Introduction to Probabilistic Programming

Xin Zhang

Peking University

About the Course

- Part of the international graduate program at PKU

- Offered to both domestic and international students

- Course materials in English

- We do have international students in the class so I will speak in English

- In person, use any language you like

Instructor

- Xin Zhang, Assistant Professor, Computer Science
- Research Areas: Programming Languages, Software Engineering, Assured Artificial Intelligence
- Website: http://xinpl.github.io
- E-mail: xin@pku.edu.cn
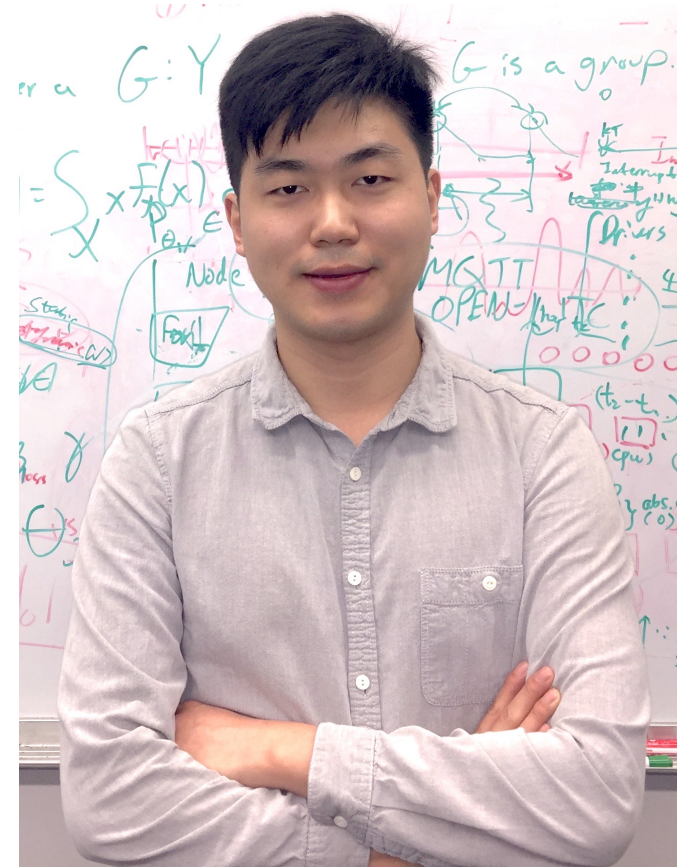
TA

- Ruyi Ji, PhD Student
- Programming Languages Lab @ PKU
- Website: https://jiry17.github.io/
- Email: jiruyi910387714@pku.edu.cn

Logistics

- Time:
  - Thursday 9am - 12am (9:00-9:50, 10:10 – 11:00, 11:10 – 12:00)

- Location:
  - Changping Campus: 211 Teaching Building（昌平校区教学楼211）
  - Main Campus: 311 Teaching Building 2 （主校区2教311）
  - Watch online at using 小鱼易连

- Credit hours: 3

- Course Website: xinpl.github.io/courses/probprog/

Please Provide Feedback

- First semester that we give lectures across campuses
  - So there will be issues
  - Please be patient and give us feedback in time!

第一次实行双校区上课，必然会遇到一些技术挑战，请保持耐心并及时提供反馈

Course WeChat Group

Course Policy

- Laptops and phones allowed to try example programs
  - Mute them
  - Don't use them to do things that are unrelated to the course

- Academic integrity
  - http://www.dean.pku.edu.cn/web/student_info.php?type=3&id=49
  - 0 tolerance for academic dishonesty and cheating

- Speak out when you don't understand something

Grading

- Assignments: 30%

- Mid-Term (In-Class Exam): 30%

- Final (In-Class Exam): 40%

What you will learn

- What probabilistic programming is
- How to write programs in popular languages
- Relevant concepts like graphical models, probabilistic inferences
- Theoretical foundations
- Inferences and learning
- Frontiers

Schedule (Tentative)

