Probabilistic & Approximate Computing

Sasa Misailovic

UIUC

In the previous episode…

Medical Test Prognosis

```
var test_effective = function() {
 var PatientSick = flip(0.01);

 var PositiveTest =
   PatientSick? flip(0.8): flip(0.096);

 condition (PositiveTest == true);

 return PatientSick;
}

Infer ({method: 'enumerate'},
        test_effective)
```

Bayes' Rule

Belief Revision

Prior

Likelihood Distribution

Posterior
Distribution

$$\Pr(\theta \mid x) = \frac{\Pr(x \mid \theta) \cdot \Pr(\theta)}{\Pr(x)}$$

Normalization
Constant

Bayes' Rule

Belief Revision

$$\boxed{\begin{aligned} \theta &\equiv sick \\ x &\equiv test \end{aligned}}$$

$$\Pr(\theta \mid x) = \frac{\Pr(x \mid \theta) \cdot \Pr(\theta)}{\Pr(x)}$$

Bayes' Rule

Belief Revision

$$\boxed{\begin{aligned} \theta &\equiv sick \\ x &\equiv test \end{aligned}}$$

$$\Pr(sick \mid test) = \frac{\Pr(test \mid sick) \cdot \Pr(sick)}{\Pr(test)}$$

$$\begin{tikzcd} & \Pr(test) \\ \displaystyle\sum_{sick \in \{T,F\}} \Pr(test, sick) \arrow[ur, bend left] & \end{tikzcd}$$

(law of total probability)

Bayes' Rule

Belief Revision

$$\Pr(sick \mid test) = \frac{\Pr(test \mid sick) \cdot \Pr(sick)}{\Pr(test)}$$

$$\sum_{sick \in \{T,F\}} \Pr(test|sick) \cdot \Pr(sick)$$

Bayes' Rule

Belief Revision

$$\Pr(sick \mid test) = \frac{\Pr(test \mid sick) \cdot \Pr(sick)}{\Pr(test)}$$

$$\Pr(test|sick = T) \cdot \Pr(sick = T) +$$
$$\Pr(test|sick = F) \cdot \Pr(sick = F)$$

Bayes’ Rule

Belief Revision

$$\Pr(sick \mid test) = \frac{\Pr(test \mid sick) \cdot 0.01}{\Pr(test)}$$

$$\Pr(test|sick = T) \cdot 0.01 + \\ \Pr(test|sick = F) \cdot 0.99$$

Bayes' Rule

Belief Revision

$$\Pr(sick \mid test) = \frac{0.80 \cdot 0.01}{\Pr(test)}$$

$$0.800 \cdot 0.01 + 0.096 \cdot 0.99$$

Inference Process

Probabilistic model: quantitatively encodes a general knowledge (or assumptions/beliefs) about a domain

Evidence: Data obtained about the studied system

Query: Property of interest about the studied system

Inference: Procedure that ***answers*** a <u>query</u> about the probabilistic <u>model</u>, given <u>evidence</u>

Definitions from: Practical Probabilistic Programming, A Pfeffer (Book, 2016)

Patient Treatment Model

```
var test_effective = function() {
 var PatientSick = flip(0.01);

 var PositiveTest =
   PatientSick? flip(0.8): flip(0.096);

 condition (PositiveTest == true);

 return PatientSick;
}

Infer ({method: 'enumerate'},
        test_effective)
```

Bayesian Inference Tasks

Predict future (compute output distribution by forward simulation)

Infer the cause of events (learn most likely cause based on evidence)

Learn from past to better predict future (update prior beliefs based on new evidence)

Models

Express dependencies (or lack thereof) between the variables

- Direct (e.g.Y = X + 1)
- Indirect (e.g.,Y = X + 1; Z = Y * 2;)

Probabilistic programs are one representation of probabilistic models

Bayesian Nets

Alternative representation of probabilistic models

Graphical representation of dependencies among random variables:

- Nodes are variables
- Links from parent to child nodes are direct dependencies between variables
- Instead of full joint distribution, now terms $\Pr(X|parents(X))$.

