Lecture 3: Bayesian Modeling (Part 2)

Advanced Topics in Machine Learning

Dr. Tom Rainforth

January 24nd, 2020

rainforth@stats.ox.ac.uk

Lecture 3 Outline

This lecture will focus on methods for constructing Bayesians models

In particular we will cover the following topics:

- Independence
- Graphical models
- Probabilistic programming

Independence

Independence

- Events are **independent** if the occurrence of one event does not affect the probability of the other
  - Equivalent definition for random variables
  - If X and Y are independent, we denote this as $X \perp\!\!\!\perp Y$
  - This is sometimes also known as **marginal independence**
- If $X \perp\!\!\!\perp Y$, then $P(X = x) = P(X = x|Y = y)$ for all x, y.
  - Independence is symmetric:

$$X \perp\!\!\!\perp Y \quad \Rightarrow \quad P(Y = y) = P(Y = y|X = x)$$

  - Through the product rule, we further have: if

$$X \perp\!\!\!\perp Y \quad \Rightarrow \quad P(X, Y) = P(X)P(Y)$$

Examples of Independent Random Variables

- The outcomes of two flipped coins
- The speed of a car and whether a speed camera is malfunctioning
- We often **assume** independence between variables when constructing a model
  - Don't forgot Bayesian probabilities are subjective beliefs

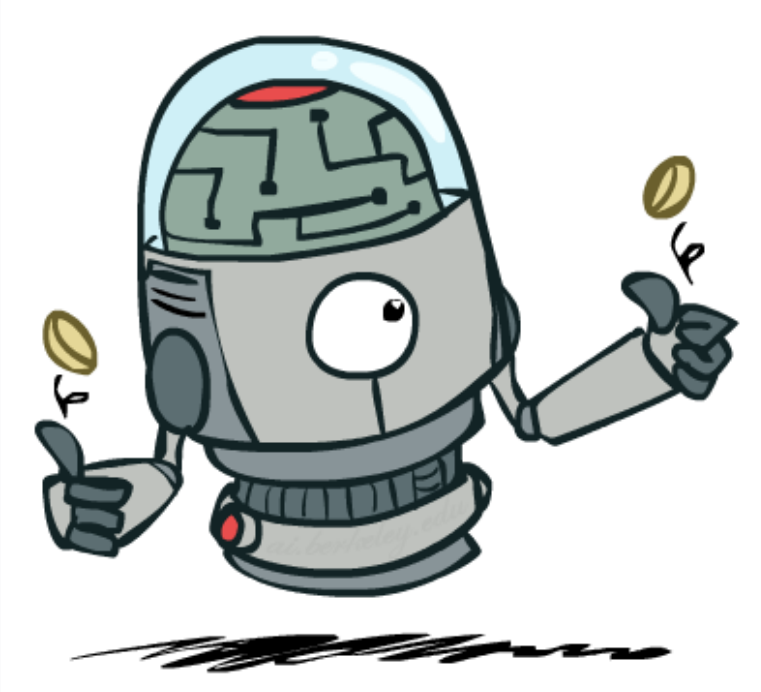


Image Credit: Pieter Abbeel

Conditional Independence

- Two events A and B are **conditionally independent** given event C if A and B are independent given that C occurred.
  - Again equivalent definition for random variables
  - If X and Y are conditionally independent given Z, we denote this as $X \perp\!\!\!\perp Y|Z$
- If $X \perp\!\!\!\perp Y|Z$, then $P(X = x|Z = z) = P(X = x|Y = y, Z = z)$ for all possible x, y, and z.
  - This is again symmetric in X and Y, but not Z

$$X \perp\!\!\!\perp Y|Z \quad \Rightarrow \quad P(Y = y|Z = z) = P(Y = y|X = x, Z = z)$$

$$X \perp\!\!\!\perp Y|Z \quad \not\Rightarrow \quad X \perp\!\!\!\perp Z|Y \quad \text{or} \quad Z \perp\!\!\!\perp Y|X$$

- Conditional independence does not imply marginal independence or vice versa

$$X \perp\!\!\!\perp Y|Z \not\Rightarrow X \perp\!\!\!\perp Y, \qquad X \perp\!\!\!\perp Y \not\Rightarrow X \perp\!\!\!\perp Y|Z$$

Conditional Independence Examples

- The amount of traffic and the number of umbrellas being used are not independent
- But they are conditionally independent given it is raining
- A speed camera being switched on and a car speeding are marginally independent
- But they are not conditionally independent given the camera is triggered

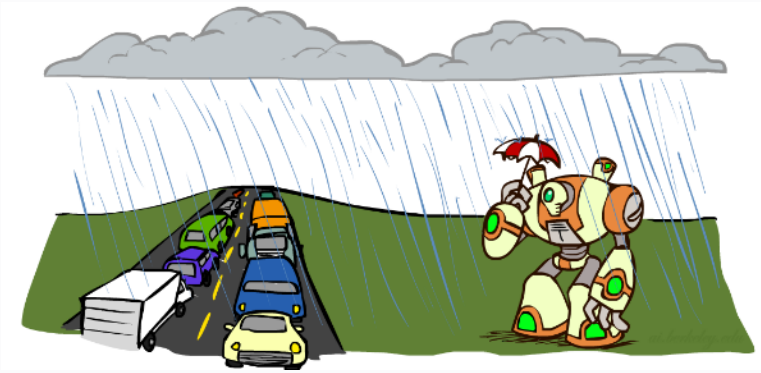


Image Credit: Pieter Abbeel

Image Credit: Autocar

Why are Independence Relationships Important?

- (Conditional) Independence between variables is one of the most important **modeling assumptions** we can make when constructing Bayesian models
- The **chain rule** is an extension of the product rule that allows us to break down any joint distribution into a product over conditional probability distributions:

$$\begin{aligned} P(X_{1:N}) &= P(X_1)P(X_2|X_1)\dots P(X_N|X_1, X_2, \dots, X_{N-1}) \\ &= P(X_1)\prod_{n=2}^{N} P(X_n|X_{1:n-1}) \end{aligned} \tag{1}$$

- Note we can use any ordering of the variables we want and the chain rule also applies densities, e.g.

$$p(x_{1:N}) = p(x_N)\prod_{n=1}^{N-1} p(x_n|x_{n+1:N})$$

Why are Independence Relationships Important? (2)

- Different orderings of the variables in the joint breakdown are known as **factorizations** of the joint
- Independence assumptions allow us to make simplifications that show up in the factorizations
- For example, the **Markov property**

$$X_n \perp\!\!\!\perp X_{1:N} \backslash X_n \mid X_{n-1}, X_{n+1} \tag{2}$$

allows us to factorize the joint as

$$P(X_{1:N}) = P(X_1) \prod_{n=2}^{N} p(X_n | X_{n-1}) \tag{3}$$

Example: Flipping Lots of Coins

- We flip N independent coins
- The joint space of possible outcomes is 2^N which becomes impractical to deal with for even moderate N
- Utilizing the known independence, we can reason about each outcome separately such that the output space is only $2 \times N$