- Introduction with WebPPL (Week 1 & 2)
- Relevant backgrounds (Week 3 & 4)
- Theoretical foundations (Week 5 & 6)
- Mid-Term (Week 7)
- Inference (Week 8 & 9)
- Learning (Week 10)
- Probabilistic Logic Programming (Week 11 & 12)
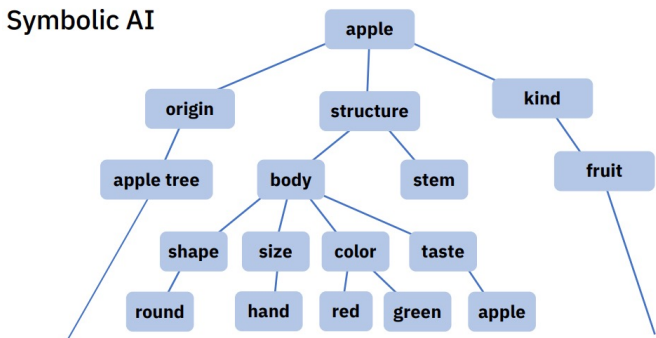- Advanced Topics (Week 13-15)
- Final (Week 16)

After This Class

- Hopefully, you can
  - Write probabilistic programs
  - Have an idea how probabilistic programming languages work underneath

How would you build an AI?
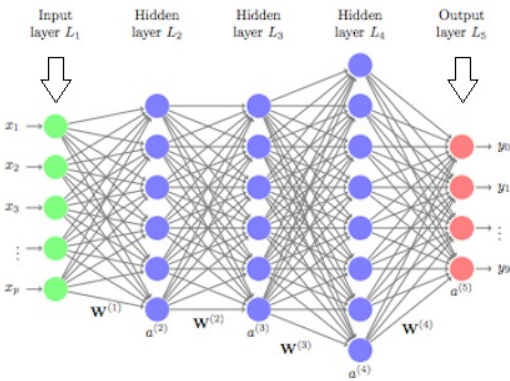# What should it be able to do?
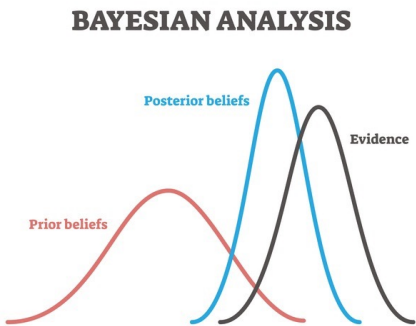
Different Styles of AI

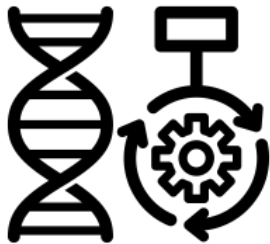## Five Tribes of Machine Learning

Symbolists   **Connectionists**   **Bayesians**   **Evolutionaries**   **Analogizers**