The graph has no cycles! DAG

Variable Dependencies

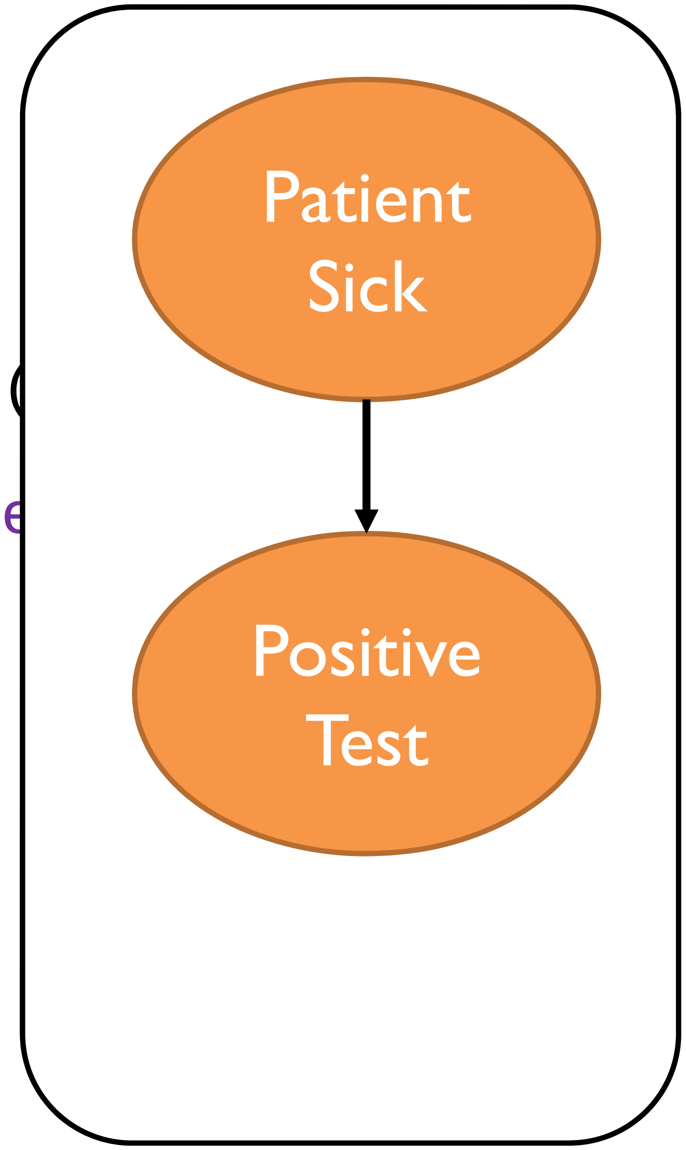
```
var test_effective = function()
 var PatientSick = flip(0.01);

 var PositiveTest =
   PatientSick? flip(0.8): flip(

 condition (PositiveTest == true

 return PatientSick;
}

Infer ({method: 'enumerate'},
        test_effective)
```



Variable Dependencies

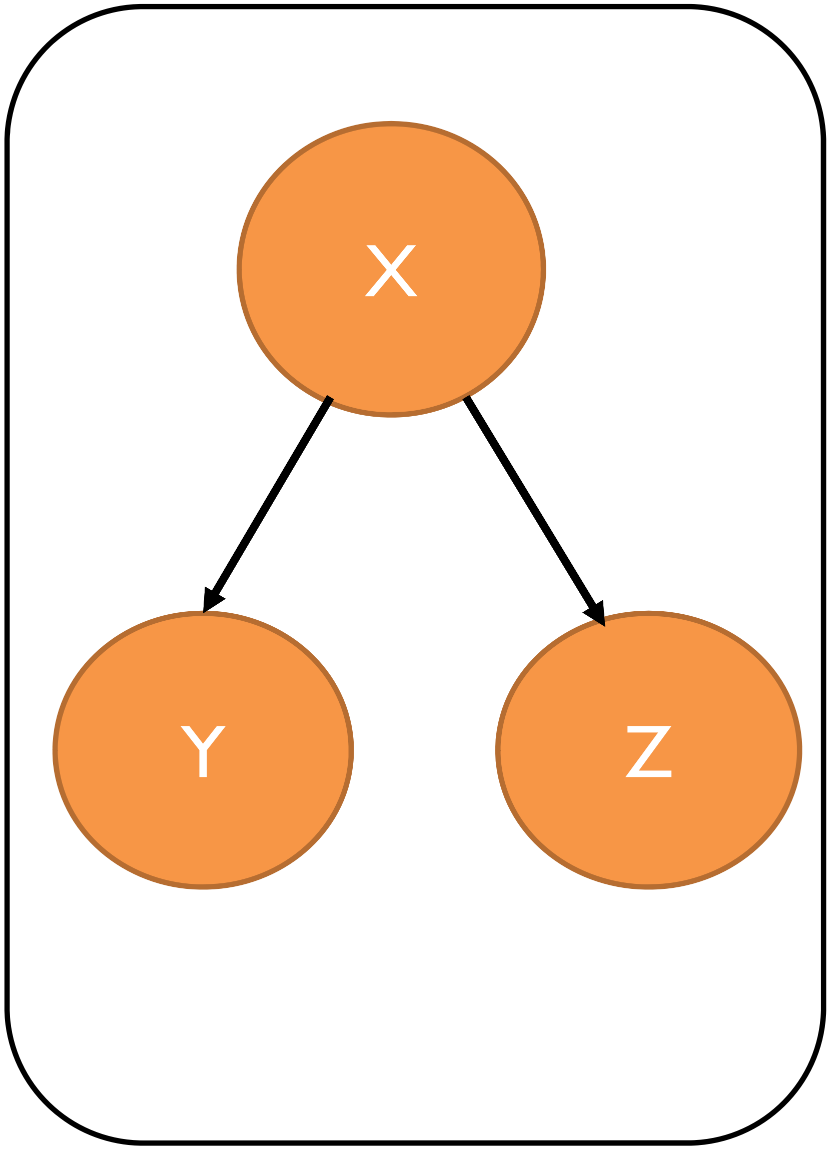
```
var test_x = function() {
 var x = flip(0.50);

 var y = x?
   flip(0.1): flip(0.2);

 var z = x?
    flip(0.3): flip(0.4);

 condition(x == 1)

 return [y, z]
}
```



