(Y)$ in the form

- We will represent morphisms of the form
$\mathbf{f} : X_1 \times X_2 \times \cdots \times X_K \to D(Y)$, where $K \in \mathbb{N}$, as

String diagram convention

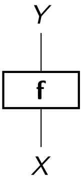
- We will represent morphisms of the type $\mathbf{f} : X \to D(Y)$ in the form



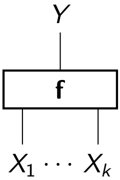
- We will represent morphisms of the form $\mathbf{f} : X_1 \times X_2 \times \cdots \times X_K \to D(Y)$, where $K \in \mathbb{N}$, as

String diagram convention

- We will represent morphisms of the type $\mathbf{f} : X \to D(Y)$ in the form



- We will represent morphisms of the form $\mathbf{f} : X_1 \times X_2 \times \cdots \times X_K \to D(Y)$, where $K \in \mathbb{N}$, as



Representing DAGs as string diagrams

Suppose that we have a DAG that looks like

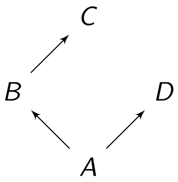
$P(ABCD) = P(A)P(B|D)P(D|A)P(C|B)$

Then we can represent it in string diagram notation as

- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

Representing DAGs as string diagrams

Suppose that we have a DAG that looks like



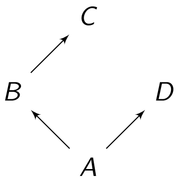
$P(ABCD) =$
$P(A)\,P(B|D)\,P(D|A)\,P(C|B)$

Then we can represent it in string diagram notation as

- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

Representing DAGs as string diagrams

Suppose that we have a DAG that looks like



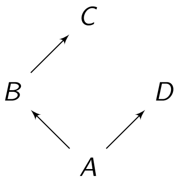
$$P(ABCD) = P(A)\,P(B|D)\,P(D|A)\,P(C|B)$$

Then we can represent it in string diagram notation as

- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

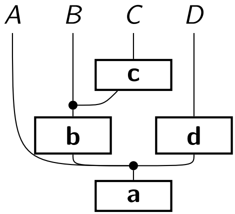
Representing DAGs as string diagrams

Suppose that we have a DAG that looks like



$P(ABCD) =$
$P(A)\,P(B|D)\,P(D|A)\,P(C|B)$

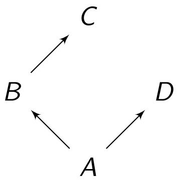
Then we can represent it in string diagram notation as



- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

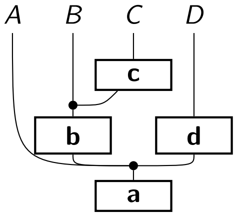
Representing DAGs as string diagrams

Suppose that we have a DAG that looks like



$$P(ABCD) = P(A)\,P(B|D)\,P(D|A)\,P(C|B)$$

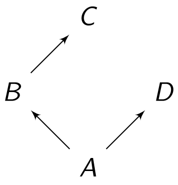
Then we can represent it in string diagram notation as



- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

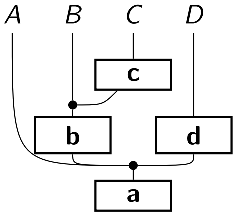
Representing DAGs as string diagrams

Suppose that we have a DAG that looks like



$P(ABCD) =$
$P(A)\,P(B|D)\,P(D|A)\,P(C|B)$

Then we can represent it in string diagram notation as



- The morphism $\mathbf{a} : 1 \to D(A)$ contains the probability $P(A)$, $\mathbf{b} : A \to D(B)$ contains the probability $P(B|A)$,etc. The entire diagram is equal to a morphism $\mathbf{f} : 1 \to D(A \times B \times C \times D)$
- When a variable is unobserved, we represent this by removing the output wire.

Point-state cut functor

Definition

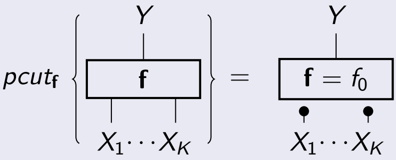
For any morphisms $\mathbf{f} : X_1 \times X_2 \times \cdots \times X_K \to D(Y)$, where $K \in \mathbb{N}$, the point-state cut functor $pcut_{\mathbf{f}}(\mathbf{f})$, acts in the following way

where $\mathbf{f} = f_0$ denotes a point-state morphism where f_0 is a fixed value, on morphisms $\mathbf{g} \neq \mathbf{f}$, $pcut_{\mathbf{f}}(\mathbf{g}) = \mathbf{g}$, and on objects X, $pcut_{\mathbf{f}}(X) = X$.