"The Master Algorithm", Pedro Domingos

Different Styles of AI

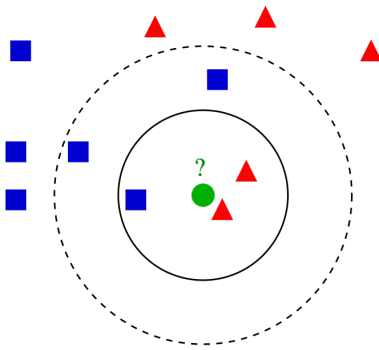## Five Tribes of Machine Learning

Symbolists   Connectionists   **Bayesians**   Evolutionaries   Analogizers

Symbolic AI

Probabilistic programming =   Programming Languages   +

BAYESIAN ANALYSIS

Posterior beliefs

Evidence

Prior beliefs
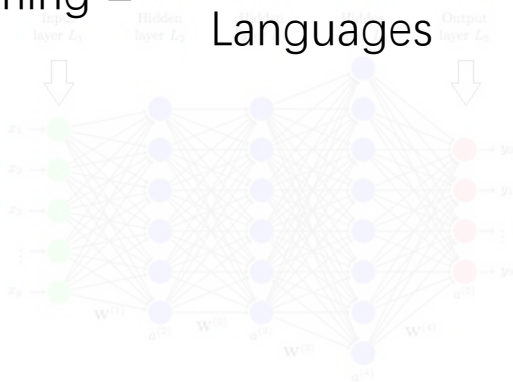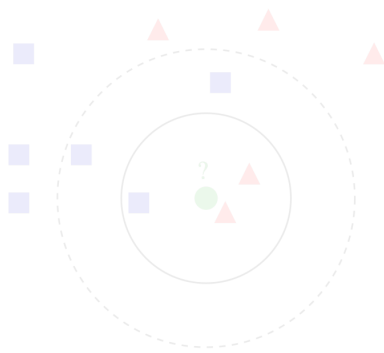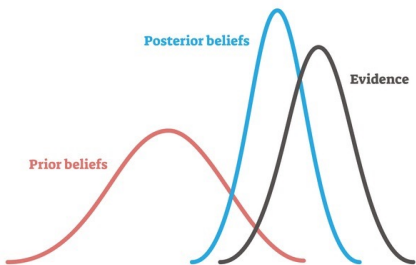
"The Master Algorithm", Pedro Domingos

Spirit of Probabilistic Programming

• Express your beliefs and uncertainties to generate data

• Adjust the model based on observed data

Bayesian

Spirit of Probabilistic Programming

- Express your beliefs and uncertainties to generate data

   If Xiaoming stays up, there is 50% chance he will drink coffee. If he
   stays up and doesn't drink coffee, he will fall asleep in class.

- Adjust the model based on observed data

   Given Xiaoming didn't fall asleep today, how likely did he stay up?

   Bayesian

Spirit of Probabilistic Programming

- Express your beliefs and uncertainties to generate data in programs

- Adjust the model based on observed data using general algorithms

Bayesian + **Programming Languages**

Old Idea that Resurged Recently

- "High level probabilistic languages have been in use since the earliest versions of FORTRAN and BASIC."- Semantics of Probabilistic Programs, Dexter Kozen, FOCS 1979

- Resurged due to
  - Novel applications
  - New inference algorithms
  - More computing power

Edward

Anglican

PYRO

Oxford, UBC, etc.
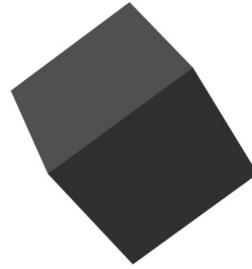
Google

Facebook

General-purpose programming languages + probabilistic constructs

Venture   **Omega**

ProbLog

MIT   MIT (I'm part of it)

Columbia, etc.

KU LEUVEN

Wikipedia lists 54 probabilistic programming languages!

List of probabilistic programming languages [edit]

