CS 4110

Probabilistic Programming
- *Optional*, choose-your-own-adventure HW #9 out
- Due on Friday, if you choose to turn it in.
What and Why

It's not about writing software.
What and Why

Probabilistic programming is a tool for statistical modeling.

OR

A probabilistic programming language is a plain old programming language with `rand(3)` and a suite of fancy analysis tools for understanding its probabilistic behavior.
An Example Model

- Takes CS 4780
- Takes CS 4242
- Takes CS 4110

Cloud of Unknowing

- Paper 1 Relevant
- Paper 2 Relevant
- Paper 3 Relevant

PL+stats, PL, stats
A Model for Humans

- Interest in Stats
  - Takes CS 4780
  - Takes CS 4242

- Interest in PL
  - Takes CS 4110
A Model for Humans

- Interest in Stats
  - Takes CS 4780

- Busy?
  - Takes CS 4242

- Interest in PL
  - Takes CS 4110
A Model for Humans

- Interest in Stats
  - Takes CS 4780
  - Takes CS 4242
- Busy?
- Interest in PL
  - Takes CS 4110

- Interest in Stats
- Paper 1 Relevant
- Paper 2 Relevant
- Paper 3 Relevant
A Model for Humans

\[ \Pr[A_{\text{NIPS}} | I_{\text{stats}} \land B] = 0.3 \]
\[ \Pr[A_{\text{NIPS}} | I_{\text{stats}} \land \neg B] = 0.8 \]
\[ \Pr[A_{\text{NIPS}} | \neg I_{\text{stats}}] = 0.1 \]

\[ \ldots \]
\[ \Pr[A_{\text{Dagstuhl}} | I_{\text{stats}} \land I_{\text{PL}}] = 0.3 \]
\[ \Pr[A_{\text{Dagstuhl}} | I_{\text{stats}} \land I_{\text{PL}} \land \neg B] = 0.8 \]
\[ \Pr[A_{\text{Dagstuhl}} | \neg (I_{\text{stats}} \lor I_{\text{PL}})] = 0.1 \]

\[ \ldots \]

\[ R_1 \sim I_{\text{PL}} \land I_{\text{stats}} \]
\[ R_2 \sim I_{\text{PL}} \]
\[ R_3 \sim I_{\text{stats}} \]

Whither reuse?
Whither abstraction?
Whither intermediate variables?
Writing even this tiny model feels like **drudgery**.

(and we haven’t even gotten to the hard part yet)
• What and Why

• **The Basics and Examples**

• Applications

• Current Problems
webppl is a small but feature-rich probabilistic programming language embedded in Javascript.

```javascript
// Conference attendance.
var attendance = function(i_pl, i_stats, busy) {
  var attendance = function (interest, busy, weight) {
    if (interest) {
      return busy & flip(0.3 * weight) & flip(0.3 * weight);
    }
  }
  return attendance;
};
```
Our First Probabilistic Program

```
var b = flip(0.5);
b ? "yes" : "no"
```
var roll = function () {
  var die1 = randomInteger(6) + 1;
  var die2 = randomInteger(6) + 1;
  return die1 + die2;
}

Enumerate(roll)
Our Basic Model in webppl

```javascript
// Class attendance model.
var attendance = function(i_pl, i_stats, busy) {
  var interest = function( ) {
    return busy ? flip(0.3) : flip(0.8);
  } else {
    return flip(0.1);
  }

  var a_4110 = attendance(i_pl, busy);
  var a_4780 = attendance(i_stats, busy);
  var a_4242 = attendance(i_pl && i_stats, busy);

  return {cs4110: a_4110, cs4780: a_4780, cs4242: a_4242};
}

// Relevance of our three papers.
var relevance = function(i_pl, i_stats) {
  var rel1 = i_pl && i_stats;
  var rel2 = i_pl;
  var rel3 = i_stats;

  return {paper1: rel1, paper2: rel2, paper3: rel3};
}

// A combined model.
var model = function() {
  // Some even random priors for our "student profile."
  var i_pl = flip(0.5);
  var i_stats = flip(0.5);
  var busy = flip(0.5);

  return [relevance(i_pl, i_stats), attendance(i_pl, i_stats, busy)];
}

var dist = Enumerate(model);
viz.auto(dist);
```
var roll = function () {
    var die1 = randomInteger(6) + 1;
    var die2 = randomInteger(6) + 1;
    if (!(die1 === 4 || die2 === 4)) {
        factor(-Infinity);
    }
    return die1 + die2;
}

Enumerate(roll)
Conditioning on Observations

// Discard any executions that don’t sum to 10.
var out = die1 + die2;
if (out !== 10) {
    factor(-Infinity);
}

// Return the values on the dice.
return [die1, die2];
Recommending Papers

// Require my conference attendance.
var att = attendance(i_pl, i_stats, busy);
require(att.cs4110 && att.cs4242 && !att.cs4780);

return relevance(i_pl, i_stats);
Inference Algorithms

Enumerate is the simplest possible inference strategy.
• What and Why

• The Basics and Examples

• Applications

• Current Problems
TrueSkill

Measure Transformer Semantics for Bayesian Machine Learning

Johannes Borgström Andrew D. Gordon
Michael Greenberg James Margetson Jurgen Van Gael