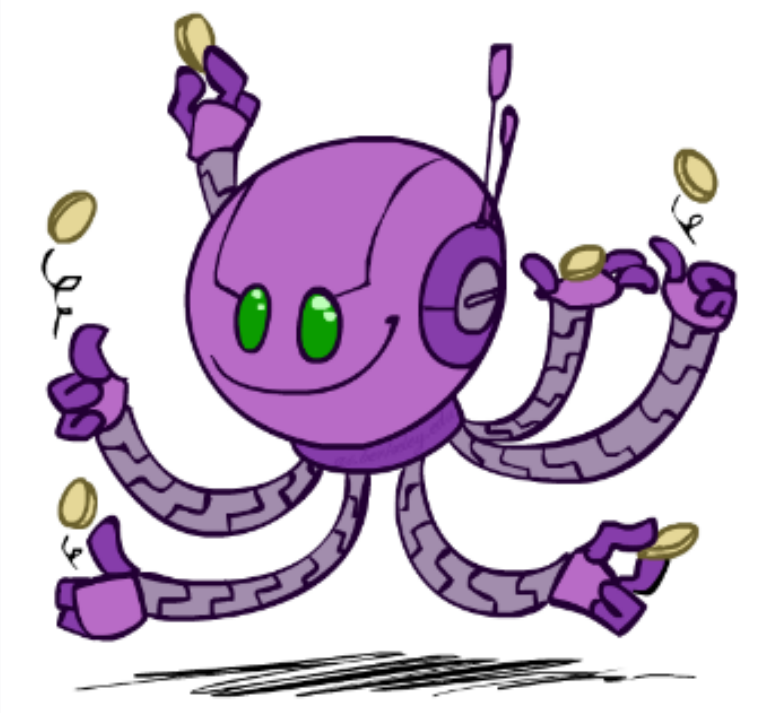


Image Credit: Pieter Abbeel

Example: Deep Generative Models

We often want to assume datapoints are independent given a generative model

Images from CelebA dataset

Factorizations

- Some factorizations allow for more simplification than others as they incorporate more information about independences
- Example: if $X_1 \perp\!\!\!\perp X_3 | X_2$ then the factorization

$$P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_2)$$

  is simpler than

$$P(X_1, X_2, X_3) = P(X_1)P(X_3|X_1)P(X_2|X_1, X_3)$$

- Critically though, we typically do not know **all** factorizations of a model when we construct it
  - For Bayesian problems we have direct access to $p(\theta)$ and $p(\mathcal{D}|\theta)$ but not $p(\mathcal{D})$ and $p(\theta|\mathcal{D})$: computation is required to uncover them
  - We can thus think about Bayesian inference as **finding unknown factorizations** of a joint

Graphical Models

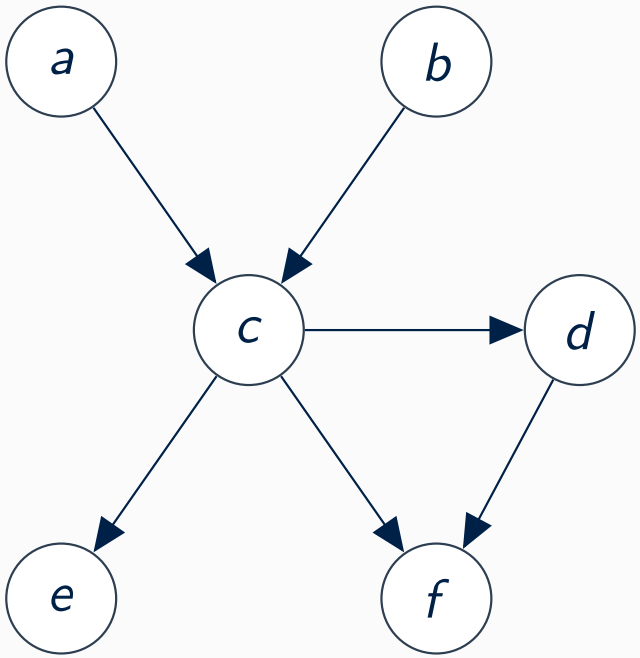
What is a Graphical Model?

- Generative models typically have many variables with a complex **dependency structure**.
- **Graphical models** are a ubiquitous method for representing and reasoning about generative models, with a particular focus on the dependency structure.
- They are formed by a number of connected nodes, where each node represents a random or observed variable in the model.
- Links between nodes represent dependencies: any two connected nodes have an explicit dependency, though unconnected nodes may still be dependent.

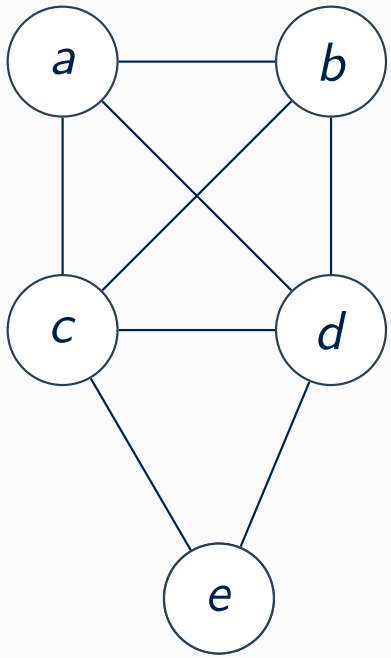
What is a Graphical Model? (2)

Graphical models can be separated into two distinct classes: **directed graphical models** and **undirected graphical models**.

Directed Graphical Model



Undirected Graphical Model



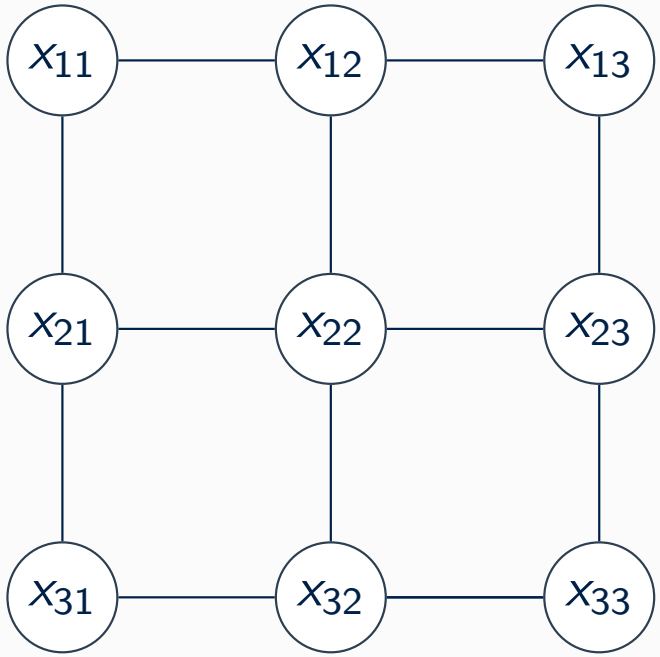
Undirected Graphical Models

- Undirected graphical models, also known as **Markov random fields**, imply no ordering to variables and are used only to express conditional independences
- They are useful for representing models that are difficult to express in a generative manner
- Independence in undirected graphical models can be deduced through the **global Markov property**:
  - Two variables A and B are conditionally independent given a subset of other variables C if there is no path between A and B that does not pass through C.
  - This implies that each variable is conditionally independent of all the other variables given its neighbors.
- We will not cover these in detail—see C M Bishop. *Pattern recognition and machine learning*. 2006, Section 8.3 instead

Undirected Graphical Model Example: Weather Prediction

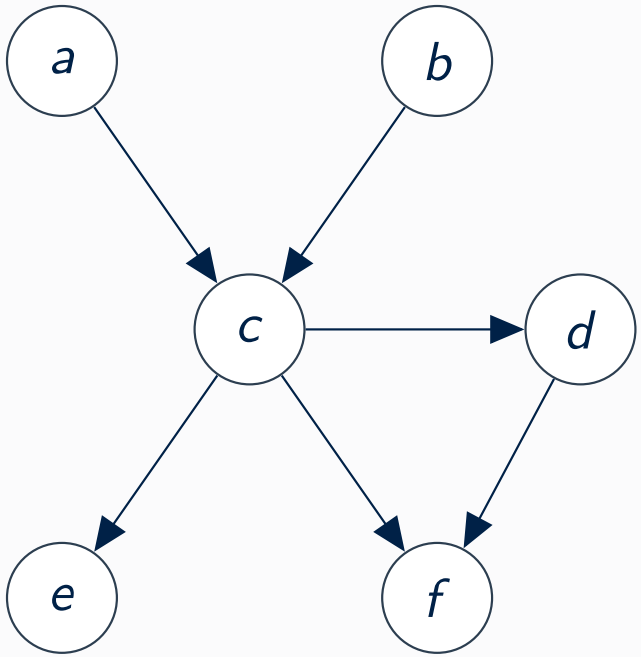
Predicting whether it will rain at various locations does not have a natural generative model

But the probability that it rains at one location is correlated to whether it rains at neighboring locations



Bayesian Networks / DAGs

- We'll focus on **directed acyclic graphical models** (DAGs), i.e. directed graphs containing no cycles or loops
- DAGs, also known as **Bayesian networks**, define a factorization for a generative model



$$p(a, b, c, d, e, f)$$
$$= p(a)p(b)p(c|a, b)$$
$$p(d|c)p(f|c, d)p(e|f)$$

Bayesian Networks / DAGs (2)

- To define a model using a DAG, we need to be able to define the probability of each variable given its **parents** (i.e. the nodes with arrows pointing towards our current node)
- For any node in a DAG containing the variables $x_{1:n}$

$$p(x_n|x_{1:N}\backslash x_n) = p(x_n|\text{parents}(x_n)) \tag{4}$$


D'YA
LIKE
DAGS?