This article **may contain an excessive amount of intricate detail that may interest only a particular audience**. Please help by spinning off or relocating any relevant information, and removing excessive detail that may be against Wikipedia's inclusion policy. *(October 2019)* *(Learn how and when to remove this template message)*
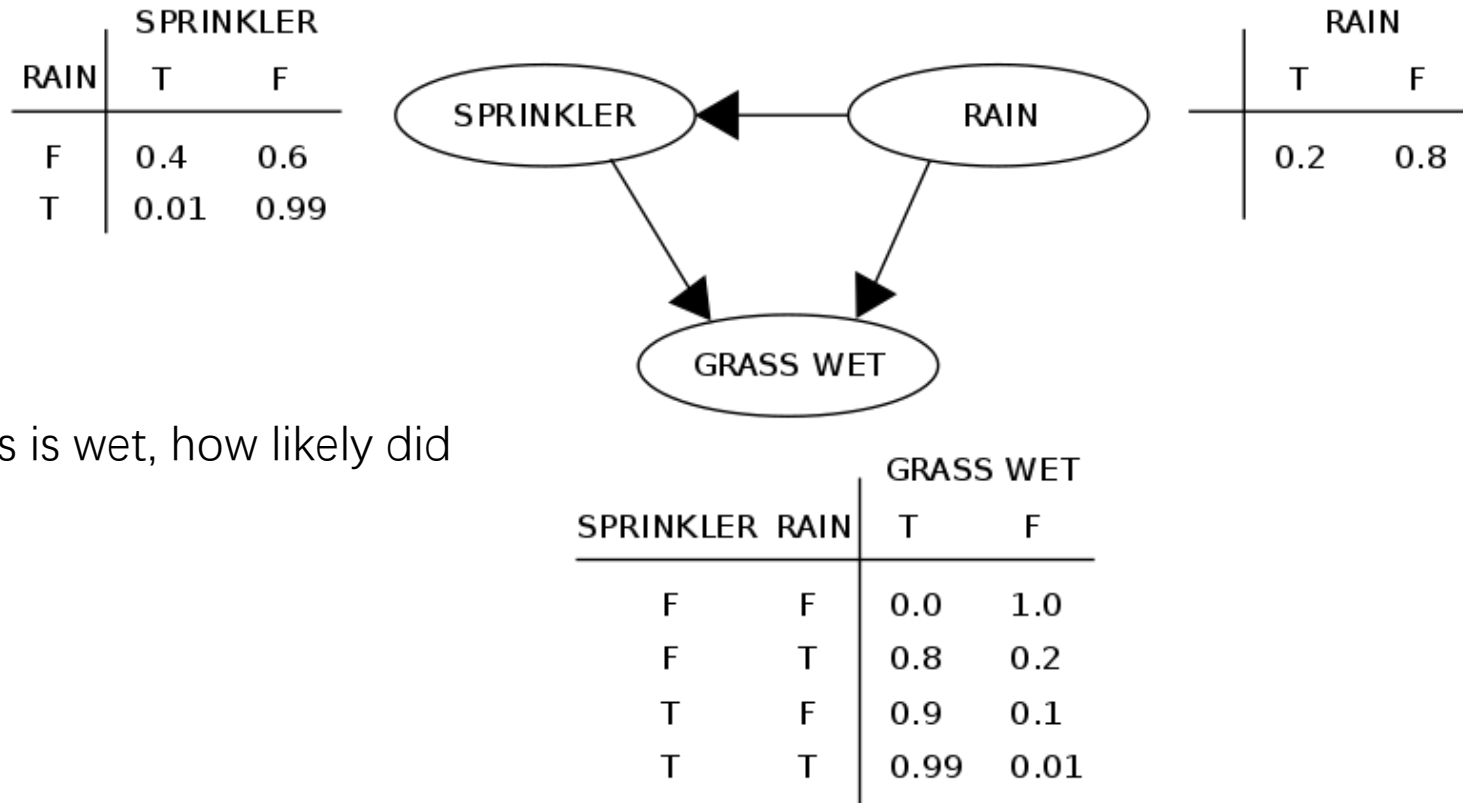
Name	Extends from	Host language
Analytica[17]		C++
bayesloop[18][19]	Python	Python
CuPPL[20]	NOVA[21]	
Venture[22]	Scheme	C++
Probabilistic-C[23]	C	C
Anglican[24]	Clojure	Clojure
IBAL[25]	OCaml	
BayesDB[26]	SQLite, Python	
PRISM[13]	B-Prolog	
Infer.NET[12]	.NET Framework	.NET Framework
dimple[10]	MATLAB, Java	
chimple[11]	MATLAB, Java	
BLOG[27]	Java	
diff-SAT[28]	Answer set programming, SAT (DIMACS CNF)	
PSQL[29]	SQL	
BUGS[14]		
FACTORIE[30]	Scala	Scala
PMTK[31]	MATLAB	MATLAB
Alchemy[32]	C++	
Dyna[33]	Prolog	
Figaro[34]	Scala	Scala
Church[35]	Scheme	Various: JavaScript, Scheme
ProbLog[36]	Prolog	Python, Jython
ProBT[37]	C++, Python	
Stan[15]		C++
Hakaru[38]	Haskell	Haskell
BAli-Phy (software)[39]	Haskell	C++

Probabilistic Programming

- Introduces new machine learning models

- Introduces new programming models

Old-School Bayesian Models

RAIN	SPRINKLER T	F
F	0.4	0.6
T	0.01	0.99

RAIN T	F
0.2	0.8

Given the grass is wet, how likely did it rain?

SPRINKLER	RAIN	GRASS WET T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

The picture is from Wikipedia.

