Advanced Probability and Statistics

Xin Zhang

Peking University

What the course was about

- An introductory course to an advanced topic
  - Still an advanced course

- The first course on probabilistic programming in China

- Offered to both domestic and international students
  - Part of the international graduate program at PKU

- Course materials in English

Schedule (Updating)

🔴 Not specific to probabilistic programming

	Date	Slides	Extended Reading
	Feb 18	Introduction	The Five Tribes of Machine Learning with Pedro Domingos
	Feb 25	Probabilistic Programming Using WebPPL	Introduction to WebPPL
🔴	Mar 3	Graphical Models 1	Chapter 8 of Pattern Recognition and Machine Learning
🔴	Mar 11	Graphical Models 2	Chapter 8 of Pattern Recognition and Machine Learning
🔴	Mar 18	Approximate Inference	Chapter 10&11 of Pattern Recognition and Machine Learning
	Mar 25	Semantics of Probabilistic Programming	Semantics of Probabilistic Programming: A Gentle Introduction
	Apri 1	Midterm	
	April 7	Inference in Probabilistic Programming I	Chapter 3 of An Introduction to Probabilistic Programming
	April 15	Inference in Probabilistic Programming II	Chapter 4 of An Introduction to Probabilistic Programming
	April 22	Probabilistic Logic Programming	Unifying Logic and Probability
🔴	April 29	Causal Inference	Chapter 1 of "Causality" by Judea Pearl (access via PKU library service)
🔴	May 13	Explainable AI	AAAI 2021 Tutorial
	May 20	Probabilistic Programming with Deep Learning	Understanding the World Through Code
🔴	May 27	LLMs with Language Control	LMQL
	June 3	Course Review	

What the course is about

- The school wants to change the course into a core course
  - "Introduction to Probabilistic Programming" doesn't sound a core course

- My thoughts about the course:
  - It is a computer science course. Let's not make it too mathematical.
  - Probabilistic programming is still very important.
  - I will include more lightweight introduction to probability theories (concentration inequalities, measure theory)
  - Course material in English, lecture in Chinese

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
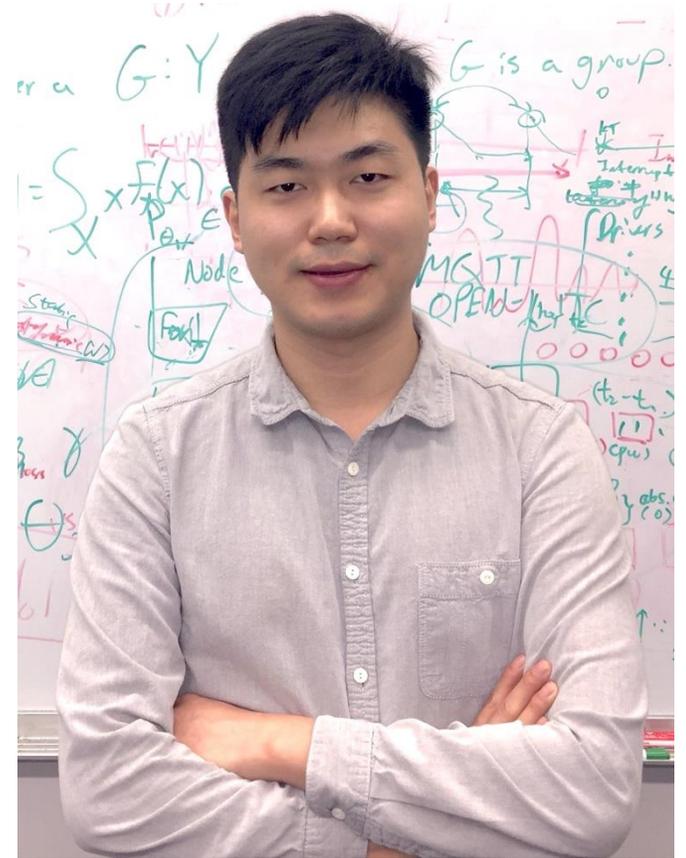
TA (Tentative)

- Haoran Lin, PhD Student
- Programming Languages Lab @ PKU
- Email: haoranlin@stu.pku.edu.cn,

Logistics

- Time:
  - Wednesday 15:10pm – 18:00pm (15:10-16:00, 16:10 – 17:00, 17:10 – 18:00)