Why Ya Should Like DAGs

- DAGs allows us to provide a piecewise explanation for the generative process
- They allow us to combine simple, local distributions into a larger, overall generative process
- This makes them a useful way to describe and construct Bayesian models

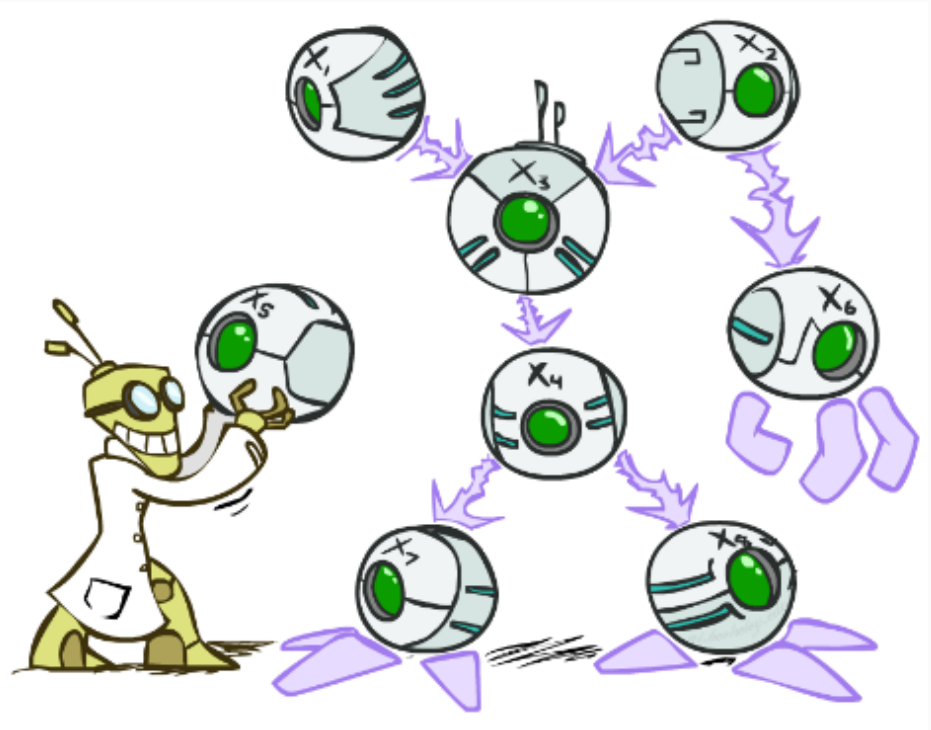


Image Credit: Pieter Abbeel

Example: Medical Diagnosis

- Imagine a medical diagnostic problem where we wish to predict if a patient has lung cancer.
- Let f denote unknown lifestyle and genetic factors of a hypothetical patient (e.g. whether they smoke)
- We encode this distribution of these as $p(f)$
- Our DAG just starts with a single node f

$p(f)$

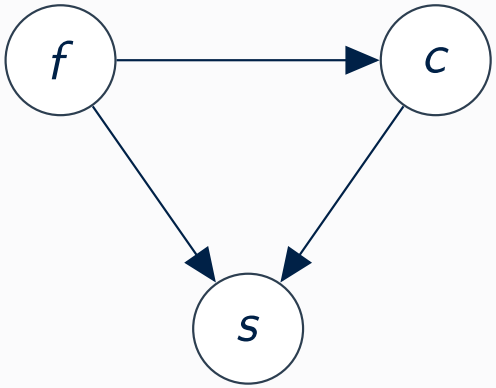
Example: Medical Diagnosis

- Given f, we can develop a model for the probability that a patient will develop lung cancer
- We encode this through the conditional distribution $p(c|f)$ where $c = 1$ indicates cancer is present
- We thus add a node c to the DAG and an arrow from f to c to represent this conditional dependency

$$p(f)p(c|f)$$

Example: Medical Diagnosis

- Given f and c, we can predict what symptoms, s, might be observed, e.g. a persistent cough
- We encode this through the conditional distribution $p(s|f, c)$ and add the new node s to the DAG
- We now have the complete DAG which provides a factorization of our model



$$p(f)p(c|f)p(s|c, f)$$

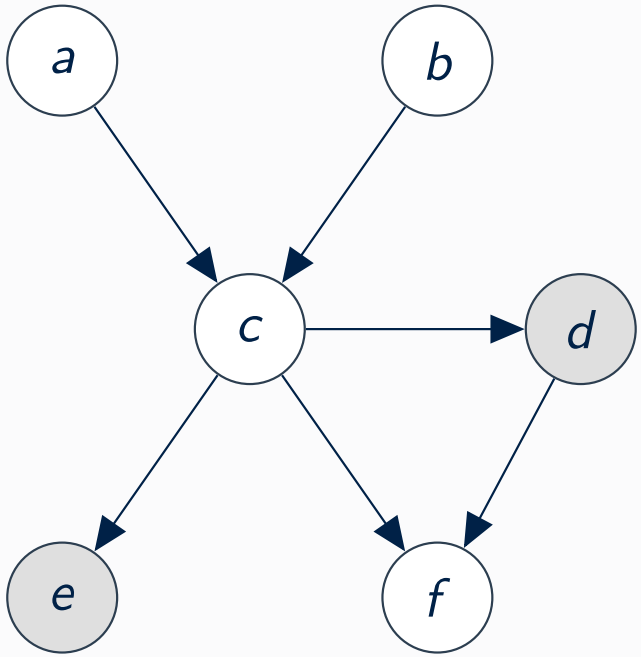
Observed Nodes

- The power DAGs becomes apparent once we start considering **observing** nodes
- This equivalent to conditioning in a Bayesian model: we fix a node to take on a certain observed value
- Observed nodes in a DAG are denoted through shading
- We can use the DAG to reason about unobserved variables conditioned on the observed ones

Image Credit: Pieter Abbeel

Observed Nodes

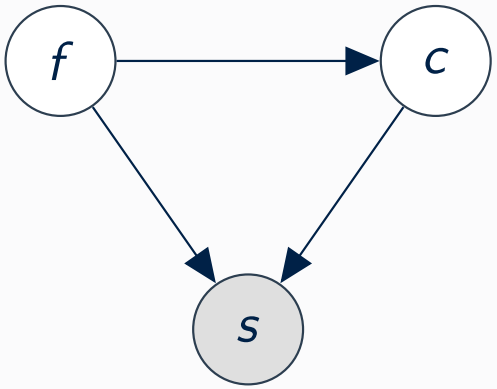
- The power DAGs becomes apparent once we start considering **observing** nodes
- This equivalent to conditioning in a Bayesian model: we fix a node to take on a certain observed value
- Observed nodes in a DAG are denoted through shading
- We can use the DAG to reason about unobserved variables conditioned on the observed ones



$$p(c|d,e) \propto$$
$$p(a)p(b)p(c|a,b)p(d|c)p(e|c)$$

Example: Medical Diagnosis

- For our medical diagnosis example we actually observe the symptoms
- We can thus perform Bayesian inference to try and infer whether the patient has cancer.
- In general, posteriors are not available in closed form and DAGs provide a useful tool for providing helpful information to calculate them: see Bishop Chapter 8

