The EfProb Library for Probabilistic Calculations

https://efprob.cs.ru.nl/

Kenta Cho   Bart Jacobs

Radboud University, Nijmegen

iCIS | Digital Security
Radboud University

CALCO Tools
15 June 2017

EfProb

EfProb = **Ef**fectus **Prob**ability

EfProb = **Ef**fectus **Prob**ability

- Effectus theory provides a general setting for discrete/continuous/quantum probability, via
  - $\mathcal{K}\ell(\mathcal{D})$, the Kleisli category for the distribution monad
  - $\mathcal{K}\ell(\mathcal{G})$, the Kleisli category for the Giry monad
  - $\mathbf{vNA}^{\mathrm{op}}$, (opposite) category of von Neumann algebras

EfProb = **Ef**fectus **Prob**ability

- Effectus theory provides a general setting for discrete/continuous/quantum probability, via
  - $\mathcal{K}\ell(\mathcal{D})$, the Kleisli category for the distribution monad
  - $\mathcal{K}\ell(\mathcal{G})$, the Kleisli category for the Giry monad
  - $\mathbf{vNA}^{\mathrm{op}}$, (opposite) category of von Neumann algebras
- Uniform language with states, predicates, channels
  - *Channel-based* probability theory

EfProb = **Ef**fectus **Prob**ability

- Effectus theory provides a general setting for discrete/continuous/quantum probability, via
  - $\mathcal{K}\ell(\mathcal{D})$, the Kleisli category for the distribution monad
  - $\mathcal{K}\ell(\mathcal{G})$, the Kleisli category for the Giry monad
  - $\mathbf{vNA}^{\mathrm{op}}$, (opposite) category of von Neumann algebras
- Uniform language with states, predicates, channels
  - *Channel-based* probability theory
- New perspectives in probability theory
  - Conditioning as an operation on a state σ and a predicate p, yielding an updated state σ/p
  - A predicate/state transformer semantics for Bayesian learning [Jacobs & Zanasi, MFPS 2016]

EfProb = **Ef**fectus **Prob**ability

- Effectus theory provides a general setting for discrete/continuous/quantum probability, via
  - $\mathcal{K}\ell(\mathcal{D})$, the Kleisli category for the distribution monad
  - $\mathcal{K}\ell(\mathcal{G})$, the Kleisli category for the Giry monad
  - $\mathbf{vNA}^{\mathrm{op}}$, (opposite) category of von Neumann algebras
- Uniform language with states, predicates, channels
  - *Channel-based* probability theory
- New perspectives in probability theory
  - Conditioning as an operation on a state σ and a predicate p, yielding an updated state σ/p
  - A predicate/state transformer semantics for Bayesian learning [Jacobs & Zanasi, MFPS 2016]

➡ EfProb: a Python library for discrete/continuous/quantum probability calculations

Talk plan

1. Language: States, Predicates, and Channels

2. Implementation and Demo

Talk plan

1. Language: States, Predicates, and Channels

2. Implementation and Demo

Discrete probability, states & predicates

Discrete probability, states & predicates

A **state** on a set X is a distribution $\sigma \in \mathcal{D}(X)$, i.e. $\sigma\colon X \to [0,1]$ with $\sum_x \sigma(x) = 1$. Often denoted as a formal convex sum, e.g. $\frac{1}{3}|a\rangle + \frac{1}{2}|b\rangle + \frac{1}{6}|c\rangle$, for $a, b, c \in X$

Discrete probability, states & predicates

A **state** on a set X is a distribution $\sigma \in \mathcal{D}(X)$, i.e. $\sigma \colon X \to [0,1]$ with $\sum_x \sigma(x) = 1$. Often denoted as a formal convex sum, e.g. $\frac{1}{3}|a\rangle + \frac{1}{2}|b\rangle + \frac{1}{6}|c\rangle$, for $a, b, c \in X$