Probabilistic Programming Models
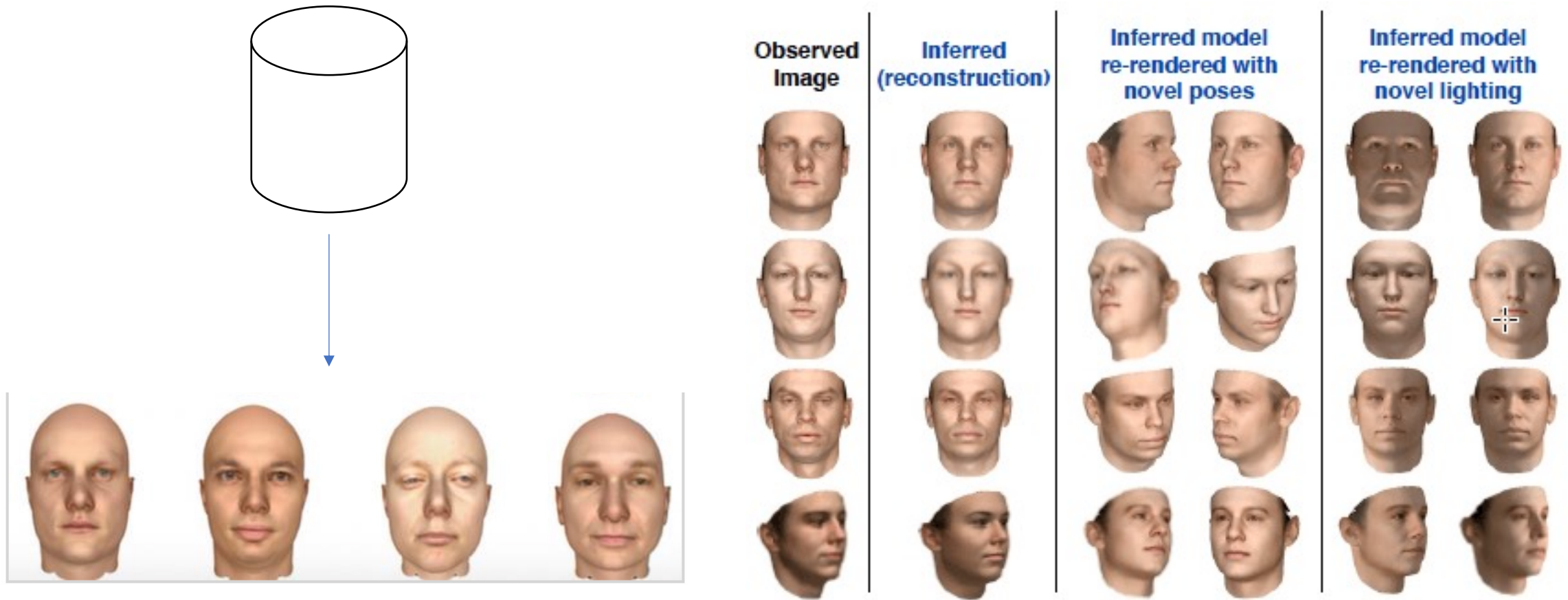
Inverse Graphics – 3D faces rendered from 2D images using only 50 lines of PPL code.
Reference: http://news.mit.edu/2015/better-probabilistic-programming-0413

Old-School Languages vs. PPL

f

→ Value

f_p

→ Distribution

Built-in support for random
variables and operations on them
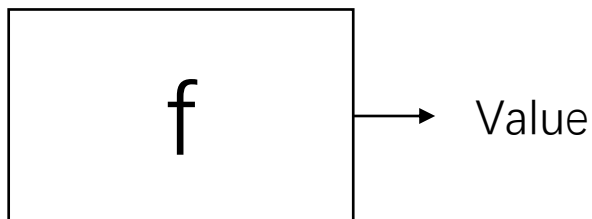
The semantics of the program is
defined on a given execution.

The semantics of the program is
defined on a distribution of executions.

First Probabilistic Program in WebPPL

- There is a gatcha game:
  - 10% chance to get a SSR card every pull.
  - A pull costs 10 RMB.
  - Xiaoming usually pulls 0-9 times a day.

- How many SSRs can he get a day?

- We can model it using a probabilistic program in WebPPL!

Subset of Javascript + Probabilistic Constructs

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- Xiaoming usually pulls 0-9 times a day.