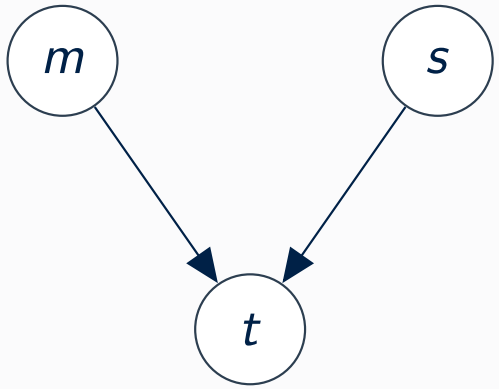


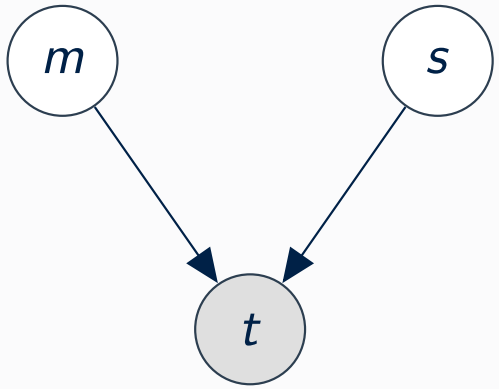
$$p(c|s) \propto p(f)p(c|f)p(s|c,f)$$

Explaining Away

- Sometimes conditioning can introduce dependencies we might not at first expect
- Consider a model for speed camera being triggered, t
- This depends on both whether the camera is malfunctioning, m, and whether the car is speeding, s, as per the DAG on the right
- m and s are clearly marginally independent
- But if t is observed they become dependent

$$p(m)p(s)p(t|m,s)$$

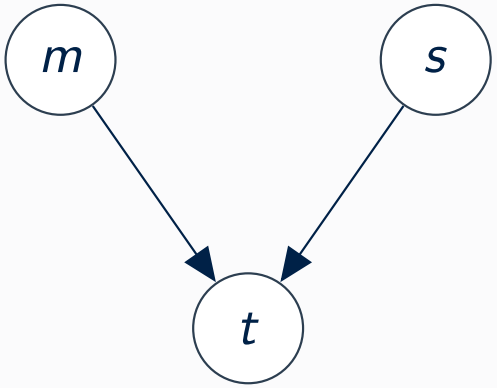


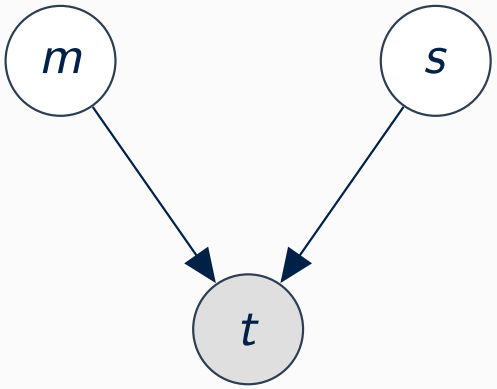


Explaining Away

- This phenomenon is known as **explaining away**
- Here m and s provided alternative explanations for t
- For example, if the camera is triggered, then the car must be speeding or the camera must either be malfunctioning
- If the camera is not triggered, we cannot have both that the car is not speeding and the camera is working

$$p(m)p(s)p(t|m,s)$$

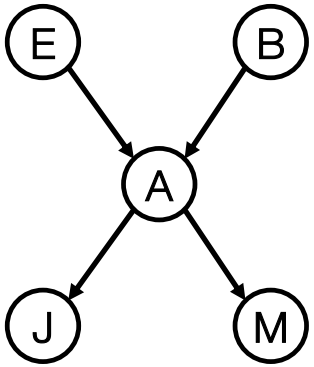


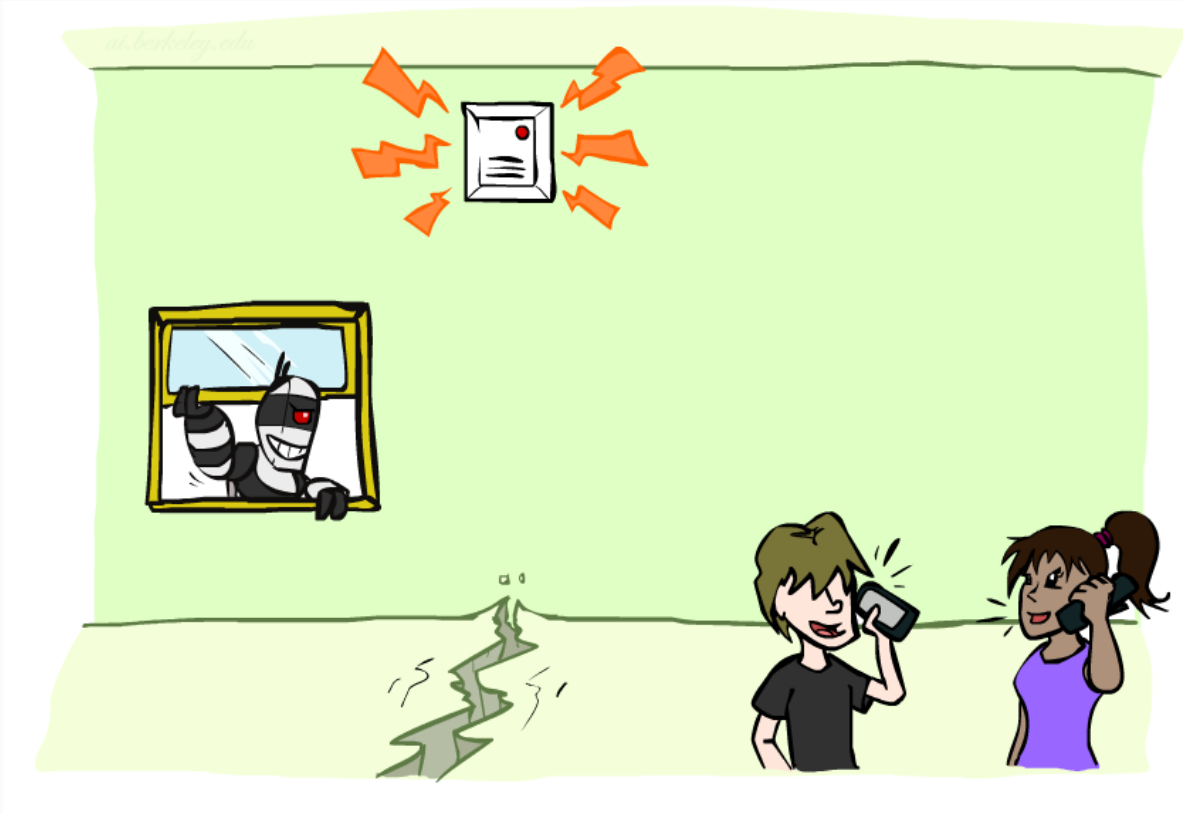


Alarm Triggering Example

Variables

- B: Burglary
- A: Alarm goes off
- M: Mary calls
- J: John calls
- E: Earthquake!