Operations for states $\sigma \in \mathcal{D}(X)$, $\tau \in \mathcal{D}(Y)$, $\omega \in \mathcal{D}(X \times Y)$

- Joint state $\sigma \otimes \tau \in \mathcal{D}(X \times Y)$ by $(\sigma \otimes \tau)(x,y) = \sigma(x) \cdot \tau(y)$
- Marginal state $\omega_1 \in \mathcal{D}(X)$ by $\omega_1(x) = \sum_y \omega(x,y)$

Discrete probability, states & predicates

A **state** on a set X is a distribution $\sigma \in \mathcal{D}(X)$, i.e. $\sigma\colon X \to [0,1]$ with $\sum_x \sigma(x) = 1$. Often denoted as a formal convex sum, e.g. $\frac{1}{3}|a\rangle + \frac{1}{2}|b\rangle + \frac{1}{6}|c\rangle$, for $a, b, c \in X$

Operations for states $\sigma \in \mathcal{D}(X)$, $\tau \in \mathcal{D}(Y)$, $\omega \in \mathcal{D}(X \times Y)$

- Joint state $\sigma \otimes \tau \in \mathcal{D}(X \times Y)$ by $(\sigma \otimes \tau)(x, y) = \sigma(x) \cdot \tau(y)$
- Marginal state $\omega_1 \in \mathcal{D}(X)$ by $\omega_1(x) = \sum_y \omega(x, y)$

A **predicate** on a set X is (any) function $p\colon X \to [0,1]$

Events $E \subseteq X$ as indicator functions $\mathbf{1}_E\colon X \to [0,1]$

Discrete probability, states & predicates

A **state** on a set X is a distribution $\sigma \in \mathcal{D}(X)$, i.e. $\sigma \colon X \to [0,1]$ with $\sum_x \sigma(x) = 1$. Often denoted as a formal convex sum, e.g. $\frac{1}{3}|a\rangle + \frac{1}{2}|b\rangle + \frac{1}{6}|c\rangle$, for $a, b, c \in X$

Operations for states $\sigma \in \mathcal{D}(X)$, $\tau \in \mathcal{D}(Y)$, $\omega \in \mathcal{D}(X \times Y)$

- Joint state $\sigma \otimes \tau \in \mathcal{D}(X \times Y)$ by $(\sigma \otimes \tau)(x, y) = \sigma(x) \cdot \tau(y)$
- Marginal state $\omega_1 \in \mathcal{D}(X)$ by $\omega_1(x) = \sum_y \omega(x, y)$

A **predicate** on a set X is (any) function $p \colon X \to [0,1]$

Events $E \subseteq X$ as indicator functions $\mathbf{1}_E \colon X \to [0,1]$

- Truth $\mathbf{1}_X$, falsity $\mathbf{0}_X$, and negation $({\sim}p)(x) = 1 - p(x)$
- Sequential conjunction $p \,\&\, q$ by $(p \,\&\, q)(x) = p(x) \cdot q(x)$

Discrete prob., validity & conditioning

For a state $\sigma \in \mathcal{D}(X)$ and a predicate $p \in [0,1]^X$, the validity

$$\sigma \models p \quad := \quad \sum_x \sigma(x) \cdot p(x) \quad \in [0,1]$$

- *Probability that p holds in σ*

Discrete prob., validity & conditioning

For a state $\sigma \in \mathcal{D}(X)$ and a predicate $p \in [0,1]^X$, the validity

$$\sigma \models p \quad := \quad \sum_x \sigma(x) \cdot p(x) \quad \in [0,1]$$

- *Probability that p holds in σ*

When the validity $\sigma \models p$ is non-zero, the conditional state $\sigma/p \in \mathcal{D}(X)$ ('σ given p') defined by

$$(\sigma/p)(x) \quad := \quad \frac{\sigma(x) \cdot p(x)}{\sigma \models p}$$

- *Updated state after we observe that p holds*

Discrete prob., validity & conditioning

For a state $\sigma \in \mathcal{D}(X)$ and a predicate $p \in [0,1]^X$, the validity