You can install WebPPL locally or directly try it at http://webppl.org!

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c){
      if( c == 0){
        return 0;
      }
      return sample(pull)+performPull(c-1);
    }
    return performPull(num_pull_inst)
};

gatcha()
```

Built-in distributions/ random variables

Let's see it!

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- Xiaoming usually pulls 0-9 times a day.

- What if I want to know the average number of SSR Xiaoming gets?
- What if I want to know what is the chance that Xiaoming gets two SSRs?
- What if I want to know the full distribution?

Well, you can do it manually, but WebPPL has built-in support for these!

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c){
      if( c == 0){
        return 0;
      }
      return sample(pull)+performPull(c-1);
    }
    return performPull(num_pull_inst)
};

gatcha()
```

Built-in distributions/ random variables

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- Xiaoming usually pulls 0-9 times a day.

  - What if I want to know the average number of SSR Xiaoming gets?
  - What if I want to know what is the chance that Xiaoming gets two SSRs?
  - What if I want to know thefull distribution?

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c){
      if( c == 0){
        return 0;
      }
      return sample(pull)+performPull(c-1);
    }
    return performPull(num_pull_inst)
};

gatcha()

var gatcha_model = Infer({model: gatcha})

display(expectation(gatcha_model))
display(Math.exp(gatcha_model.score(3)))
viz(gatcha_model)
```

Built-in distributions/ random variables

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- Xiaoming usually pulls 0-9 times a day.

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c){
      if( c == 0){
        return 0;
      }
      return sample(pull)+performPull(c-1);
    }
    return performPull(num_pull_inst)
};
```

Built-in distributions/ random variables

"Black magic" behind this function invocation. It performs **Marginal Inference**, which creates a map from values to probabilities.

```
var gatcha_model = Infer({model: gatcha})

display(expectation(gatcha_model))
display(Math.exp(gatcha_model.score(3)))
viz(gatcha_model)
```

More on Marginal Probabilities

- We can enumerate all possible worlds and calculate their probabilities.

- Marginal probability of x = X is defined as

$$P(x = X) = \Sigma_{w \in \{w | x = X \ in \ W\}} P(w)$$

We will talk about how to calculate them later in the course.

Let's make the problem more interesting

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- **If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.**
- Xiaoming usually pulls 0-9 times a day.

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c){
      if( c == 0){
        return 0;
      }
      return sample(pull)+performPull(c-1);
    }
    return performPull(num_pull_inst)
};

gatcha()

var gatcha_model = Infer({model: gatcha})

display(expectation(gatcha_model))
display(Math.exp(gatcha_model.score(3)))
viz(gatcha_model)
```

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- **If 5 continuous pulls yield 0 SSR, the 6ᵗʰ pull guarantees an SSR.**
- Xiaoming usually pulls 0-9 times a day.

  - What if I want to know the average number of SSR Xiaoming gets?
  - What if I want to know what is the chance that Xiaoming gets three SSRs?
  - What if I want to know the full distribution?

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c, num_no_ssr){
      if( c == 0){
        return 0;
      }
      if (num_no_ssr == 5){
          return 1 + performPull(c-1, 0);
      }
      else{
        var cp = sample(pull);
        if (cp)
          return 1 + performPull(c-1, 0);
        else
          return 0 + performPull(c-1, num_no_ssr+1);
      }
    }
    return performPull(num_pull_inst, 0)
};
```

Informal Semantics of Probabilistic Programs

- Probabilistic programs define a distribution of executions
  - The randomness comes from random variables
  - Given an execution e, let V be the set of random variables that are evaluated in e, we use $v(e)$ to denote the value a random variable v takes in e, then we have

$$P(e) = \prod_{v \in V} P(v = v(e))$$

  - We will introduce formal semantics later in the course!

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- **If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.**
- Xiaoming usually pulls 0-9 times a day.

Hey! What you assume seems quite reasonable. But Xiaoming never gets more than two SSRs a day!

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- Xiaoming usually pulls 0-9 times a day.
- **Xiaoming never gets more than 2 SSRs a day.**

This describes the outcome rather than the process. How should we modify the program?

New Language Construct!