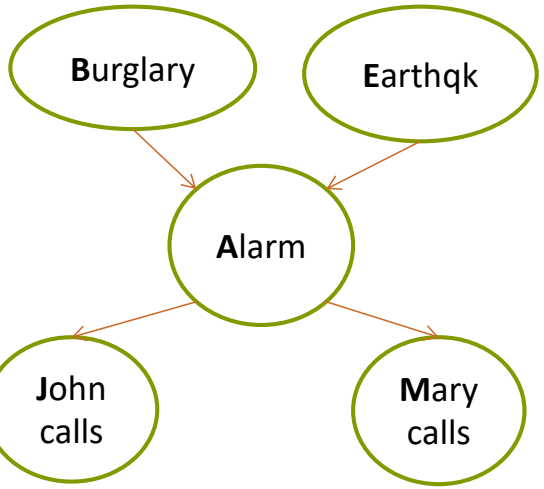




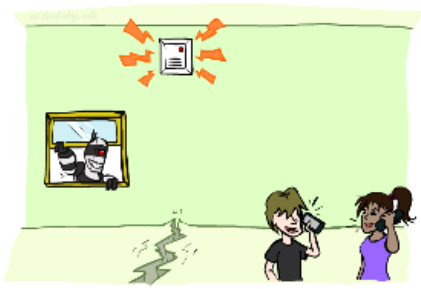
Example and image credit: Pieter Abbeel

Alarm Triggering Example (2)

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

Example and image credit: Pieter Abbeel

Alarm Triggering Example (3)

Optional homework: use this Bayes net to calculate:

$$P(\text{John Calls}|\text{Burglary}) \tag{5}$$

$$P(\text{Burglary}|\text{John Calls}) \tag{6}$$

$$P(\text{Earthquake}|\text{Mary Calls}, \text{Burglary}) \tag{7}$$

If you're looking for a challenge (optional), try your hand at this homework assignment from Frank Wood:

`http://www.robots.ox.ac.uk/~fwood/teaching/AIMS_CDT_ML_2016/homework/HW_1_sum_product/`

Note this will require substantial extra reading (Bishop Chapter 8) and most likely a few hours to implement.

D–Separation

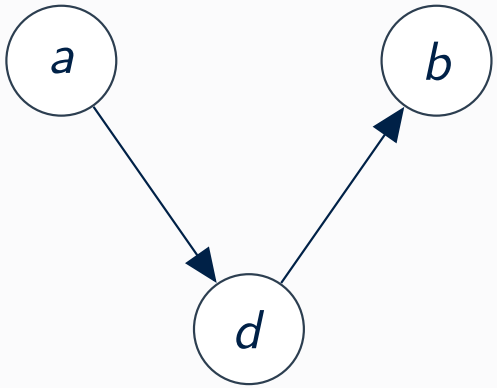
- How can we deduce the independence relationships from a DAG?
- **d-separation** gives us rules for doing this
- Consider three arbitrary, non-intersecting, subsets A, B, and C of a DAG.
- A and B are conditionally independent given C if there are no **unblocked** paths from A to B when C is observed
    - A is said to be d-separated from B by C

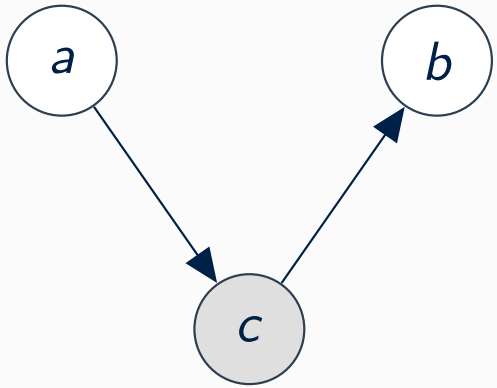
Paths

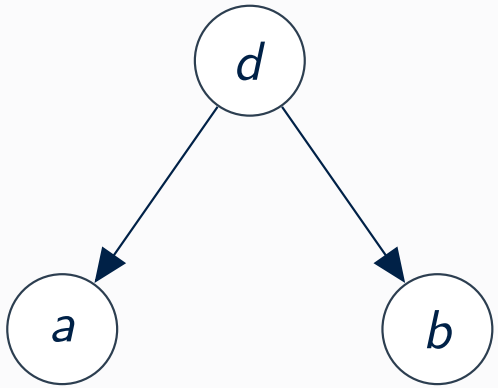
- Paths do not need to be in the same directions as the arrows in the DAG—we just have to move between connected nodes
- A path between A and B is blocked if
  1. There is an unobserved node, n, in the path where both the arrows point towards n and n has no observed descendants
  2. Consecutive arrows in the path meet at an observed node and either one or both of them points away from the node.
- Note that only the first of these rules is necessary for establishing marginal independence

Blocked or Not?

Is the path between a and b blocked?





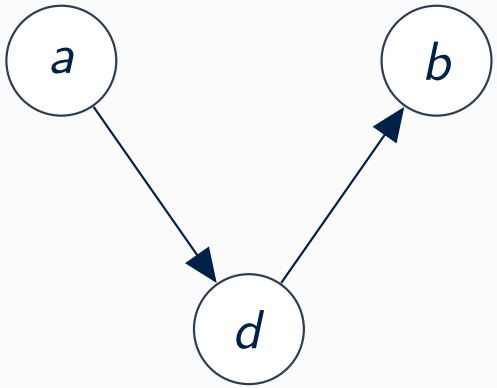


Rule 1 There is an unobserved node, n, in the path where both the arrows point towards n and n has no observed descendants

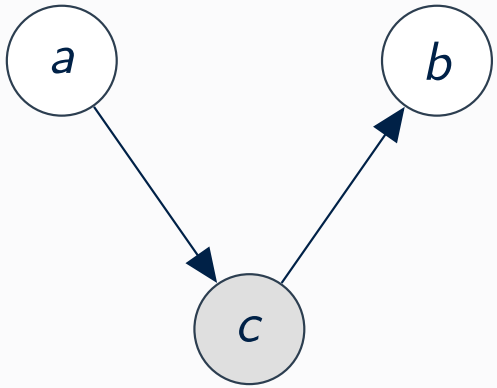
Rule 2 Consecutive arrows in the path meet at an observed node and either one or both of them points away from the node.

Blocked or Not?

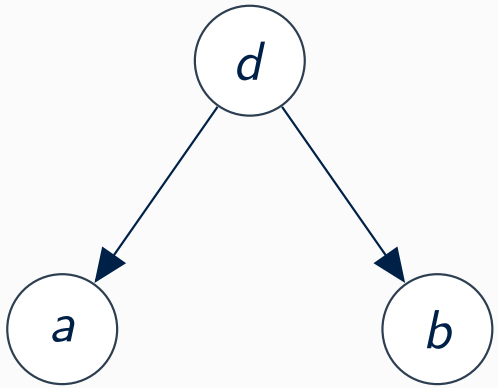
Is the path between a and b blocked?



Unblocked



Blocked: 2nd rule



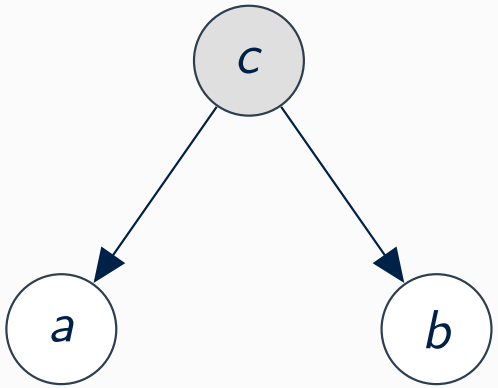
Unblocked

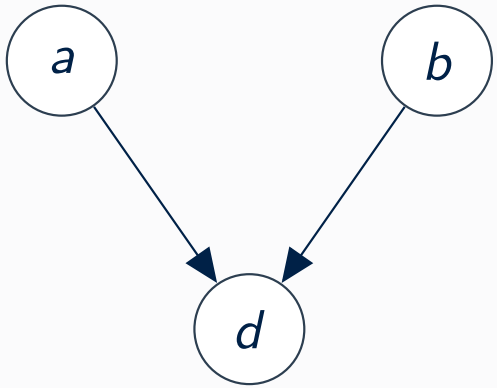
Rule 1 There is an unobserved node, n, in the path where both the arrows point towards n and n has no observed descendants

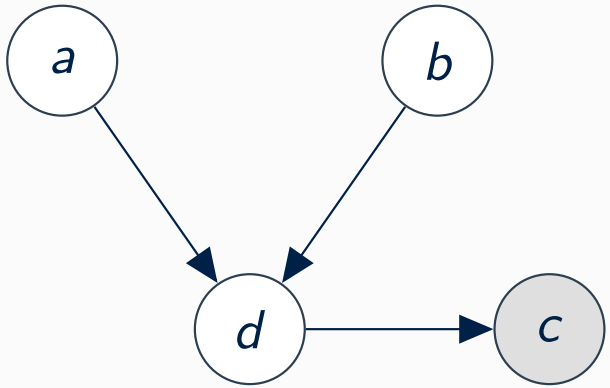
Rule 2 Consecutive arrows in the path meet at an observed node and either one or both of them points away from the node.