Point-state cut functor

Definition

For any morphisms $\mathbf{f} : X_1 \times X_2 \times \cdots \times X_K \to D(Y)$, where $K \in \mathbb{N}$, the point-state cut functor $pcut_{\mathbf{f}}(\mathbf{f})$, acts in the following way



where $\mathbf{f} = f_0$ denotes a point-state morphism where f_0 is a fixed value, on morphisms $\mathbf{g} \neq \mathbf{f}$, $pcut_{\mathbf{f}}(\mathbf{g}) = \mathbf{g}$, and on objects X, $pcut_{\mathbf{f}}(X) = X$.

Marginalizing on string diagrams

If we have a morphism given by $\mathbf{f}(a, x|w)$ represented by the following diagram

To marginalize out x, we have the following manipulations.

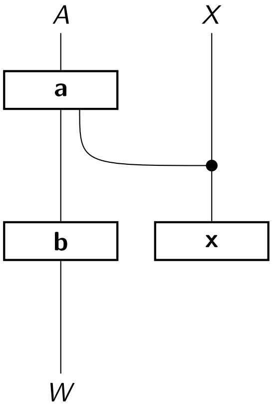
Marginalizing on string diagrams

If we have a morphism given by $\mathbf{f}(a, x|w)$ represented by the following diagram

To marginalize out x, we have the following manipulations.

Marginalizing on string diagrams

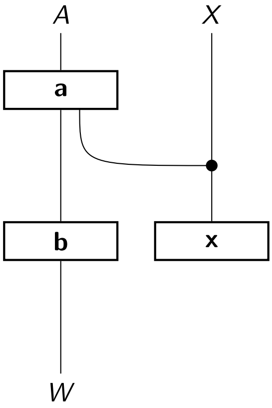
If we have a morphism given by $\mathbf{f}(a, x|w)$ represented by the following diagram



To marginalize out x, we have the following manipulations.

Marginalizing on string diagrams

If we have a morphism given by $\mathbf{f}(a, x|w)$ represented by the following diagram



To marginalize out x, we have the following manipulations.

Marginalizing on string diagrams

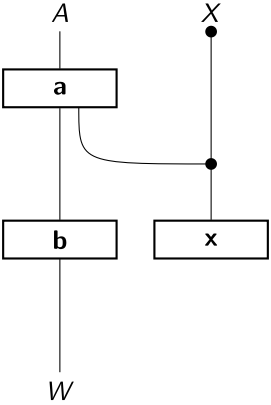
We use the affine property of the category to discard the output of x

Marginalizing on string diagrams

We use the affine property of the category to discard the output of x

Marginalizing on string diagrams

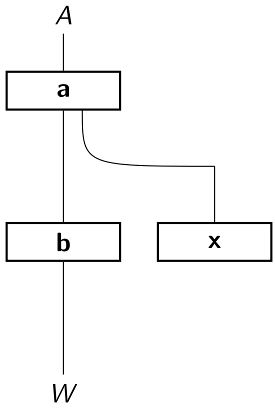
We use the affine property of the category to discard the output of x



Marginalizing on string diagrams

After marginalizing out x we have obtained the morphism $\mathbf{f}(a|w)$.

Marginalizing on string diagrams



After marginalizing out x we have obtained the morphism $\mathbf{f}\,(a|w)$.

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}\,(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

Given a morphism $\mathbf{f}(a, x|w)$ then fixing operates with the following procedure;

- Marginalize out all the inputs to the morphism we want to fix on
- Replace the morphism with an identity wire
- Extend this new identity wire to become a new input to the entire diagram
- Marginalize out the variable we want to fix at the output of the entire diagram
- After fixing, include a point state for the interventional variable added at the input of the complete string

The fixing morphism

In the string diagram formalism, this is represented in the following example

Example

Suppose we have a morphism, $\mathbf{f}(a, x|w)$, the fixing morphism $\mathcal{F}_x(\mathbf{f})$, acts in the following way

Fixing in this example has the following effect, $\mathcal{F}_x(\mathbf{f}(a, x|w)) = \mathbf{f}(a|x, w)$. In some cases, x will just be marginalized out, and on a morphism $\mathbf{f}(a, y|w)$, fixing on x has the following effect $\mathcal{F}_x(\mathbf{f}(a, y|w)) = \mathbf{f}(a, y|w)$.