- Location:
  - Changping Campus: 109 Teaching Building（新燕园109）

- Credit hours: 3

- Course Website: http://xinpl.github.io/courses/probstat/2026/

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 (Exam): 40%

What you will learn

• Bayesian learning

• What probabilistic programming is

• Graphical models

• Probabilistic inferences

• Semantics of probabilistic programming and lightweight measure theory

• Frontiers

Schedule (Tentative)

- Introduction to probabilistic programming with WebPPL (Week 1&2)
- Graphical Models (Week 3 & 4)
- Approximate Inference and its Theory I (Week 5)
- Approximate Inference and its Theory II (Week 6)
- Inference in probabilistic programming I (Week 7)
- Mid-Term (Week 8)
- Inference in probabilistic programming II (Week 9)
- Semantics of probabilistic programming with measure theory (Week 10)
- Probabilistic Logic Programming (Week 11)
- Advanced Topics (Week 12-15)

After This Class

- Hopefully, you can
  - Know what graphical models are
  - Write probabilistic programs
  - Have an idea about common probabilistic inference algorithms

How would you build an AI?
What should it be able to do?

Different Styles of AI
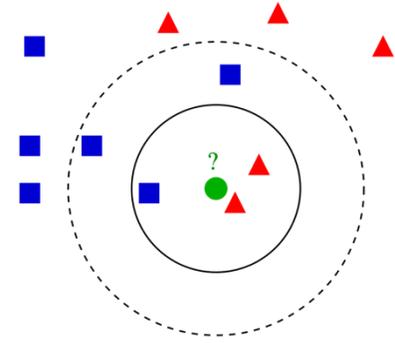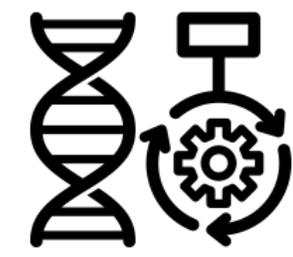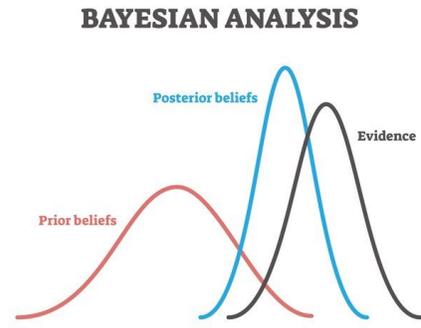
Five Tribes of Machine Learning

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

"The Master Algorithm", Pedro Domingos
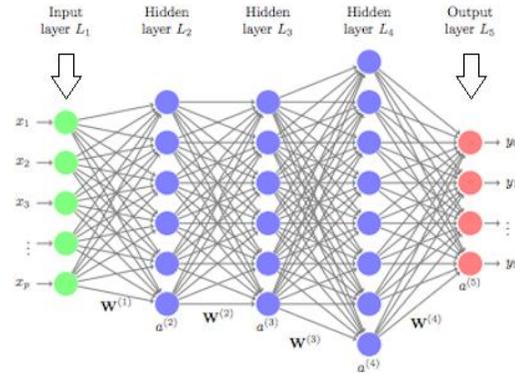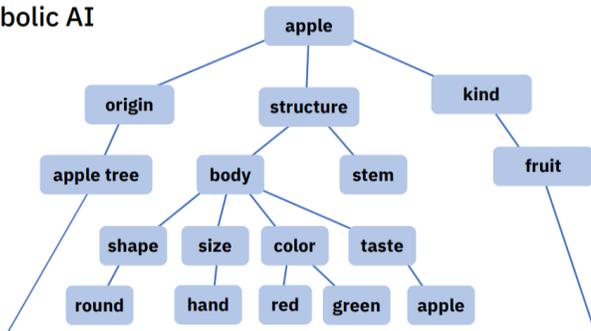
Different Styles of AI

