Lecture 20

Probabilistic PL

the Church Programming Language

Ras Bodik
Ali and Mangpo

Hack Your Language!
CS164: Introduction to Programming Languages and Compilers, Spring 2013
UC Berkeley

Today

Scheme refresher

Church is an extension of Scheme

Generative Models

Conditioning

Scheme Refresher

http://projects.csail.mit.edu/church/wiki/Church_Basics:_Exercises

Scheme

Define symbols (variables)

```
(define x 3)
(define y 4)
```

Define functions

```
;; long way
(define f2 (lambda (a b) (expt (+ a b) 3)))

;; short way
(define (f1 a b) (expt (+ a b) 3))
```

Generative Models

http://projects.csail.mit.edu/church/wiki/Generative_Models

Random Samples

Flip a coin

```
(flip)
```

Flip a biased coin

```
(flip 0.7)
```

Draw a histogram of 1000 samples

```
(hist (repeat 1000 flip) "Flips")
```

A Causal Model for Medical Diagnosis

representing causal knowledge:

```
(define lung-cancer (flip 0.01))
(define cold (flip 0.2))

(define cough (or cold lung-cancer))

cough
```

This program generates random conditions for a patient in a doctor's office. It first specifies the base rates of two diseases the patient could have: lung cancer is rare while a cold is common, and there is an independent chance of having each disease. The program then specifies a process for generating a common symptom of these diseases -- an effect with two possible causes: The patient coughs if they have a cold or lung cancer (or both).

A more complex model

```
(define lung-cancer (flip 0.01))
(define TB (flip 0.005))
(define cold (flip 0.2))
(define stomach-flu (flip 0.1))
(define other (flip 0.1))

(define cough (or (and cold (flip 0.5))
                  (and lung-cancer (flip 0.3))
                  (and TB (flip 0.7))
                  (and other (flip 0.01))))

(define fever (or (and cold (flip 0.3)) (and stomach-flu (flip 0.5)) (andTB (flip 0.1)) (and other (flip 0.01))))
(define chest-pain (or (and lung-cancer (flip 0.5)) (and TB (flip 0.5))(and other(flip 0.01))))
(define shortness-of-breath (or (and lung-cancer (flip 0.5)) (and TB (flip0.2)) (and other (flip 0.01))))

(list  "cough" cough
       "fever" fever
       "chest-pain" chest-pain
       "shortness-of-breath" shortness-of-breath)
```

Modeling coins

```
(define (make-coin weight)
      (lambda () (if (flip weight) 'h 't)))

(define fair-coin (make-coin 0.5))
(define trick-coin (make-coin 0.95))
(define bent-coin (make-coin 0.25))

(hist (repeat 20 fair-coin) "20 fair coin flips")
(hist (repeat 20 trick-coin) "20 trick coin flips")
(hist (repeat 20 bent-coin) "20 bent coin flips")

"done"
```

Bending a coin

```
(define (make-coin weight)
      (lambda () (if (flip weight) 'h 't)))

(define (bend coin)
  (lambda () (if (equal? (coin) 'h)
              ( (make-coin 0.7) )
              ( (make-coin 0.1) ) )))

(define fair-coin (make-coin 0.5))
(define bent-coin (bend fair-coin))

(hist (repeat 100 bent-coin) "bent coin")

"done"
```

Persistent Randomness

http://projects.csail.mit.edu/church/wiki/Generative_Models

Persisting random decisions

What is the value of this expression?

```
(equal? (flip) (flip))
```

And this one?

```
(define (eye-color person) (uniform-draw '(blue green brown)))

(list
 (eye-color 'bob)
 (eye-color 'alice)
 (eye-color 'bob) )
```

Persitence

The mem high-order function:

```
(define mem-flip (mem flip))
(equal? (mem-flip) (mem-flip))
```

The second example

```
(define eye-color
 (mem
  (lambda (person) (uniform-draw '(blue green brown)))))

(list
 (eye-color 'bob)
 (eye-color 'alice)
 (eye-color 'bob) )
```

Tug of War

```
(define strength (mem (lambda (person) (if (flip) 10 5))))

(define lazy (lambda (person) (flip (/ 1 3))))

(define (total-pulling team)
 (apply +
       (map (lambda (person)
(if (lazy person) (/ (strength person) 2)(strength person)))
           team)))

(define (winner team1 team2)
      (if (< (total-pulling team1) (total-pullingteam2))
            team2
            team1))
```

Tug of War

```
...

(list "Tournament results:"
    (winner '(alice bob) '(sue tom))
    (winner '(alice bob) '(sue tom))
    (winner '(alice sue) '(bob tom))
    (winner '(alice sue) '(bob tom))
    (winner '(alice tom) '(bob sue))
    (winner '(alice tom) '(bob sue)))
```

Conditioning

http://projects.csail.mit.edu/church/wiki/Conditioning

query

```
(query
  generative-model
  what-we-want-to-know
  what-we-know)
```

Model

```
(define A (if (flip) 1 0))
(define B (if (flip) 1 0))
(define C (if (flip) 1 0))
(define D (+ A B C))
D
```

query

```
(define (take-sample)
 (rejection-query

  (define A (if (flip) 1 0))
  (define B (if (flip) 1 0))
  (define C (if (flip) 1 0))
  (define D (+ A B C))

  A

  (equal? D 3)
  )
 )
(hist (repeat 100 take-sample) "Value of A, given that D is 3")
```

Answer this

- "The probability of breast cancer is 1% for a woman at 40 who participates in a routine screening. If a woman has breast cancer, the probability is 80% that she will have a positive mammography. If a woman does not have breast cancer, the probability is 9.6% that she will also have a positive mammography. A woman in this age group had a positive mammography in a routine screening.
- What is the probability that she actually has breast cancer?"

Model

```
(define samples
 (mh-query 100 100
   (define breast-cancer (flip 0.01))

   (define positive-mammogram (if breast-cancer (flip 0.8) (flip 0.096)))

   breast-cancer

   positive-mammogram
 )
)
(hist samples "breast cancer")
```