The fixing morphism

In the string diagram formalism, this is represented in the following example

Example

Suppose we have a morphism, $\mathbf{f}(a, x|w)$, the fixing morphism $\mathcal{F}_x(\mathbf{f})$, acts in the following way

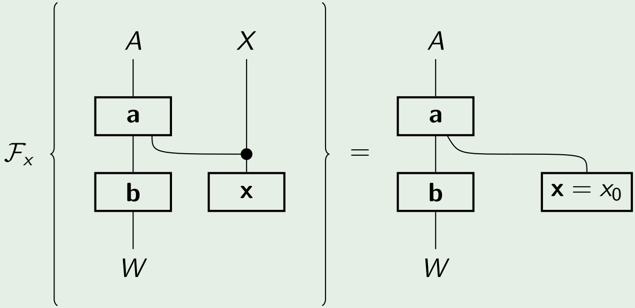
Fixing in this example has the following effect, $\mathcal{F}_x(\mathbf{f}(a, x|w)) = \mathbf{f}(a|x, w)$. In some cases, x will just be marginalized out, and on a morphism $\mathbf{f}(a, y|w)$, fixing on x has the following effect $\mathcal{F}_x(\mathbf{f}(a, y|w)) = \mathbf{f}(a, y|w)$.

The fixing morphism

In the string diagram formalism, this is represented in the following example

Example

Suppose we have a morphism, $\mathbf{f}(a, x|w)$, the fixing morphism $\mathcal{F}_x(\mathbf{f})$, acts in the following way



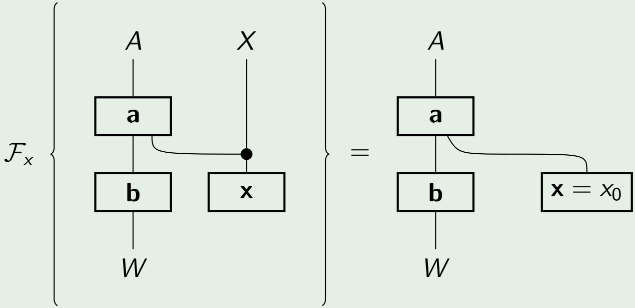
Fixing in this example has the following effect, $\mathcal{F}_x(\mathbf{f}(a, x|w)) = \mathbf{f}(a|x, w)$. In some cases, x will just be marginalized out, and on a morphism $\mathbf{f}(a, y|w)$, fixing on x has the following effect $\mathcal{F}_x(\mathbf{f}(a, y|w)) = \mathbf{f}(a, y|w)$.

The fixing morphism

In the string diagram formalism, this is represented in the following example

Example

Suppose we have a morphism, $\mathbf{f}\,(a, x|w)$, the fixing morphism $\mathcal{F}_x\,(\mathbf{f})$, acts in the following way



Fixing in this example has the following effect, $\mathcal{F}_x\,(\mathbf{f}\,(a, x|w)) = \mathbf{f}\,(a|x, w)$. In some cases, x will just be marginalized out, and on a morphism $\mathbf{f}\,(a, y|w)$, fixing on x has the following effect $\mathcal{F}_x\,(\mathbf{f}\,(a, y|w)) = \mathbf{f}\,(a, y|w)$.

Outline

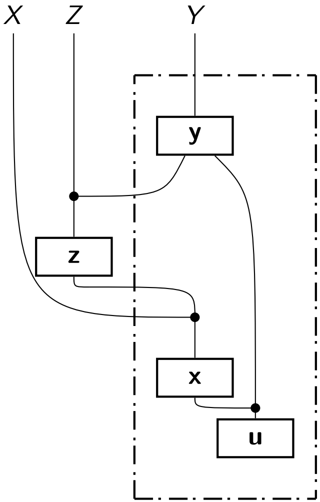
Districts and kernels

String diagram perspective

- Districts are kernels (string diagrams) in which a set of unobserved common cause morphisms provides an input to other variables such as X, Y and this occurs transitively across all unobserved common cause morphisms.
- Here the district is $\mathbf{f}(x, y|z)$with internal kernel morphisms x, y, u.