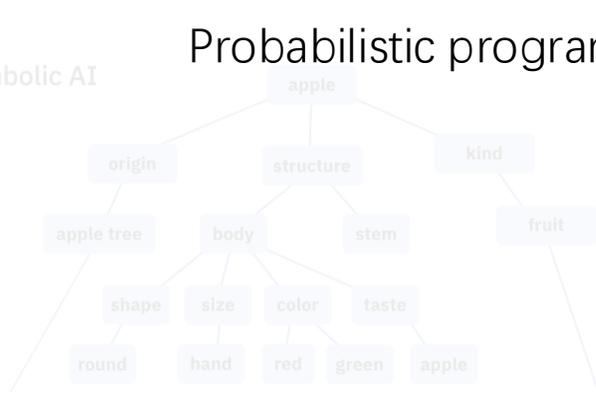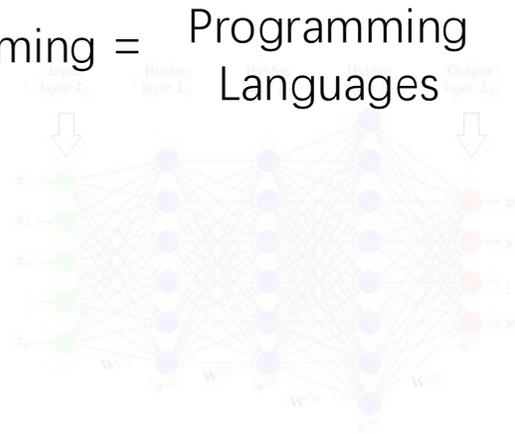## Five Tribes of Machine Learning

Symbolists　　　Connectionists　　　**Bayesians**　　　Evolutionaries　　Analogizers

Symbolic AI

Probabilistic programming = Programming Languages　+

BAYESIAN ANALYSIS

Posterior beliefs

Evidence

Prior beliefs
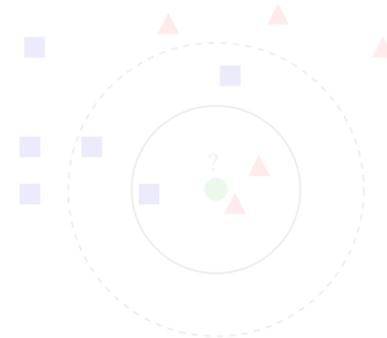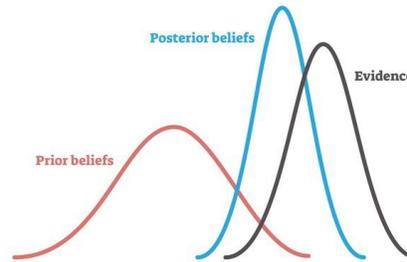
"The Master Algorithm", Pedro Domingos

Doesn't LLM kill everything?

- Well, you can ask ChatGPT/Deepseek itself if probabilistic programming is still relevant

Yes, probabilistic programming remains very useful even with the rise of LLMs. While LLMs excel at capturing patterns in large datasets and generating fluent text, probabilistic programming offers several complementary advantages:

- **Explicit Uncertainty Modeling:** Probabilistic programming languages let you model uncertainty directly, which is crucial for applications that need reliable uncertainty quantification and decision-making under uncertainty.

- **Interpretable Models:** These frameworks encourage the development of models that are transparent and interpretable, making it easier to understand the underlying assumptions and reasoning.

- **Integration of Domain Knowledge:** They allow for the incorporation of prior knowledge and domain-specific constraints, which can be particularly valuable when data is scarce or when you need to enforce known relationships.

- **Hybrid Approaches:** There's a growing interest in combining probabilistic programming with neural models (including LLMs) to leverage the strengths of both paradigms—capturing complex patterns from data while maintaining robust, interpretable uncertainty estimation.

In summary, while LLMs are powerful tools for many tasks, probabilistic programming remains an essential approach for scenarios where understanding and managing uncertainty is key.

概率编程在大模型的时代下还有用吗?

- **金融风控**：大模型分析非结构化数据（如财报文本），概率模型评估违约概率和风险敞口。
- **医疗诊断**：大模型从医学文献中提取知识，贝叶斯网络整合患者数据并提供诊断置信度。

---

5. 未来趋势

- **技术融合**：概率编程语言（如Stan、PyMC）与深度学习框架（PyTorch、JAX）的深度集成（如 NumPyro、Pyro）。
- **自动化概率建模**：结合大模型的代码生成能力，自动构建和优化概率模型。
- **边缘计算**：轻量化概率模型与大模型协作，在资源受限场景（如物联网设备）中实现实时推理。

结论

在大模型时代，概率编程不仅没有过时，反而因其在**不确定性建模**、**可解释性**、**小数据学习**等方面的不可替代性，成为构建可靠AI系统的关键组件。两者的结合将推动更鲁棒、透明且适应复杂场景的智能系统发展。

Doesn't LLM kill everything?