$$\sigma \models p \quad := \quad \sum_x \sigma(x) \cdot p(x) \quad \in [0,1]$$

- *Probability that p holds in σ*

When the validity $\sigma \models p$ is non-zero, the conditional state $\sigma/p \in \mathcal{D}(X)$ ('σ given p') defined by

$$(\sigma/p)(x) \quad := \quad \frac{\sigma(x) \cdot p(x)}{\sigma \models p}$$

- *Updated state after we observe that p holds*

Comparing to traditional notation, for events $A, B \subseteq X$

- $\mathrm{P}(A) \quad = \quad \sigma \models \mathbf{1}_A$
- $\mathrm{P}(A \mid B) \quad = \quad \sigma/\mathbf{1}_B \models \mathbf{1}_A$

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map' $c\colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map'
$c\colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$
(X-indexed states on Y, or a Kleisli map $X \to \mathcal{D}(Y)$)

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map'
$c\colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$
(X-indexed states on Y, or a Kleisli map $X \to \mathcal{D}(Y)$)

State/predicate transformation by a channel $c\colon X \to Y$

For a state $\sigma \in \mathcal{D}(X)$,

state $c \gg \sigma \in \mathcal{D}(Y)$ by $(c \gg \sigma)(y) = \sum_x c(x,y) \cdot \sigma(x)$

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map' $c\colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$ (X-indexed states on Y, or a Kleisli map $X \to \mathcal{D}(Y)$)

State/predicate transformation by a channel $c\colon X \to Y$

For a state $\sigma \in \mathcal{D}(X)$,

state $c \gg \sigma \in \mathcal{D}(Y)$ by $(c \gg \sigma)(y) = \sum_x c(x,y) \cdot \sigma(x)$

For a predicate $q \in [0,1]^Y$,

predicate $c \ll q \in [0,1]^X$ by $(c \ll q)(x) = \sum_y q(y) \cdot c(x,y)$

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map'
$c \colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$
(X-indexed states on Y, or a Kleisli map $X \to \mathcal{D}(Y)$)

State/predicate transformation by a channel $c \colon X \to Y$

$\mathrm{Stat}(X) \xrightarrow{c \gg} \mathrm{Stat}(Y)$

For a state $\sigma \in \mathcal{D}(X)$,

state $c \gg \sigma \in \mathcal{D}(Y)$ by $(c \gg \sigma)(y) = \sum_x c(x,y) \cdot \sigma(x)$

$\mathrm{Pred}(X) \xleftarrow{c \ll} \mathrm{Pred}(Y)$

For a predicate $q \in [0,1]^Y$,

predicate $c \ll q \in [0,1]^X$ by $(c \ll q)(x) = \sum_y q(y) \cdot c(x,y)$

Discrete probability, channels

A **channel** of type $X \to Y$ is a 'stochastic map'
$c\colon X \times Y \to [0,1]$ such that $\sum_y c(x,y) = 1$ for all $x \in X$
(X-indexed states on Y, or a Kleisli map $X \to \mathcal{D}(Y)$)

State/predicate transformation by a channel $c\colon X \to Y$

For a state $\sigma \in \mathcal{D}(X)$,

$$\mathrm{Stat}(X) \xrightarrow{c \gg} \mathrm{Stat}(Y)$$

state $c \gg \sigma \in \mathcal{D}(Y)$ by $(c \gg \sigma)(y) = \sum_x c(x,y) \cdot \sigma(x)$

For a predicate $q \in [0,1]^Y$,

$$\mathrm{Pred}(X) \xleftarrow{c \ll} \mathrm{Pred}(Y)$$

predicate $c \ll q \in [0,1]^X$ by $(c \ll q)(x) = \sum_y q(y) \cdot c(x,y)$

Forward and backward learning are described as
$c \gg (\sigma/p)$ and $\sigma/(c \ll q)$ [Jacobs & Zanasi, MFPS 2016]