Districts and kernels

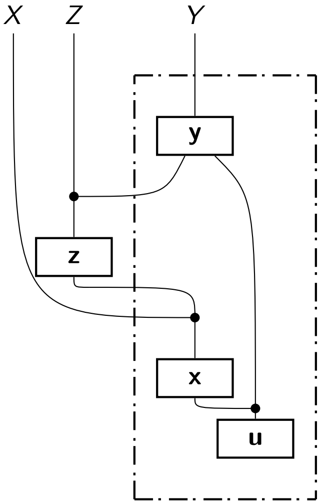
String diagram perspective



- Districts are kernels (string diagrams) in which a set of unobserved common cause morphisms provides an input to other variables such as X, Y and this occurs transitively across all unobserved common cause morphisms.
- Here the district is $\mathbf{f}(x, y|z)$with internal kernel morphisms x, y, u.

Districts and kernels

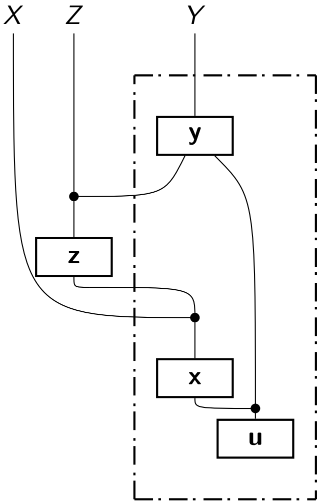
String diagram perspective



- Districts are kernels (string diagrams) in which a set of unobserved common cause morphisms provides an input to other variables such as X, Y and this occurs transitively across all unobserved common cause morphisms.
- Here the district is $\mathbf{f}(x, y|z)$with internal kernel morphisms x, y, u.

Districts and kernels

String diagram perspective



- Districts are kernels (string diagrams) in which a set of unobserved common cause morphisms provides an input to other variables such as X, Y and this occurs transitively across all unobserved common cause morphisms.
- Here the district is $\mathbf{f}(x, y|z)$with internal kernel morphisms x, y, u.

Outline

Back-door adjustment through fixing

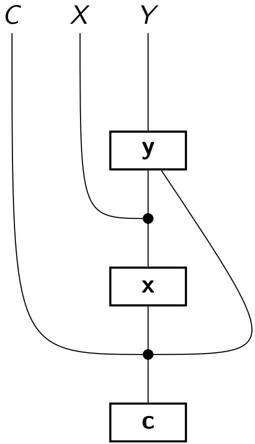
String diagram approach

Here we have the typical Back-door DAG in the string diagram formalism, in this example the morphism is given as $\mathbf{f}(c, x, y|)$,

Back-door adjustment through fixing

String diagram approach

Here we have the typical Back-door DAG in the string diagram formalism, in this example the morphism is given as $\mathbf{f}(c, x, y|)$,



Back-door adjustment through fixing

String diagram approach

When we apply $\mathcal{F}_x$ to the diagram it simulates intervention on x obtaining $\mathcal{F}_x(\mathbf{f}(c,x,y|)) = \mathbf{f}(c,y|x)$,

Back-door adjustment through fixing

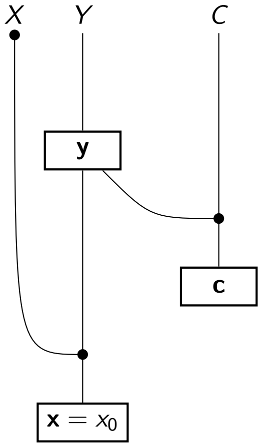
String diagram approach

When we apply $\mathcal{F}_x$ to the diagram it simulates intervention on x obtaining $\mathcal{F}_x\left(\mathbf{f}\left(c,x,y|\right)\right)=\mathbf{f}\left(c,y|x\right),$

Back-door adjustment through fixing

String diagram approach

When we apply $\mathcal{F}_x$ to the diagram it simulates intervention on x obtaining $\mathcal{F}_x\left(\mathbf{f}\left(c, x, y|\right)\right) = \mathbf{f}\left(c, y|x\right)$,



Back-door adjustment through fixing

String diagram approach

After diagram simplification we obtain,

This is precisely $\mathbf{f}(c, y|x) = \mathbf{f}(c|)\,\mathbf{f}(y|x, c)$.