Variable Dependencies

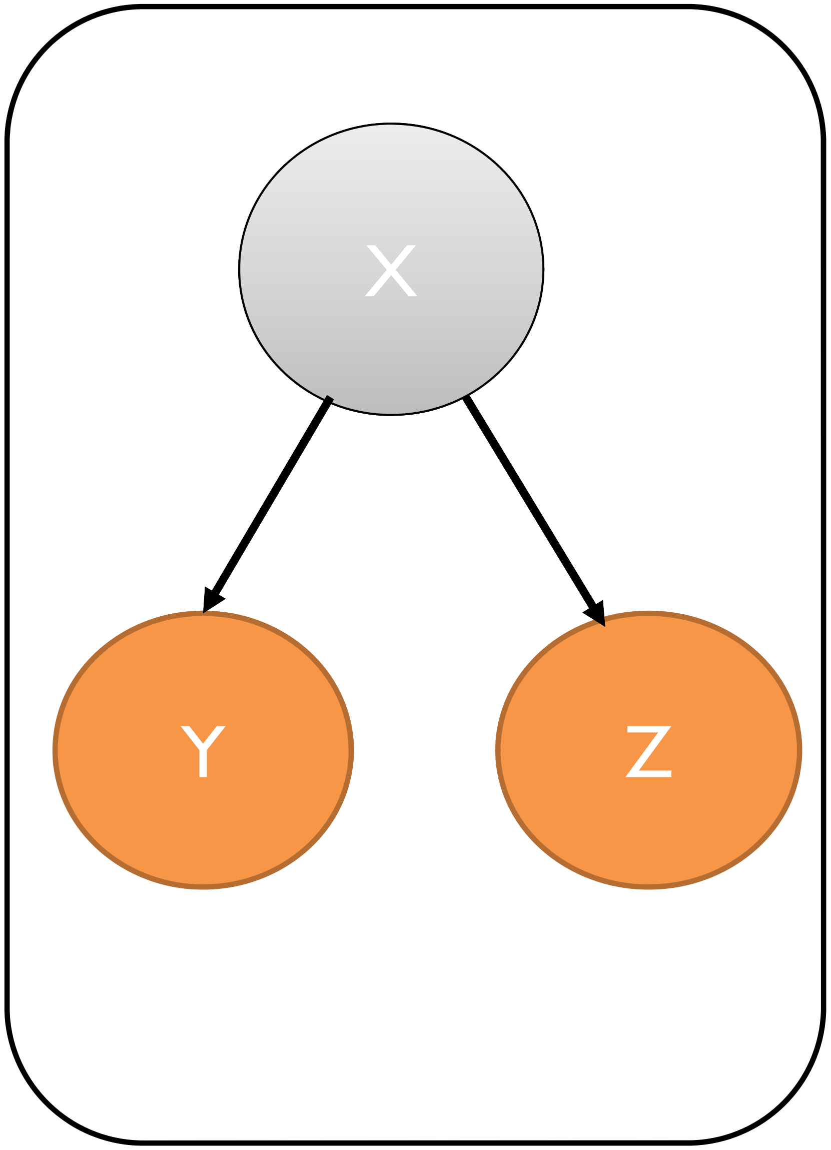
```
var test_x = function() {
 var x = flip(0.50);

 var y = x?
   flip(0.1): flip(0.2);

 var z = x?
    flip(0.3): flip(0.4);

 condition(x == 1)

 return [y, z]
}
```