Uniform language: continuous / quantum

For **continuous probability**:

States are probability density functions $\sigma\colon X \to \mathbb{R}_{\geq 0}$ such that $\int \sigma(x)\,\mathrm{d}x = 1$ (where $X \subseteq \mathbb{R}^n$)

Predicates are (measurable) functions $p\colon X \to [0,1]$

Channels are $c\colon X \times Y \to \mathbb{R}_{\geq 0}$ such that $\int c(x,y)\,\mathrm{d}y = 1$ for all $x \in X$

Uniform language: continuous / quantum

For **continuous probability**:

States are probability density functions $\sigma\colon X \to \mathbb{R}_{\geq 0}$ such that $\int \sigma(x)\,\mathrm{d}x = 1$ (where $X \subseteq \mathbb{R}^n$)

Predicates are (measurable) functions $p\colon X \to [0,1]$

Channels are $c\colon X \times Y \to \mathbb{R}_{\geq 0}$ such that $\int c(x,y)\,\mathrm{d}y = 1$ for all $x \in X$

- Operations such as $\sigma \models p$, $c \gg \sigma$ by replacing $\sum$ by $\int$

Uniform language: continuous / quantum

For **continuous probability**:

States are probability density functions $\sigma\colon X \to \mathbb{R}_{\geq 0}$ such that $\int \sigma(x)\,\mathrm{d}x = 1$ (where $X \subseteq \mathbb{R}^n$)

Predicates are (measurable) functions $p\colon X \to [0,1]$

Channels are $c\colon X \times Y \to \mathbb{R}_{\geq 0}$ such that $\int c(x,y)\,\mathrm{d}y = 1$ for all $x \in X$

- Operations such as $\sigma \models p$, $c \gg \sigma$ by replacing $\sum$ by $\int$

For **quantum probability**:

States are positive trace-one matrices

Predicates are positive matrices with eigenvalues ≤ 1

Channels are completely positive trace-preserving maps

Uniform language: continuous / quantum

For **continuous probability**:

States are probability density functions $\sigma\colon X \to \mathbb{R}_{\geq 0}$ such that $\int \sigma(x)\,\mathrm{d}x = 1$ (where $X \subseteq \mathbb{R}^n$)

Predicates are (measurable) functions $p\colon X \to [0,1]$

Channels are $c\colon X \times Y \to \mathbb{R}_{\geq 0}$ such that $\int c(x,y)\,\mathrm{d}y = 1$ for all $x \in X$

- Operations such as $\sigma \models p$, $c \gg \sigma$ by replacing $\sum$ by $\int$

For **quantum probability**:

States are positive trace-one matrices

Predicates are positive matrices with eigenvalues ≤ 1

Channels are completely positive trace-preserving maps

- Operations $\sigma \models p$ etc. exist in quantum theory.
  Note: $p \,\&\, q \neq q \,\&\, p$

Talk plan

1. Language: States, Predicates, and Channels
2. Implementation and Demo

EfProb: Implementation in Python

- For discrete (and quantum) probability, EfProb is restricted to finite sets, reducing the calculations into manipulations of arrays/matrices

EfProb: Implementation in Python

- For discrete (and quantum) probability, EfProb is restricted to finite sets, reducing the calculations into manipulations of arrays/matrices
- Continuous probability involves numerical integration (by SciPy library)

EfProb: Implementation in Python

- For discrete (and quantum) probability, EfProb is restricted to finite sets, reducing the calculations into manipulations of arrays/matrices
- Continuous probability involves numerical integration (by SciPy library)
- *Not* sampling-based computation via MCMC
  - cf. BLOG, Church, Anglican, ...