Back-door adjustment through fixing

String diagram approach

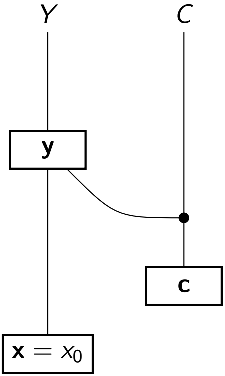
After diagram simplification we obtain,

This is precisely $\mathbf{f}(c, y|x) = \mathbf{f}(c|)\, \mathbf{f}(y|x, c)$.

Back-door adjustment through fixing

String diagram approach

After diagram simplification we obtain,

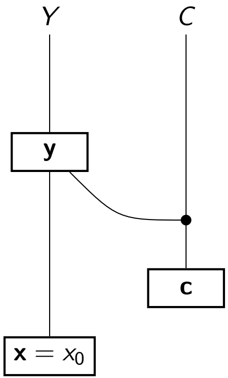


This is precisely $\mathbf{f}(c, y|x) = \mathbf{f}(c|)\,\mathbf{f}(y|x, c)$.

Back-door adjustment through fixing

String diagram approach

After diagram simplification we obtain,



This is precisely $\mathbf{f}\,(c, y|x) = \mathbf{f}\,(c|)\,\mathbf{f}\,(y|x, c)$.

Back-door adjustment through fixing

String diagram approach

We then marginalize out c,

Back-door adjustment through fixing

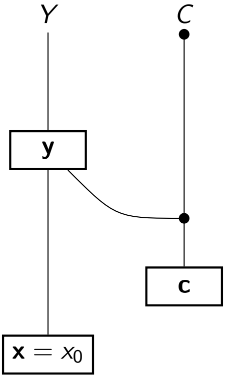
String diagram approach

We then marginalize out c,

Back-door adjustment through fixing

String diagram approach

We then marginalize out c,



Back-door adjustment through fixing

String diagram approach

Once we marginalize out c we obtain a diagram with the composite morphism $\mathbf{f}(c, y|x) = \sum_c \mathbf{f}(c|)\,\mathbf{f}(y|x, c)$ i.e. the back-door adjustment formula,

Back-door adjustment through fixing

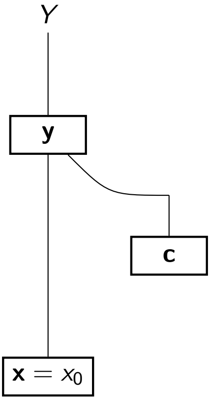
String diagram approach

Once we marginalize out c we obtain a diagram with the composite morphism $\mathbf{f}(c, y|x) = \sum_c \mathbf{f}(c|)\,\mathbf{f}(y|x, c)$ i.e. the back-door adjustment formula,

Back-door adjustment through fixing

String diagram approach

Once we marginalize out c we obtain a diagram with the composite morphism $\mathbf{f}(c, y|x) = \sum_c \mathbf{f}(c|)\,\mathbf{f}(y|x, c)$ i.e. the back-door adjustment formula,



Outline

Front-door adjustment through fixing

- In the front-door DAG, U is a hidden variable which influence both X and Y
- By latent projection U is replaced by a bidirected edge between X and Y
- In the string diagram representation we have the latent morphism $\mathbf{f}(u|)$ as an input to morphisms $\mathbf{f}(x|u)$ and $\mathbf{f}(y|z,u)$, but u itself is not exposed to the output.

Front-door adjustment through fixing

- In the front-door DAG, U is a hidden variable which influence both X and Y
- By latent projection U is replaced by a bidirected edge between X and Y
- In the string diagram representation we have the latent morphism $\mathbf{f}(u|)$ as an input to morphisms $\mathbf{f}(x|u)$ and $\mathbf{f}(y|z,u)$, but u itself is not exposed to the output.

Front-door adjustment through fixing

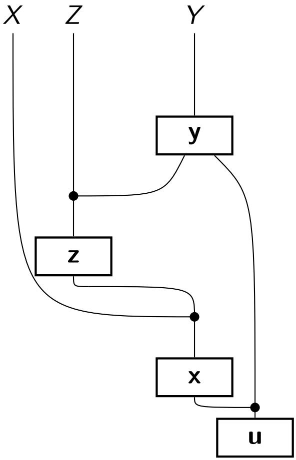
- In the front-door DAG, U is a hidden variable which influence both X and Y
- By latent projection U is replaced by a bidirected edge between X and Y
- In the string diagram representation we have the latent morphism $\mathbf{f}(u|)$ as an input to morphisms $\mathbf{f}(x|u)$ and $\mathbf{f}(y|z,u)$, but u itself is not exposed to the output.