Reminder: Independence

Definition:

$$\boldsymbol{Pr(X, Y) = Pr(X) \cdot Pr(Y)}$$

But also*:

$$\boldsymbol{Pr(X \mid Y) = Pr(X)}$$
$$\boldsymbol{Pr(Y \mid X) = Pr(Y)}$$

*Using the fact that for any two variables $\boldsymbol{Pr(X, Y) = Pr(X|Y) \cdot Pr(Y)}$

Variable Dependencies

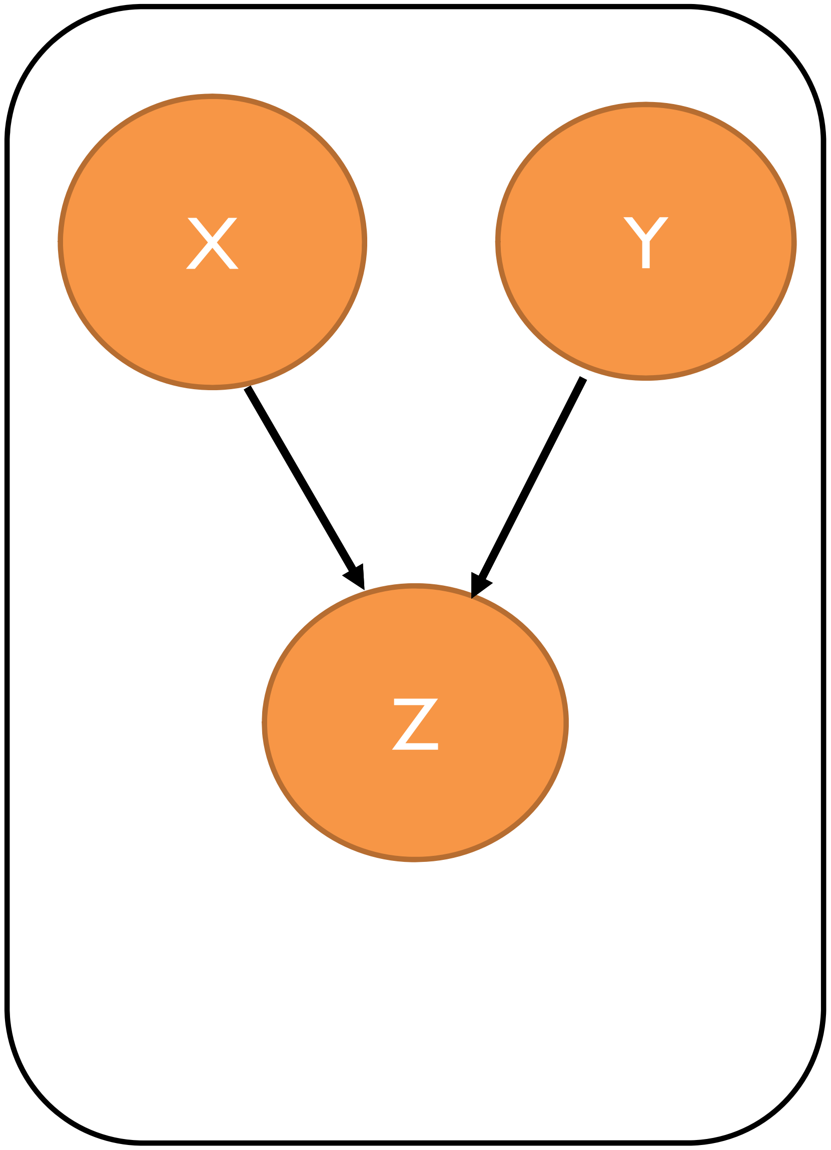
```
var test_z = function(){
 var x = flip(0.50);

 var y = flip(0.1);

 var z = x+y;

 condition(z == 1);

 return x;
}
```