// prior distributions, the hypothesis
let skill() = random (Gaussian(10.0,20.0))
let Alice,Bob,Cyd = skill(),skill(),skill()
// observe the evidence
let performance player = random (Gaussian(player,1.0))
observe (performance Alice > performance Bob) //Alice beats Bob
observe (performance Bob > performance Cyd) //Bob beats Cyd
observe (performance Alice > performance Cyd) //Alice beats Cyd
// return the skills
Alice,Bob,Cyd
webppl Vision Demo

```javascript
var newScore = targetImage.distance(finalGeneratedImage) / 1000; // Increase factor(newScore);

if (!showOutputImage) {
    finalGeneratedImage.destroy();
}

counter.push(1);

return lines,
}, 2500);

// Show target image for comparison
loadImage(Draw(50, 50, true), "/assets/img/beach.png")
```
## Models

<table>
<thead>
<tr>
<th>Concept Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inducing Arithmetic Functions</td>
</tr>
<tr>
<td>Causal Support</td>
</tr>
<tr>
<td>Rational Rules</td>
</tr>
<tr>
<td>Word Learning as Bayesian Inference</td>
</tr>
<tr>
<td>Bayes Net Structure Learning</td>
</tr>
</tbody>
</table>
• What and Why
• The Basics and Examples
• Applications
• Current Problems
The R2 Probabilistic Programming Tool is a research project within the Programming Languages and Tools group at Microsoft Research on probabilistic programming. Our goal is to build a user friendly and scalable probabilistic programming system by employing powerful techniques from language design, program analysis and verification.

Details
Type          Download
File Name     r2-0.0.1.zip

Note: By installing, copying, or otherwise using R2, you are covered by the Microsoft software license terms available at research.microsoft.com/deeplearning/LICENSE.txt.
R2’s weakest preconditions

```javascript
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;

// Discard any executions that don’t sum to 10.
var out = die1 + die2;
require(out === 10);
```

wasted work!
R2’s weakest preconditions

```javascript
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;

require(
    (die1 == 3 && die2 == 7) || ...);
var out = die1 + die2;
require(out === 10);
```
R2’s weakest preconditions

```javascript
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;
require((die1 == 3 && die2 == 7) || ...);
var out = die1 + die2;
```
Probabilistic assertions: design goals

Work on a messy, mainstream language (C and C++)

Efficiently check statistical properties of the output

We don’t care about conditioning
\texttt{passert } e, \ p, \ c

e must hold \textbf{with probability } p
\textbf{at confidence } c
float obfuscated(float n) {
    return n + gaussian(0.0, 1000.0);
}

float average_salary(float* salaries) {
    total = 0.0;
    for (int i = 0; i < COUNT; ++i)
        total += obfuscated(salaries[i]);
    avg = total / len(salaries);
    p_avg = ...;
}

assert e, p, c

distribution extraction via symbolic execution
statistical optimizations verification

Bayesian network IR
distribution extraction
via symbolic execution

float obfuscated(float n) {
    return n + gaussian(0.0, 1000.0);
}

float average_salary(float* salaries) {
    total = 0.0;
    for (int i = 0; i < COUNT; ++i)
        total += obfuscated(salaries[i]);
    avg = total / len(salaries);
    p_avg = ...;
}

passert e, p, c

Bayesian network IR

✓
Distribution extraction: random draws are symbolic

\[ b = a + \text{gaussian}(0.0, 1.0) \]

symbolic heap

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4.2</td>
</tr>
<tr>
<td>b</td>
<td>4.2 + $G_{0,1}$</td>
</tr>
</tbody>
</table>
input: $a = 4.2$

$\rightarrow b = \text{gaussian}(0.0, 1.0)$
input: $a = 4.2$

$b = \text{gaussian}(0.0, 1.0)$

$c = a + b$
input: \( a = 4.2 \)  
\( b = \text{gaussian}(0.0, 1.0) \)  
\( c = a + b \)  
\( d = c + b \)
input: 

\[ a = 4.2 \]
\[ b = \text{gaussian}(0.0, 1.0) \]
\[ c = a + b \]
\[ d = c + b \]
input: \( a = 4.2 \)
\( b = \text{gaussian}(0.0, 1.0) \)
\( c = a + b \)
\( d = c + b \)

if \( b > 0.5 \)
\( e = 2.0 \)
else
\( e = 4.0 \)
input: $a = 4.2$
$b = \text{gaussian}(0.0, 1.0)$
$c = a + b$
$d = c + b$
if $b > 0.5$
  $e = 2.0$
else
  $e = 4.0$
passert $e \leq 3.0$,

0.9, 0.9
float obfuscated(float n) {
    return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
    total = 0.0;
    for (int i = 0; i < COUNT; ++i)
        total += obfuscated(salaries[i]);
    avg = total / len(salaries);
    p_avg = ...;
}

Bayesian network IR

distribution extraction
via symbolic execution

statistical
optimizations

verification
\[ X \sim G(\mu_X, \sigma_X^2) \]
\[ Y \sim G(\mu_Y, \sigma_Y^2) \]
\[ Z = X + Y \]
\[ \Rightarrow Z \sim G(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2) \]

\[ X \sim U(a, b) \]
\[ Y = cX \]
\[ \Rightarrow Y \sim U(ca, cb) \]

\[ X \sim U(a, b) \]
\[ Y \sim X \leq c \]
\[ a \leq c \leq b \]
\[ \Rightarrow Y \sim B\left(\frac{c - a}{b - a}\right) \]

\[ X_1, X_2, \ldots, X_n \sim D \]
\[ Y = \sum_i X_i \]
\[ \Rightarrow Y \sim G(n\mu_D, n\sigma_D^2) \]
float obfuscated(float n) {
    return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
    total = 0.0;
    for (int i = 0; i < COUNT; ++i)
        total += obfuscated(salaries[i]);
    avg = total / len(salaries);
    p_avg = ...;
}

assert e, p, c