Blocked or Not (2)?

Is the path between a and b blocked?





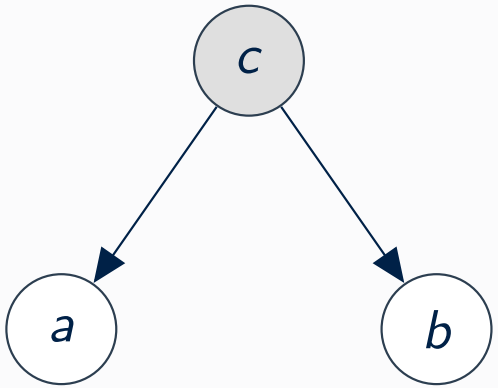


Rule 1 There is an unobserved node, n, in the path where both the arrows point towards n and n has no observed descendants

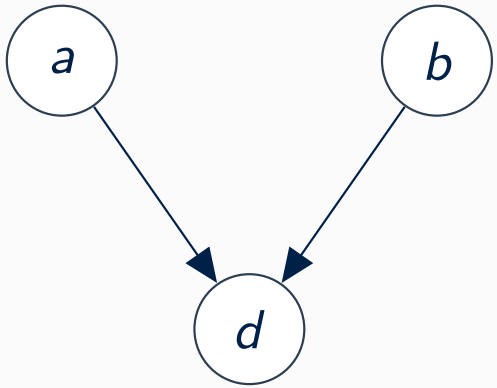
Rule 2 Consecutive arrows in the path meet at an observed node and either one or both of them points away from the node.

Blocked or Not (2)?

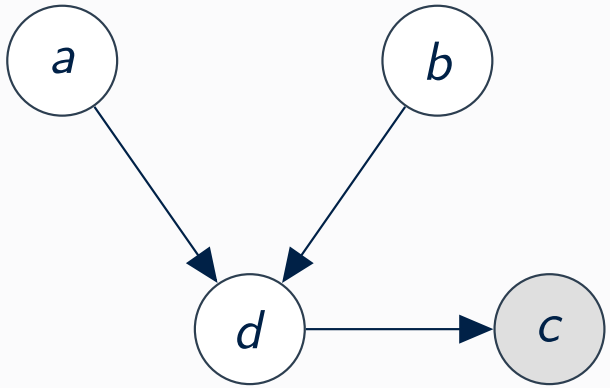
Is the path between a and b blocked?



Blocked: 2nd rule



Blocked: 1st rule



Unblocked: explaining away

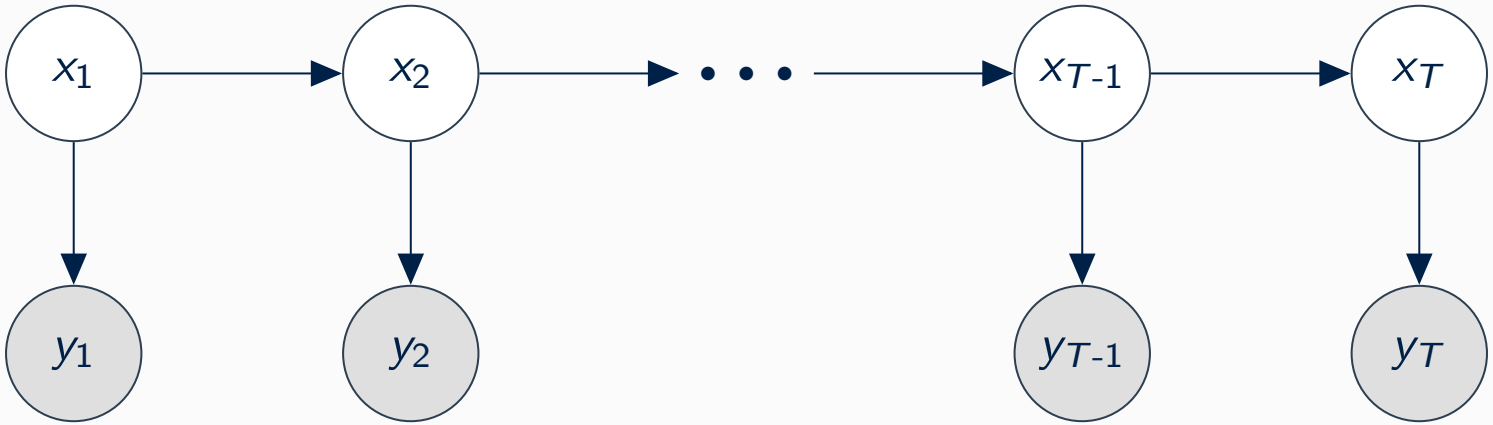
Rule 1 There is an unobserved node, n, in the path where both the arrows point towards n and n has no observed descendants

Rule 2 Consecutive arrows in the path meet at an observed node and either one or both of them points away from the node.

Hidden Markov Models

We finish our introduction to graphical models by considering a common model where the dependency structure is very important: **Hidden Markov Models** (HMMs)

- Bishop has almost a whole chapter dedicated to these (Chapter 13)



Hidden Markov Models (2)

As we see from its DAG, a HMM has T latent variables $x_{1:T}$ and T observations $y_{1:T}$. The joint distribution is as follows

$$p(x_{1:T}, y_{1:T}) = p(x_1)p(y_1|x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})p(y_t|x_t), \qquad (8)$$