Variable Dependencies

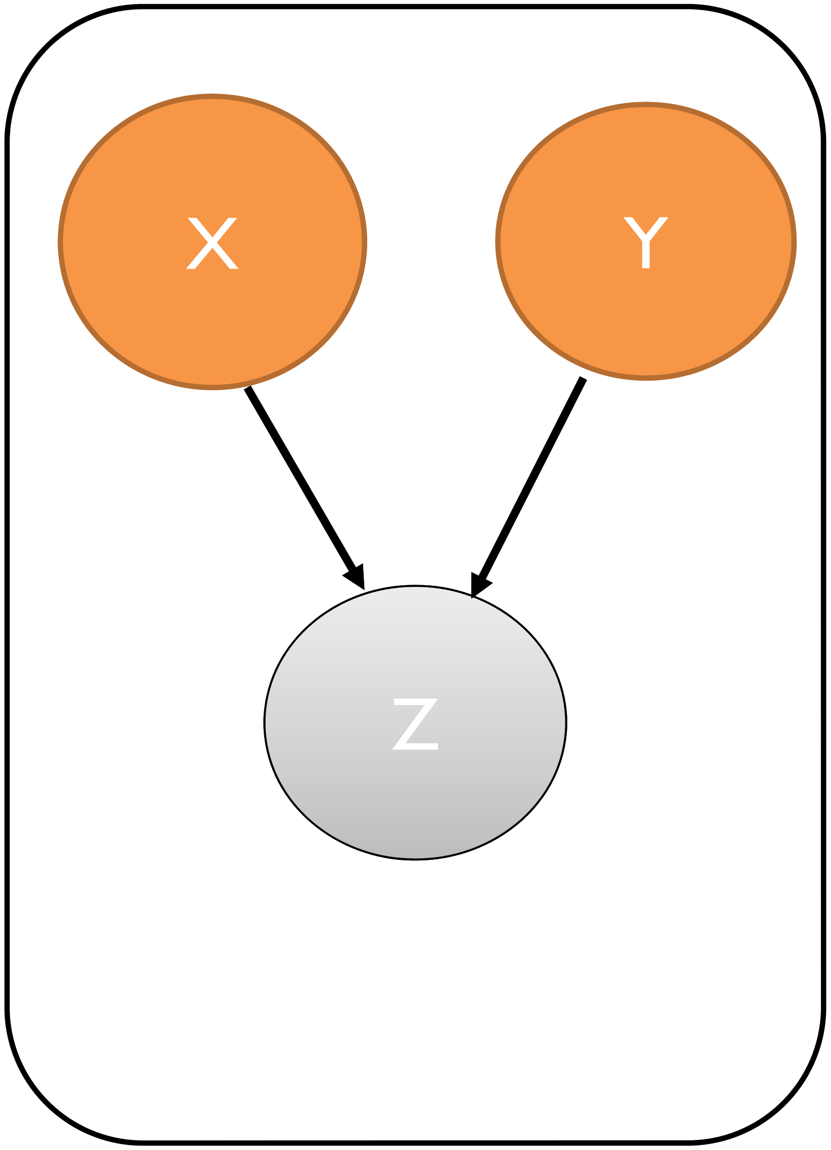
```
var test_z = function(){
 var x = flip(0.50);

 var y = flip(0.1);


 var z = x+y;


 condition(z == 1);


 return x;
}
```



Queries

Posterior distribution – what we got

Expected value – $\mathbb{E}(\boldsymbol{X}) = \sum_{\boldsymbol{x} \in \boldsymbol{Dom}(\boldsymbol{X})} \boldsymbol{x} \cdot \Pr(\boldsymbol{x})$

Most likely value – Mode of the distribution

Exact Inference

Naïve approach: Compute $P(x_1, x_2, \dots, x_n)$

Better approach:

Take advantage of (conditional) independencies

- Whenever we can expose conditional independence, e.g., $\mathrm{P}(x_1, \mathrm{x}_2|\mathrm{x}_3) = \mathrm{P}(\mathrm{x}_1|\mathrm{x}_3) \cdot \mathrm{P}(\mathrm{x}_2|\mathrm{x}_3)$ the computation is more efficient

Compute distributions from parents to children

Burglar Alarm

Your home has an **alarm**, which can be activated by a **burglar** or by an **earthquake** (or occasionally make a false alarm)

Two neighbors **Mary** and **John** can **call** if either of them hears the **alarm** and

Neighbor **John called** you to say the alarm is ringing

Question: Is there a burglar in your home?