Front-door adjustment through fixing

- In the front-door DAG, U is a hidden variable which influence both X and Y
- By latent projection U is replaced by a bidirected edge between X and Y
- In the string diagram representation we have the latent morphism $\mathbf{f}\left(u|\right)$ as an input to morphisms $\mathbf{f}\left(x|u\right)$ and $\mathbf{f}\left(y|z,u\right)$, but u itself is not exposed to the output.

Front-door adjustment through fixing

- In the front-door DAG, U is a hidden variable which influence both X and Y
- By latent projection U is replaced by a bidirected edge between X and Y
- In the string diagram representation we have the latent morphism $\mathbf{f}(u|)$ as an input to morphisms $\mathbf{f}(x|u)$ and $\mathbf{f}(y|z,u)$, but u itself is not exposed to the output.



Front-door adjustment through fixing

We start with the the chain factored front-door diagram

We can identify z as a district and have a valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_y\left(\mathbf{f}\left(z|x\right)\right)\right)$.

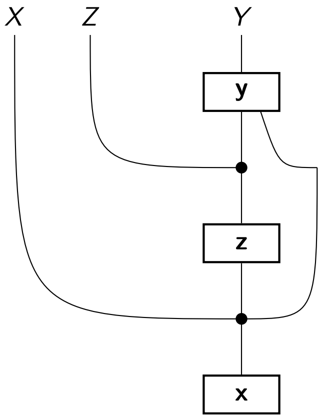
Front-door adjustment through fixing

We start with the the chain factored front-door diagram

We can identify z as a district and have a valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_y\left(\mathbf{f}\left(z|x\right)\right)\right)$.

Front-door adjustment through fixing

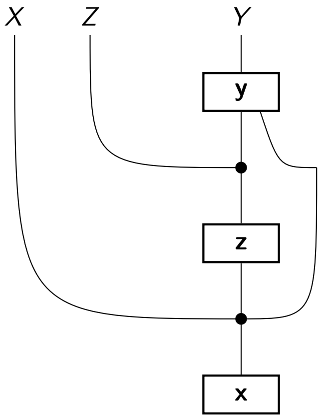
We start with the the chain factored front-door diagram



We can identify z as a district and have a valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_y\left(\mathbf{f}\left(z|x\right)\right)\right)$.

Front-door adjustment through fixing

We start with the the chain factored front-door diagram



We can identify z as a district and have a valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_y\left(\mathbf{f}\left(z|x\right)\right)\right)$.

Front-door adjustment through fixing

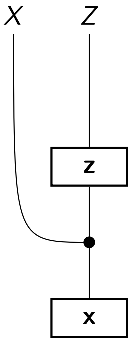
Once we fix on y and simplifying the string diagram we obtain,

Front-door adjustment through fixing

Once we fix on y and simplifying the string diagram we obtain,

Front-door adjustment through fixing

Once we fix on y and simplifying the string diagram we obtain,



Front-door adjustment through fixing

After fixing on x and using diagram simplification eliminates X at the output, effectively replacing it with the constant morphism at value X, which obtains the string diagram,

This is the final kernel diagram for this district.

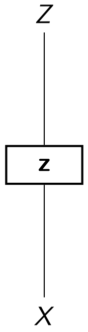
Front-door adjustment through fixing

After fixing on x and using diagram simplification eliminates X at the output, effectively replacing it with the constant morphism at value X, which obtains the string diagram,

This is the final kernel diagram for this district.

Front-door adjustment through fixing

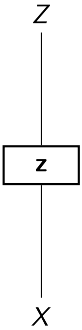
After fixing on x and using diagram simplification eliminates X at the output, effectively replacing it with the constant morphism at value X, which obtains the string diagram,



This is the final kernel diagram for this district.

Front-door adjustment through fixing

After fixing on x and using diagram simplification eliminates X at the output, effectively replacing it with the constant morphism at value X, which obtains the string diagram,



This is the final kernel diagram for this district.

Front-door adjustment through fixing

Once again we start with the chain factored front-door diagram,

Here we can identify a district y with valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_z\left(\mathbf{f}\left(y|z,x\right)\right)\right)$.