EfProb: Implementation in Python

- For discrete (and quantum) probability, EfProb is restricted to finite sets, reducing the calculations into manipulations of arrays/matrices
- Continuous probability involves numerical integration (by SciPy library)
- *Not* sampling-based computation via MCMC
  - cf. BLOG, Church, Anglican, ...
- Works well for moderate-size problems; but not meant for large-scale computation

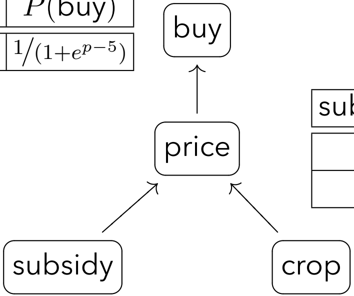
EfProb: Implementation in Python

- For discrete (and quantum) probability, EfProb is restricted to finite sets, reducing the calculations into manipulations of arrays/matrices
- Continuous probability involves numerical integration (by SciPy library)
- *Not* sampling-based computation via MCMC
    - cf. BLOG, Church, Anglican, ...
- Works well for moderate-size problems; but not meant for large-scale computation
- Can be helpful eg. for research and teaching

Demo: Hybrid Bayesian Network example

From [Cobb & Shenoy, 2006]

price	$P(\text{buy})$
p	$1/(1+e^{p-5})$



subsidy	crop	$P(\text{price})$
t	c	$\mathcal{N}(20-c,1)$
f	c	$\mathcal{N}(10-c,1)$

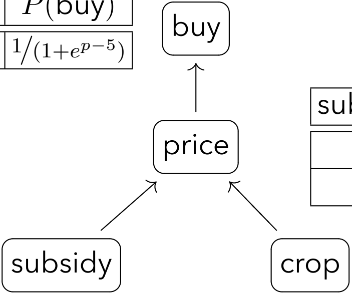
$P(\text{subsidy})$
0.7

$P(\text{crop})$
$\mathcal{N}(5,1)$

Demo: Hybrid Bayesian Network example

From [Cobb & Shenoy, 2006]

price	$P(\text{buy})$
p	$1/(1+e^{p-5})$



subsidy	crop	$P(\text{price})$
t	c	$\mathcal{N}(20-c,1)$
f	c	$\mathcal{N}(10-c,1)$

$P(\text{subsidy})$
0.7

$P(\text{crop})$
$\mathcal{N}(5,1)$

- ‘subsidy’ and ‘buy’ have discrete domain $\{t, f\}$
- ‘crop’ and ‘price’ have continuous domain $\mathbb{R}$
- $\mathcal{N}(\mu,\sigma)$ normal (Gaussian) distribution

Conclusions

- Effectus theory offers a uniform language for probability, based on states, predicates, and channels, with validity, conditioning, etc.
- EfProb implements the ideas as a Python library, useful for probability calculations
  - `efprob_dc` for discrete/continuous
  - `efprob_qu` for quantum

Conclusions

- Effectus theory offers a uniform language for probability, based on states, predicates, and channels, with validity, conditioning, etc.
- EfProb implements the ideas as a Python library, useful for probability calculations
  - `efprob_dc` for discrete/continuous
  - `efprob_qu` for quantum
- EfProb Manual is available, containing a lot of examples

Conclusions

- Effectus theory offers a uniform language for probability, based on states, predicates, and channels, with validity, conditioning, etc.
- EfProb implements the ideas as a Python library, useful for probability calculations
  - `efprob_dc` for discrete/continuous
  - `efprob_qu` for quantum
- EfProb Manual is available, containing a lot of examples
- Ongoing: disintegration

Conclusions

- Effectus theory offers a uniform language for probability, based on states, predicates, and channels, with validity, conditioning, etc.
- EfProb implements the ideas as a Python library, useful for probability calculations
  - `efprob_dc` for discrete/continuous
  - `efprob_qu` for quantum
- EfProb Manual is available, containing a lot of examples
- Ongoing: disintegration

Thank you!

Please visit `https://efprob.cs.ru.nl/`