Based on the version from "Probabilsitic Programming", Gordon, Henzinger, Nori, Rayamani (ICSE-FoSE 2014)

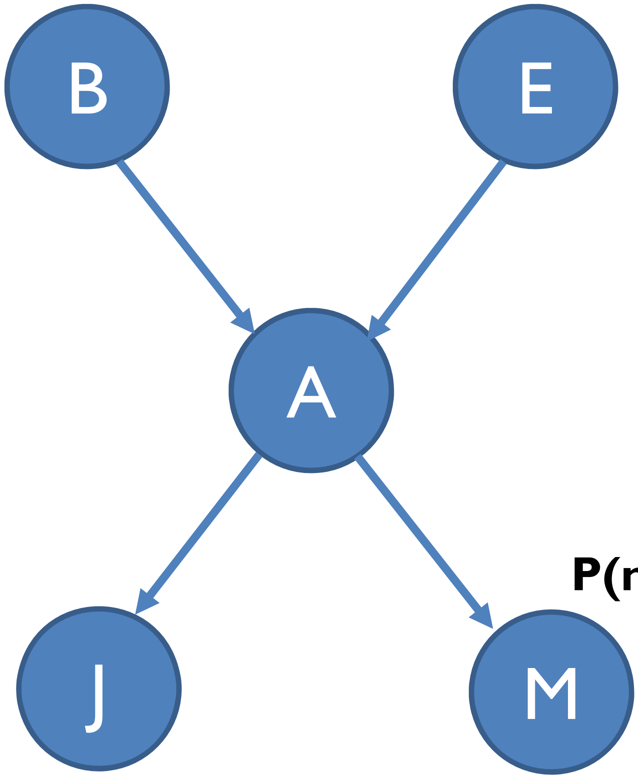
Burglar Alarm

P(b)

b^0	b^1
0.999	0.001

P(e)

e^0	e^1
0.998	0.002



P(a | b, e)

	a^0	a^1
b^0e^0	0.95	0.05
b^0e^1	0.94	0.06
b^1e^0	0.09	0.91
b^1e^1	0.001	0.990

P(j | a)

	j^0	j^1
a^0	0.9	0.1
a^1	0.05	0.95

P(m | a)

	j^0	j^1
a^0	0.7	0.3
a^1	0.01	0.99

How to come up with priors?

Statistics – results of previous studies
(e.g., probability of illness, weather
on a particular day)

Beliefs – user's subjective information
(ice cream preference 8/10)

Uninformed – any option is equally likely
(uniformly distributed priors)

Burglar Alarm

```
var burglary_alarm = function() {
  var earthquake = flip(0.002)
  var burglary = flip(0.001)

  var alarm = burglary && earthquake? flip(0.999) :
                    burglary? flip(0.91):
                    earthquake? flip(0.06) : flip(0.05)
  var johnCalls = alarm? flip(0.95) : flip(0.1)
  var maryCalls = alarm? flip(0.99): flip(0.3)

  condition (johnCalls && !maryCalls)
  return burglary
}
```

Complexity of Exact Inference

Number of variables: $\boldsymbol{n}$

Naïve enumeration: complexity is $O(2^n)$

Variable Elimination: if the maximum number of parents of the nodes is $k \in \{1, \dots, n\}$, then the complexity is $n \cdot O(2^k)$.

For many models this is a good improvement, but always possible to construct pathological models.

Beyond Bayesian Net Models

Geometric Distribution: Probability of the number of Bernoulli trials to get one success

```
var geometric = function() {
  return flip(.5) ? 0 : geometric() + 1;
}

var dist = Infer({method: 'enumerate', maxExecutions: 10},
                 geometric);
viz.auto(dist);
```

Continuous Models

TrueSkill:

- Measure player skills in various sports

Each player has an unknown parameter skill that cannot be directly measured (i.e., it is hidden)

What we can observe is how the in-game performance of the player (which depends on the skill) compares to the performance of the other player

TrueSkill Model

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

$$Skill \sim Gaussian(100,10)$$

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

$$Perf \sim Gaussian(Skill, 15)$$

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

$$PerfPlayerA > PerfPlayerB$$

TrueSkill Example

```
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 *(Exact and Approximate)*

Sampling (Rejection – Church)

Sampling (MCMC – Church & Stan & Figaro)

Variational Inference (Fun & Infer.NET)

Exact Symbolic (PSI & Hakaru)