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,10);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c, num_no_ssr){
      if( c == 0){
        return 0;
      }
      if (num_no_ssr == 5){
        return 1 + performPull(c-1, 0);
      }
      else{
        var cp = sample(pull);
        if (cp)
          return 1 + performPull(c-1, 0);
        else
          return 0 + performPull(c-1, num_no_ssr+1);
      }
    }
    return performPull(num_pull_inst)
};
```

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- Xiaoming usually pulls 0-9 times a day.
- **Xiaoming never gets more than 2 SSRs a day.**

This describes the outcome rather than the process. How should we modify the program?

New Language Construct!

```
var gatcha = function(){
  ...
    return performPull(num_pull_inst)
};

var gatcha1 = function(){
  var num_ssrs = gatcha()
  condition(num_ssrs <= 2)
  return num_ssrs
}

var gatcha_model = Infer({model: gatcha1})

display(expectation(gatcha_model))

display(Math.exp(gatcha_model.score(3)))

viz(gatcha_model)
```

Filter out executions that do not satisfy external knowledge, and computes a conditional distribution

More on Condition

var d = sample(**Categorical**({ps: [0.2, 0.3, 0.5], vs: [1,2,3]}))

condition(d != 3)

Value of d	Probability
1	0.2
2	0.3
3	0.5

More on Condition

var d = sample(**Categorical**({ps: [0.2, 0.3, 0.5], vs: [1,2,3]}))

condition(d != 3)

Value of d	Probability
1	0.2
2	0.3
3	0.5

More on Condition

var d = sample(**Categorical**({ps: [0.2, 0.3, 0.5], vs: [1,2,3]}))

condition(d != 3)

Value of d	Probability
1	0.2 / (0.2+0.3)
2	0.3 / (0.2+.3)
3	0.5

More on Condition

var d = sample(**Categorical**({ps: [0.2, 0.3, 0.5], vs: [1,2,3]}))

condition(d != 3)

Value of d	Probability
1	0.4
2	0.6
3	0.5

More on Condition

- Recall our informal semantics

$$P(e) = \prod_{v \in V} P(v = v(e))$$

- We use $P(e|c)$ to denote the probability of an execution e after a condition statement c is added to the program, and we use $c(e)$ to represent whether c holds on e, then

$$P(e|c) = \begin{cases} 0, & if\ c(e) = false \\ \dfrac{P(e)}{\sum_{e' \in \{e''|c(e'')\}} P(e')}, & if\ c(e) = true \end{cases}$$

More on Condition

- It might be obvious, but condition does construct a conditional distribution:

$$P(e|c) = \frac{P(e \cap c)}{P(c)} = \frac{P(e \cap c)}{\sum_{e' \in E} P(c \cap e')} = \frac{P(c|e) * P(e)}{\sum_{e' \in E} P(c|e') * P(e')} = \begin{cases} 0, & \text{if } c(e) = false \\ \dfrac{P(e)}{\sum_{e' \in \{e''|c(e'')\}} P(e')}, & \text{if } c(e) = true \end{cases}$$

Main Probabilistic Constructs in WebPPL

- Built-in random variables and sample together describe a random process in a constructive way

- Condition provides a way to incorporate external knowledge about the output

They together provide a mechanism to do Bayesian learning

What you believe + What you observe

A Simple Bayesian Inference Example

- A disease can happen to 1% of the population
- A test method has the following property:

	Tested Positive	Tested Negative
Actually Positive	90%	10%
Actually Negative	10%	90%

- Xiaoming is tested positive. How likely does he carry the disease?

A Simple Bayesian Inference Example

- We can calculate the probability using Bayes' theorem

$$P(positive \,|tested\ positive) = P(positive) \times \frac{P(tested\ positive|positive)}{P(tested\ positive)}$$

$$= 0.01 \times \frac{0.9}{0.01 * 0.9 + 0.99 * 0.1}$$

$$= 0.083$$

A Simple Bayesian Inference Example

- The probabilistic program is as follow:

```
var medic_test = function(){
  var positive = sample(Bernoulli({p:0.01}))
  var test_positive = function(pg){
    if(pg){
      return sample(Bernoulli({p:0.9}))
    }
    else
      return !sample(Bernoulli({p:0.9}))
  }
  condition(test_positive(positive))
  return positive
}

var m = Infer({model:medic_test})
display(Math.exp(m.score(true)))
viz(m)
```

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- **20% of players believe the rate is not as advertised, but only 8%.**
- **To test if the assumption is true, Xiaoming pulled 20 times, and got 4 SSRs.**

What is the chance that the rate is actually 8%?

Try to calculate it using Bayes' theorem!

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- **20% of players believe the rate is not as advertised, but only 8%.**
- **To test if the assumption is true, Xiaoming pulled 20 times, and got 4 SSRs.**

In WebPPL, it is easy!