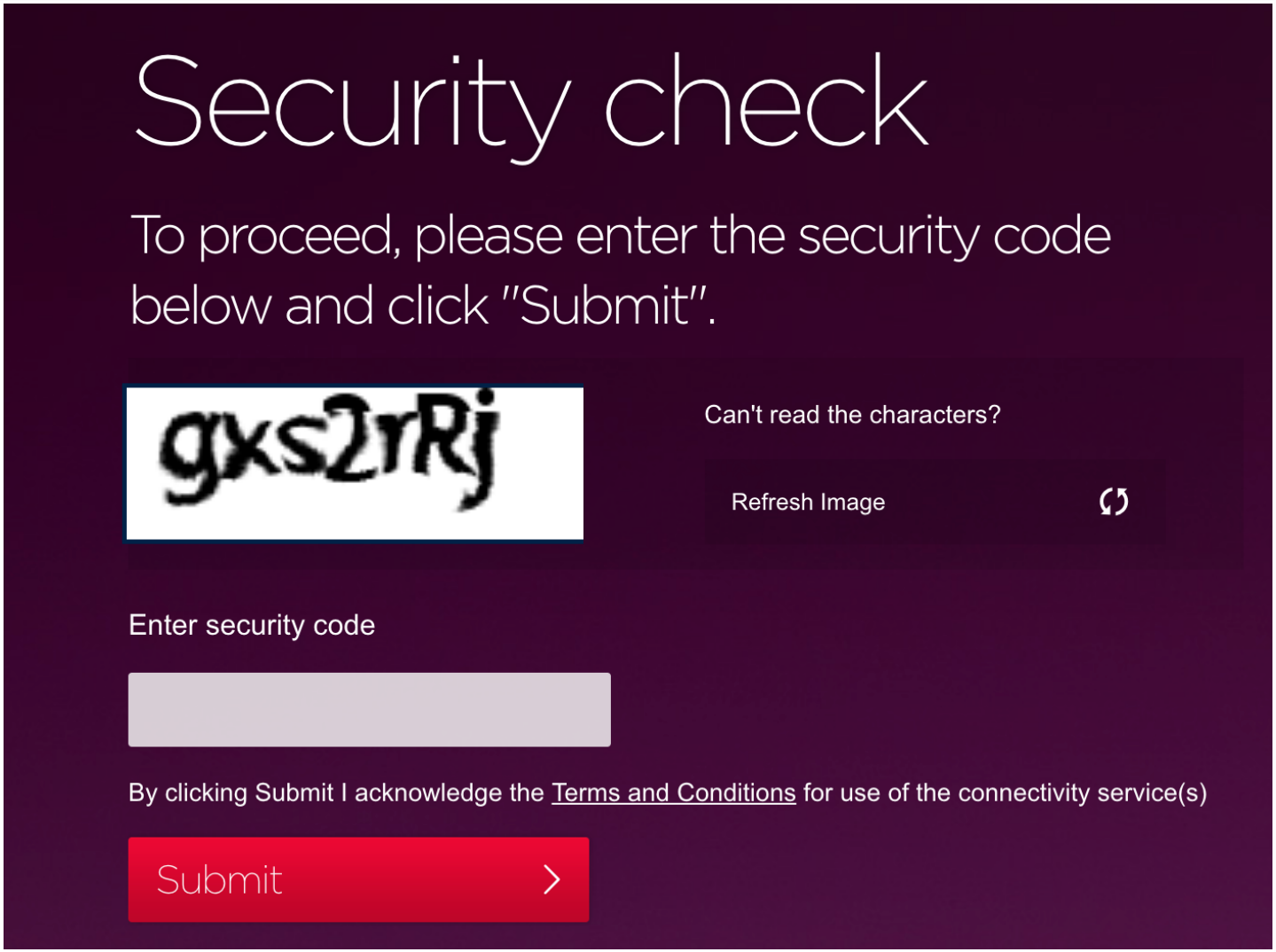
where each x_t is independent of $x_{1:t-2}$ and $y_{1:t-1}$ given x_{t-1}, and of $x_{t+2:T}$ and $y_{t+1:T}$ given x_{t+1}.

This Markov property means the model has **no memory** as information is passed forwards (or backwards) only through the value of the previous (or next) latent state:

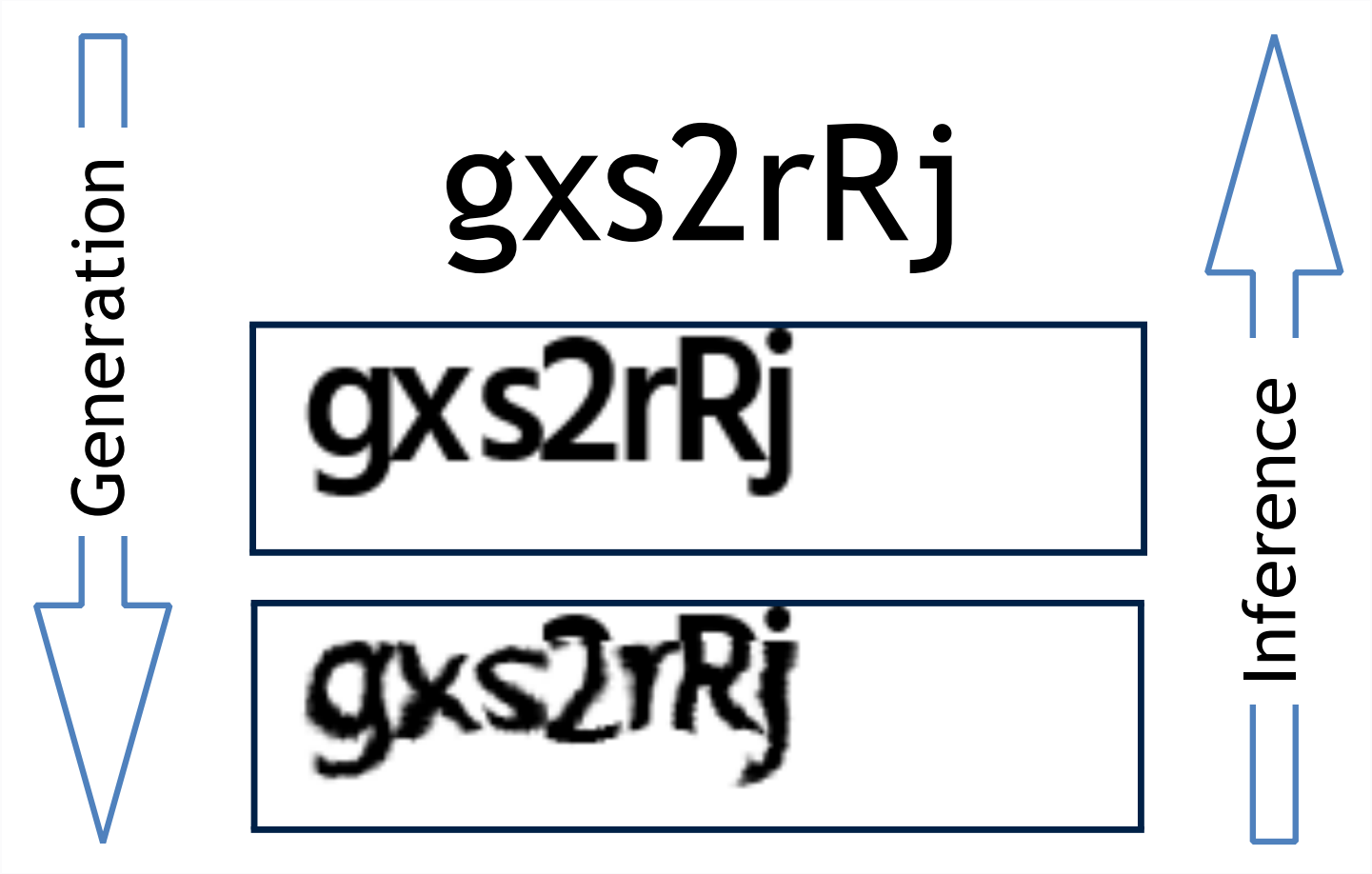
Many dynamical systems obey the Markov property: HMMs are extensively used for a number of tasks involving sequential data, such as tracking and DNA sequencing