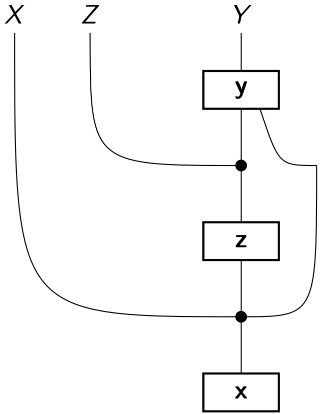
Front-door adjustment through fixing

Once again we start with the chain factored front-door diagram,

Here we can identify a district y with valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_z\left(\mathbf{f}\left(y|z,x\right)\right)\right)$.

Front-door adjustment through fixing

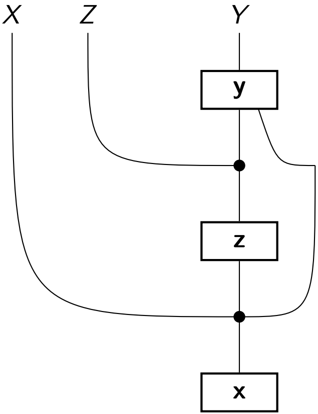
Once again we start with the chain factored front-door diagram,



Here we can identify a district y with valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_z\left(\mathbf{f}\left(y|z,x\right)\right)\right)$.

Front-door adjustment through fixing

Once again we start with the chain factored front-door diagram,



Here we can identify a district y with valid fixing sequence $\mathcal{F}_x\left(\mathcal{F}_z\left(\mathbf{f}\left(y|z,x\right)\right)\right)$.

Front-door adjustment through fixing

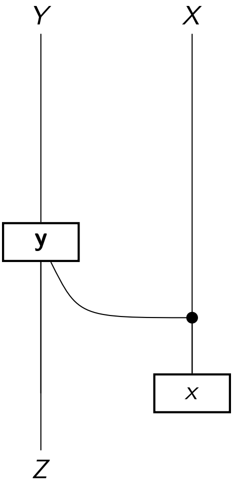
After fixing on z and using diagram simplification we have,

Front-door adjustment through fixing

After fixing on z and using diagram simplification we have,

Front-door adjustment through fixing

After fixing on z and using diagram simplification we have,



Front-door adjustment through fixing

After fixing on x, and since X has no children in the string diagram at this point in the fixing sequence X is discarded from the output but remains as a hidden morphism, so we obtain,

This is the final kernel diagram for this district.

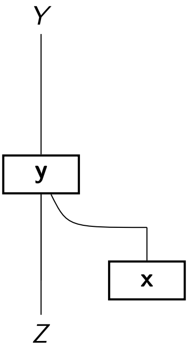
Front-door adjustment through fixing

After fixing on x, and since X has no children in the string diagram at this point in the fixing sequence X is discarded from the output but remains as a hidden morphism, so we obtain,

This is the final kernel diagram for this district.

Front-door adjustment through fixing

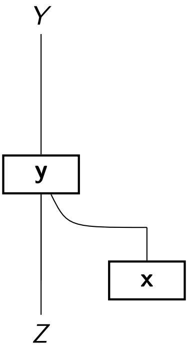
After fixing on x, and since X has no children in the string diagram at this point in the fixing sequence X is discarded from the output but remains as a hidden morphism, so we obtain,



This is the final kernel diagram for this district.

Front-door adjustment through fixing

After fixing on x, and since X has no children in the string diagram at this point in the fixing sequence X is discarded from the output but remains as a hidden morphism, so we obtain,



This is the final kernel diagram for this district.

Front-door adjustment through fixing

Re-combining district kernel diagrams

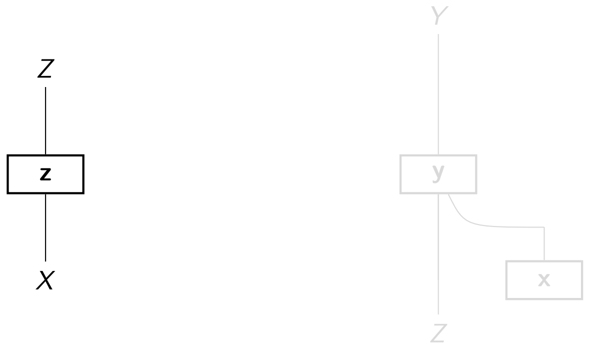
Recall the final kernel diagrams,

We can compose these two sequentially adhering to topological consistency.