- They can be combined
  - PP can be used to program LLMs (https://arxiv.org/pdf/2306.03081.pdf)

  - They can serve as different parts of an AI (https://arxiv.org/pdf/2306.12672.pdf)
    - LLM can understand the context well, while PP can do reasoning

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

Oxford, UBC, etc.

Google

PYRO
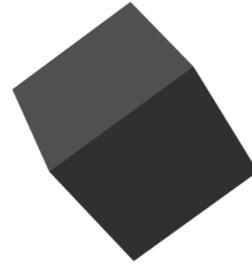
Facebook

General-purpose programming languages + probabilistic constructs

Venture

Omega

ProbLog

MIT

MIT (I was part of it)

Columbia, etc.

KU LEUVEN

Wikipedia lists 55 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
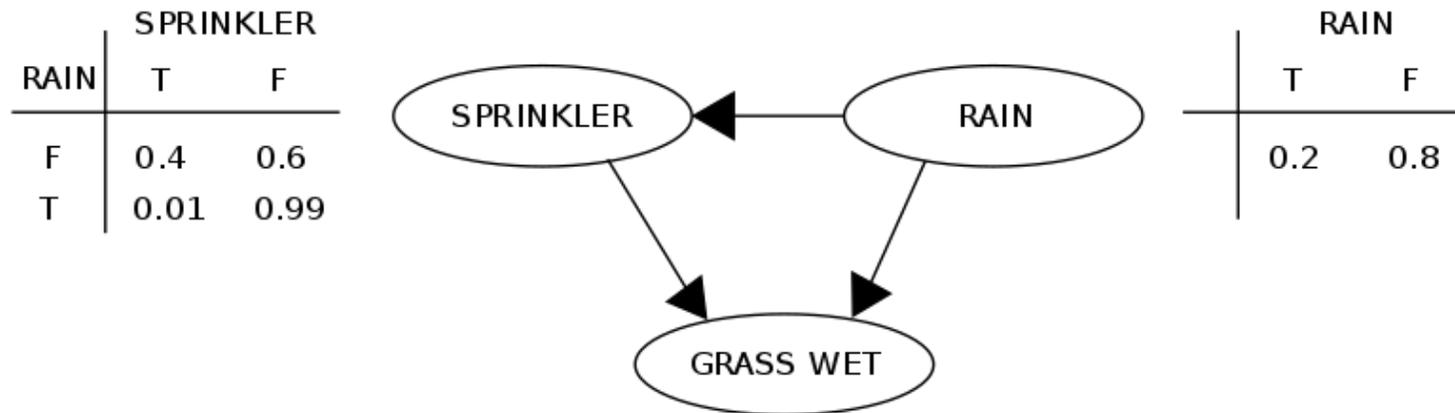
	RAIN	
	T	F
	0.2	0.8