```
var gatcha = function(){
    var cheated = sample(Bernoulli({p:0.2}));
    var pull = function(){
      if (cheated){
        return Bernoulli({p:0.08});
      }
      else
        return Bernoulli({p:0.1});
    }
    var num_pull_inst = 20;

    var performPull = function(c, num_no_ssr){
       ...
    }
    var num_ssrs = performPull(num_pull_inst, 0)
    condition(num_ssrs == 4)
    return cheated;
};

var gatcha_model = Infer({model: gatcha})

display(expectation(gatcha_model))
```

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- **When spending over 1000 RMB, for every new pull, there is a chance that the bank would call Xiaoming.**
- **Suppose the chance is (x-1000)/1000, and Xiaoming got called and then stopped, how many SSRs has he pulled so far?**

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- **When spending over 1000 RMB, for every new pull, there is a chance that the bank would call Xiaoming.**
- **Suppose the chance is (x-1000)/1000, and Xiaoming got called and then stopped, how many SSRs has he pulled so far?**

Can you express this problem using a conventional graphical model like a Bayesian network?

```
var gatcha = function(){
    ...
    var VS = rangeArray(0,1000);
    var num_pull = Categorical({vs: VS});
    ...
    var performPull = function(c, num_no_ssr){
      if( c == 0){
        return [0,false];
      }
      var cost = (num_pull_inst -c )*10;
      if(cost > 2000)
        return [0, true]
      if(cost > 1000)
        if(sample(Bernoulli({p: (cost - 1000)/1000.0}))){
          return [0, true]
        }
        ...
    }
    var pull_result = performPull(num_pull_inst, 0)
    condition(pull_result[1])
    return pull_result[0]
};
```

Probabilistic Programming So Far

- Built-in support for random variables
  - **Categorical**, **Bernoulli** ….
  - Sample

- A general language to describe the sampling process
  - Subet of Javascript

- The ability to impose conditions on any state
  - Condition

Probabilistic Programming So Far

- Built-in support for random variables
  - **Categorical**, **Bernoulli** ….
  - Sample

- A general language to describe the sampling process
  - Subet of Javascript

- The ability to impose conditions on any state
  - Condition

A convenient way to express highly complex distributions

Don't worry about how to calculate it! Just think about what is it!

```
var gatcha = function(){
    var pull = Bernoulli({p:0.1});
    var VS = rangeArray(0,1000);
    var num_pull = Categorical({vs: VS});
    var num_pull_inst = sample(num_pull);
    var performPull = function(c, num_no_ssr){
      if( c == 0){
        return 0;
      }
      if (num_no_ssr == 5){
        return 1 + performPull(c-1, 0);
      }
      else{
        var cp = sample(pull);
        if (cp)
          return 1 + performPull(c-1, 0);
        else
          return 0 + performPull(c-1, num_no_ssr+1);
      }
    }
    return performPull(num_pull_inst)
};
}
```

```
var gatcha1 = function(){
  var num_ssrs = gatcha()
  condition(num_ssrs <= 2)
  return num_ssrs
}

var gatcha_model = Infer({model: gatcha1})

display(expectation(gatcha_model))

viz(gatcha_model)
```

- 10% chance to get a SSR card every pull.
- A pull costs 10 RMB.
- If 5 continuous pulls yield 0 SSR, the 6th pull guarantees an SSR.
- Xiaoming usually pulls 0-999 times a day.
- **Xiaoming never gets more than 2 SSRs a day.**

Next Class

- More about WebPPL
  - We have talked about the core probabilistic constructs

- Representative applications using WebPPL