Front-door adjustment through fixing

Re-combining district kernel diagrams

Recall the final kernel diagrams,

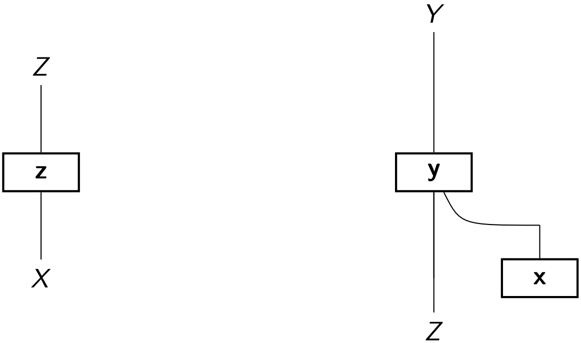


We can compose these two sequentially adhering to topological consistency.

Front-door adjustment through fixing

Re-combining district kernel diagrams

Recall the final kernel diagrams,

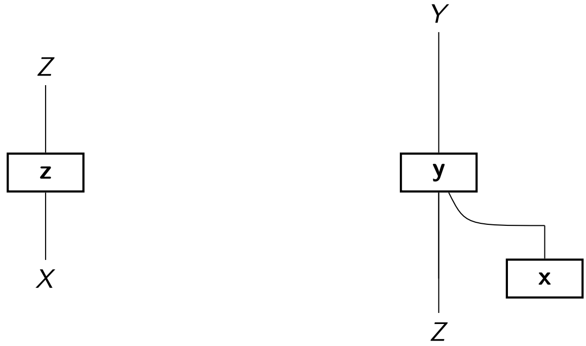


We can compose these two sequentially adhering to topological consistency.

Front-door adjustment through fixing

Re-combining district kernel diagrams

Recall the final kernel diagrams,



We can compose these two sequentially adhering to topological consistency.

Front-door adjustment through fixing

Re-combining district kernel diagrams

This results in the combined diagram representing the intervention on the desired variable, x. The resulting expression for the interventional diagram is the front-door adjustment formula.

Front-door adjustment through fixing

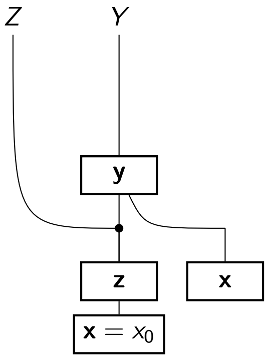
Re-combining district kernel diagrams

This results in the combined diagram representing the intervention on the desired variable, x. The resulting expression for the interventional diagram is the front-door adjustment formula.

Front-door adjustment through fixing

Re-combining district kernel diagrams

This results in the combined diagram representing the intervention on the desired variable, x. The resulting expression for the interventional diagram is the front-door adjustment formula.



Summary

- In this work we have introduced a method in which to conduct causal inference in a non-probabilistic setting
- We have shown how it is possible to create formalism in which to conduct causal inference and machine learning together
- We have shown how we can apply this work to classical do-calculus problems such as the back-door and front-door adjustment
- This novel approach gives us a new perspective on DAG models, specifically when it comes to topological ordering.

Summary

- In this work we have introduced a method in which to conduct causal inference in a non-probabilistic setting
- We have shown how it is possible to create formalism in which to conduct causal inference and machine learning together
- We have shown how we can apply this work to classical do-calculus problems such as the back-door and front-door adjustment
- This novel approach gives us a new perspective on DAG models, specifically when it comes to topological ordering.

Summary

- In this work we have introduced a method in which to conduct causal inference in a non-probabilistic setting
- We have shown how it is possible to create formalism in which to conduct causal inference and machine learning together
- We have shown how we can apply this work to classical do-calculus problems such as the back-door and front-door adjustment
- This novel approach gives us a new perspective on DAG models, specifically when it comes to topological ordering.

Summary

- In this work we have introduced a method in which to conduct causal inference in a non-probabilistic setting
- We have shown how it is possible to create formalism in which to conduct causal inference and machine learning together
- We have shown how we can apply this work to classical do-calculus problems such as the back-door and front-door adjustment
- This novel approach gives us a new perspective on DAG models, specifically when it comes to topological ordering.

Summary

- In this work we have introduced a method in which to conduct causal inference in a non-probabilistic setting
- We have shown how it is possible to create formalism in which to conduct causal inference and machine learning together
- We have shown how we can apply this work to classical do-calculus problems such as the back-door and front-door adjustment
- This novel approach gives us a new perspective on DAG models, specifically when it comes to topological ordering.