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
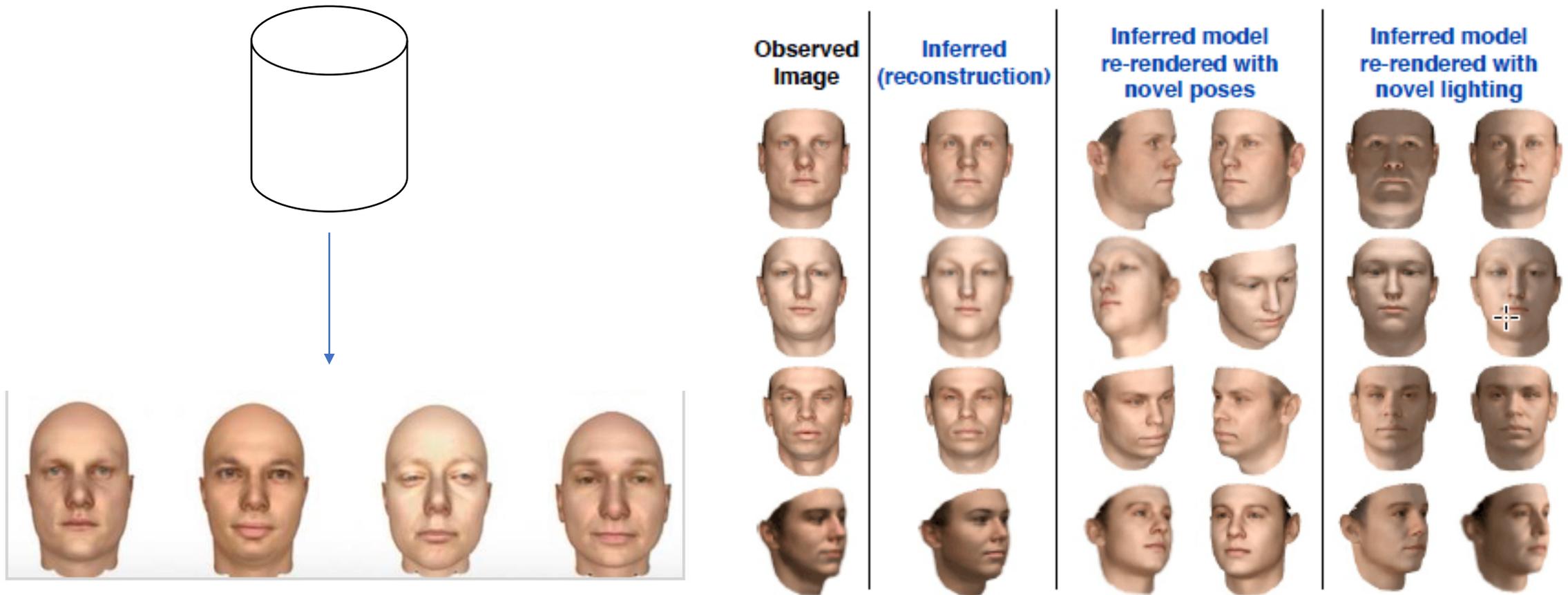
Inverse Graphics – 3D faces rendered from 2D images using only 50 lines of PPL code.
Reference: http://news.mit.edu/2015/better-probabilistic-programming-0413
Another example: https://www.youtube.com/watch?v=126X_P_y6hE

Old-School Languages vs. PPL

f → Value

f_p → Distribution

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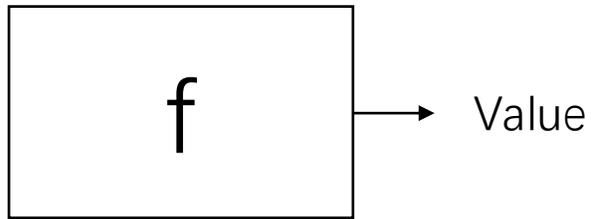
The semantics of the program is
defined on a given execution.

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

First Probabilistic Program in WebPPL

- There is a gacha 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 gacha = 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)
};

gacha()
```

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 gacha = 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)
};

gacha()
```

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 the full distribution?

```
var gacha = 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)
};

gacha()

var gacha_model = Infer({model: gacha})

display(expectation(gacha_model))
display(Math.exp(gacha_model.score(2)))
viz(gacha_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.

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

```
var gacha = 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

```
var gacha_model = Infer({model: gacha})

display(expectation(gacha_model))
display(Math.exp(gacha_model.score(2)))
viz(gacha_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 gacha = 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)
};

gacha()

var gacha_model = Infer({model: gacha})

display(expectation(gacha_model))
display(Math.exp(gacha_model.score(3)))
viz(gacha_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 gacha = 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 gacha = 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 gacha = function(){
    ...
    return performPull(num_pull_inst)
};

var gacha1 = function(){
  var num_ssrs = gacha()
  condition(num_ssrs <= 2)
  return num_ssrs
}

var gacha_model = Infer({model: gacha1})

display(expectation(gacha_model))

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

viz(gacha_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:

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

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

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

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

$$= 0.083$$

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 gacha = 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 gacha_model = Infer({model: gacha})

display(expectation(gacha_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 gacha = 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!

That sounds too good to be true.
There must be a catch here ...

Well, some of the probabilistic programs can be really slow to run.

People have been working on how to make probabilistic programs run fast, which we will discuss later in the course.

```
var gacha = 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 gacha1 = function(){
    var num_ssrs = gacha()
    condition(num_ssrs <= 2)
    return num_ssrs
}

var gacha_model = Infer({model: gacha1})

display(expectation(gacha_model))

viz(gacha_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