Probabilistic & Approximate Computing

Sasa Misailovic
UIUC
In the previous episode...
TrueSkill: Model

**Player skill:** initially, we assume all players have similar (randomly assigned) skills, centered around some average:

\[ \text{Skill} \sim \text{Gaussian}(100,10) \]

**Player performance:** it is based on the skill, but can be either higher or lower, depending on the moment of inspiration:

\[ \text{Perf} \sim \text{Gaussian}(\text{Skill}, 15) \]

**Tournament scores:** Each player plays against each other, we can observe that a player with better performance won

\[ \text{Perf}_{\text{PlayerA}} > \text{Perf}_{\text{PlayerB}} \]
TrueSkill Example

```javascript
var trueSkill = function(){

    var skillA = gaussian(100, 10);
    var skillB = gaussian(100, 10);
    var skillC = gaussian(100, 10);

    var perfA1 = gaussian(skillA, 15), perfB1 = gaussian(skillB, 15);
    condition (perfA1 > perfB1);

    var perfB2 = gaussian(skillB, 15), perfC2 = gaussian(skillC, 15);
    condition (perfB2 > perfC2);

    var perfA3 = gaussian(skillA, 15), perfC3 = gaussian(skillC, 15);
    condition (perfA3 > perfC3);

    return skillA;
}

var res = Infer({method: 'MCMC', samples: 50000}, trueSkill)
print("Expected value: "+expectation(res));
viz.auto(res);
```
Inference with Continuous and Hybrid Models \textit{(Exact and Approximate)}

Sampling (Rejection – Church)

Sampling (MCMC – Church & Stan & Figaro)

Variational Inference (Fun & Infer.NET)

Exact Symbolic (PSI & Hakaru)
Approximate Inference

What queries to ask?

ClickGraph example  

Maximum of two Gaussians

Comparison of PSI (exact inference) and Infer.NET (Variational approximate inference)  
From “PSI: Exact Symbolic Inference for Probabilistic Programs”, CAV 2016
Markov Chain \textit{(discrete time)}

Graph \((N, E, P, n_0)\) with the following elements:

- **Nodes** \(n \in N\) represent states
- **Directed edges** \(e \in E \subseteq N \times N\) are transitions from parent node to child node
- **Transition probability function** \(P : E \to [0, 1]\) labels each edge with a probability of transition.
  - For each node \(n\), and outgoing edges \(e' \in \{(n, n') \in E\}\) it holds that \(\sum P(e') = 1\)
- **Starting node** \(n_0\)
Markov Chain Example

S1 -> S2 with probability 0.5
S2 -> S1 with probability 0.5
Markov Chain Example
Markov Chain Properties

Distribution after \( k \) steps of iterating the chain:

\[
\Pr(X_k | X_{k-1}, X_{k-2}, \ldots X_0) = \Pr(X_k | X_{k-1})
\]

*Markov property:* the next transition depends only on the current state, not on the history.

Note, \( \Pr(X_k = n' | X_{k-1} = n) = P((n, n')) \) – the transition probability between the edge \((n, n')\).

The probability of a trace is then equal to the product of the transition probabilities.
Example: Simulating a Dice Roll

Implement a die roll using a fair coin (*)

* Example from “Probabilistic Programming”, Gordon, Henzinger, Nori, Rajamani (ICSE-FoSE, 2014)
Example: Simulating a Dice Roll

```javascript
var diceroll_iter = function(x){ // x is the current state
    if ( 1<=x && x<=6) return x;

    var coin = flip(0.5);
    var xnext = // xnext is the next state
        x == 0x0? (coin? 0xA : 0xB) :
        x == 0xA? (coin? 0xC : 0xD) :
        x == 0xB? (coin? 0xE : 0xF) :
        x == 0xC? (coin? 0xA : 1):
        x == 0xD? (coin? 2: 3):
        x == 0xE? (coin? 4: 5):
        x == 0xF? (coin? 6: 0xB) : -100;

    return diceroll_iter(xnext);
}

var diceroll = function() { return diceroll_iter(0); }
```

var res = Infer({method: 'enumerate', maxExecutions: 24}, diceroll);
viz.auto(res);
Example: Simulating a Dice Roll

```javascript
var diceroll_iter = function(x) { ... }

var diceroll = function() { ... }

var tworolls = function() {
  var r1 = diceroll();
  var r2 = diceroll();

  condition (r1 + r2 > 9)
  return r1;
}

var res = Infer({method: 'enumerate',
                 maxExecutions: 200},
                tworolls);
viz.auto(res);
```
Hidden Markov Model

Has the base structure of a Markov model, but measurements may have noise
Hidden Markov Model

(Bayes Net representation)
Hidden Markov Model (Discrete)

```javascript
var transition = function(s) {
    return s ? flip(0.7) : flip(0.3)
}

var measure = function(s) {
    return s ? flip(0.9) : flip(0.1)
    //return s  // <- This is just a Markov model
}

var hmm = function(n) {
    var prev = (n==1) ? {states: [], measurements:[]} : hmm(n-1)

    var newState = transition(prev.states[prev.states.length-1])
    var newObs = measure(newState)

    return {states: prev.states.concat([newState]),
            measurements: prev.measurements.concat([newObs])}
}

var trueObs = [false, true, false, true, false]

var model = function(){
    var r = hmm(trueObs.length)
    condition(_.isEqual(r.observations, trueObs))
    return r.states
};
```

For motivation and more details, see: The Design and Implementation of Probabilistic Programming Languages, Chapter 4 (Early, incremental evidence) [http://dippl.org/chapters/04-factorseq.html](http://dippl.org/chapters/04-factorseq.html)