Probabilistic Programming

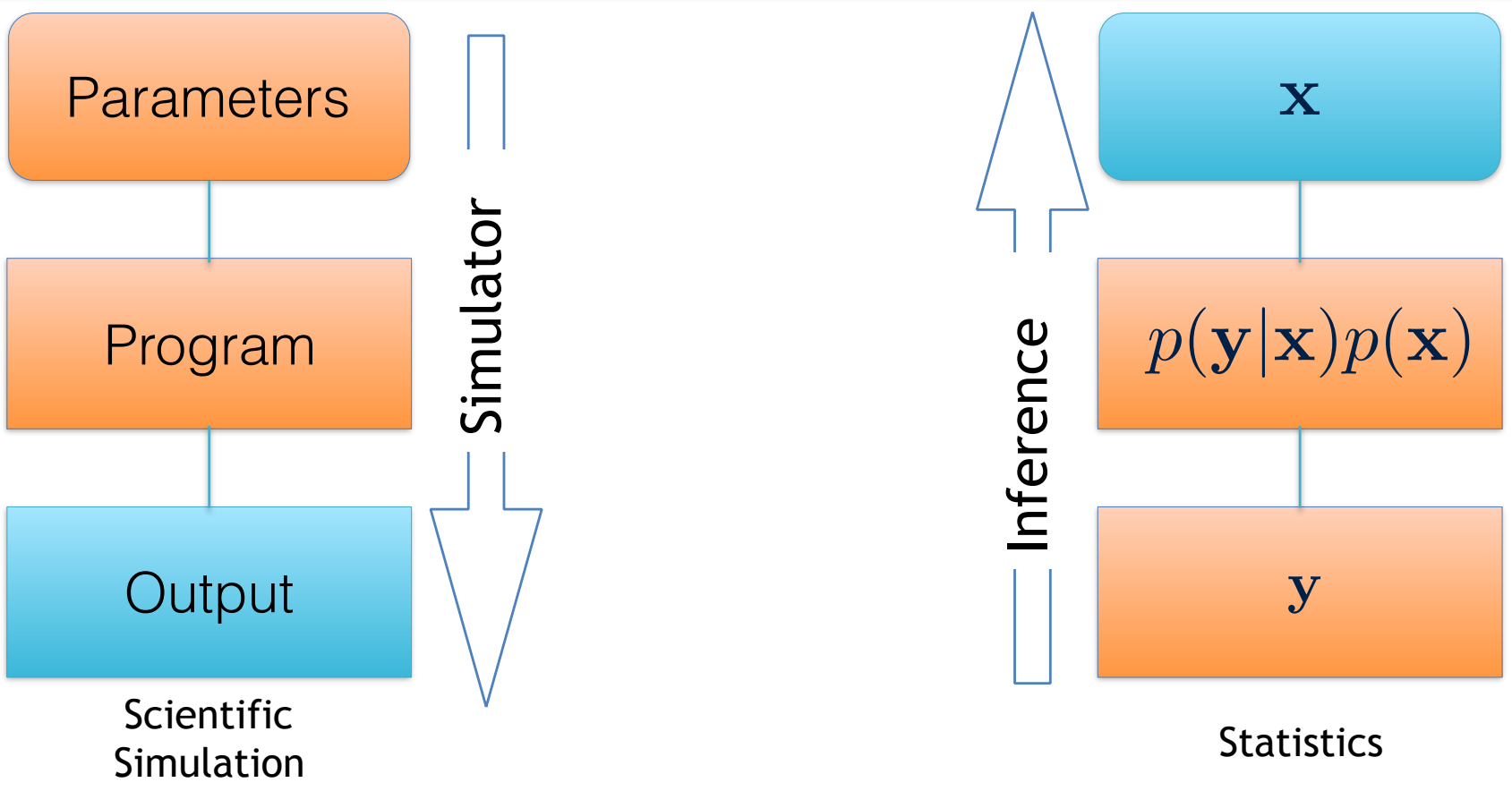
Captchas Revisited



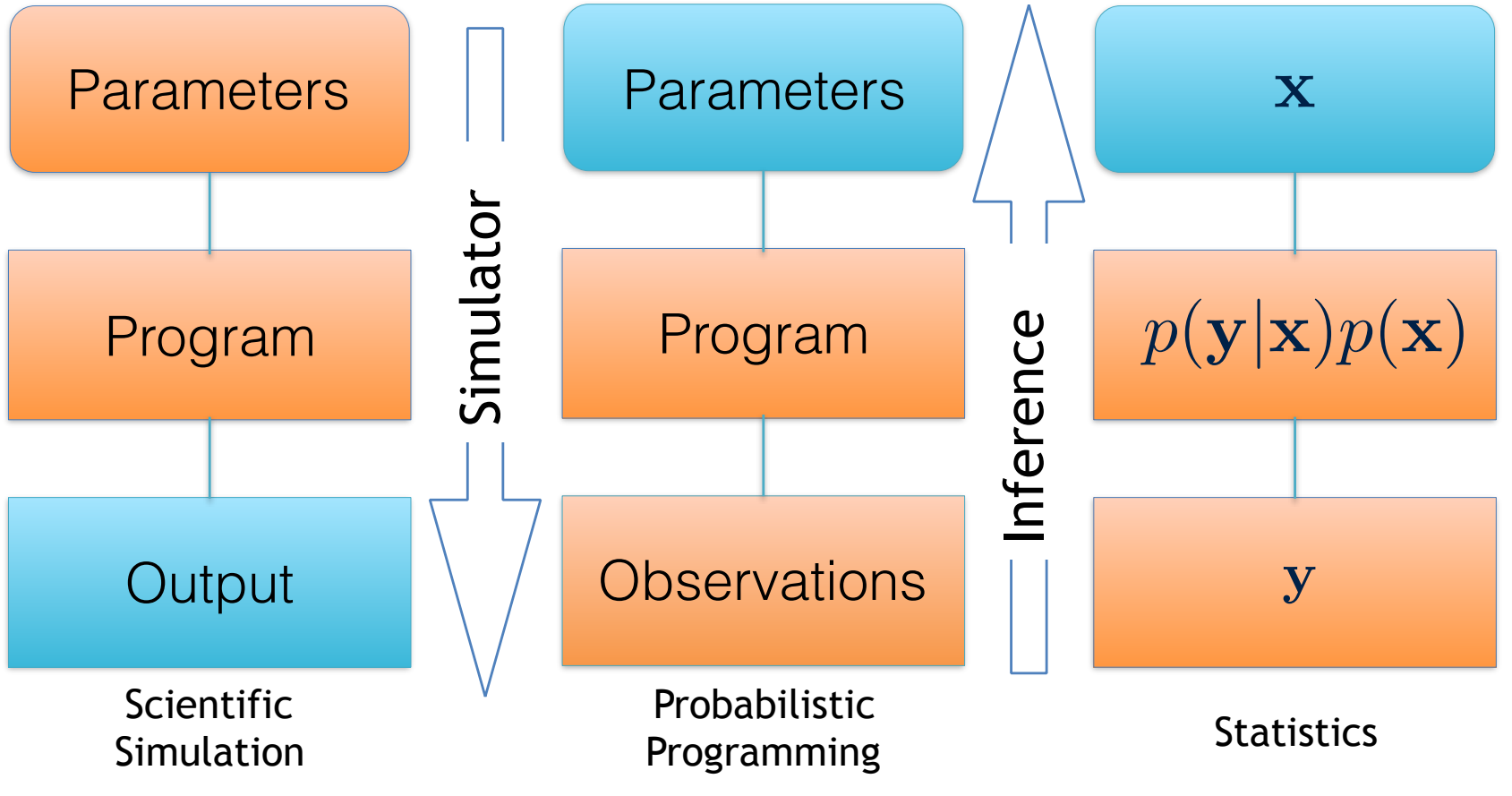
Captchas Revisited (2)



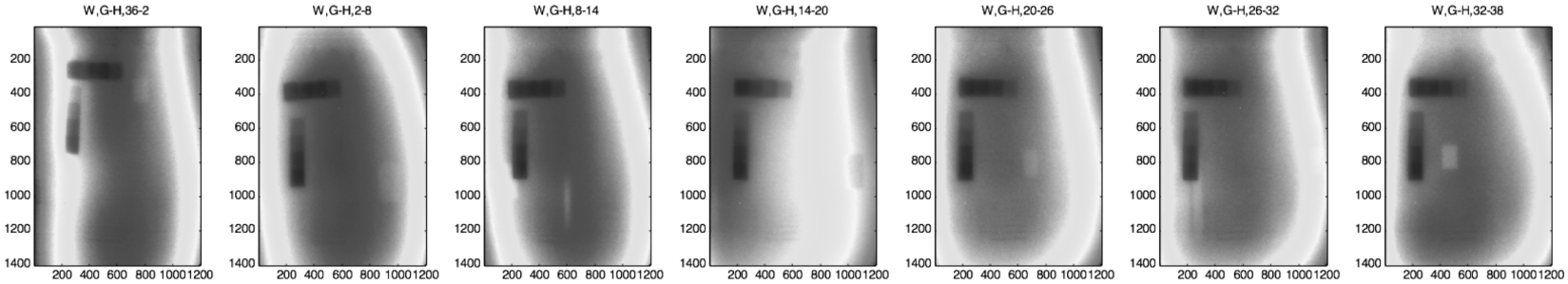
Probabilistic Programs are Simulator–Based Models

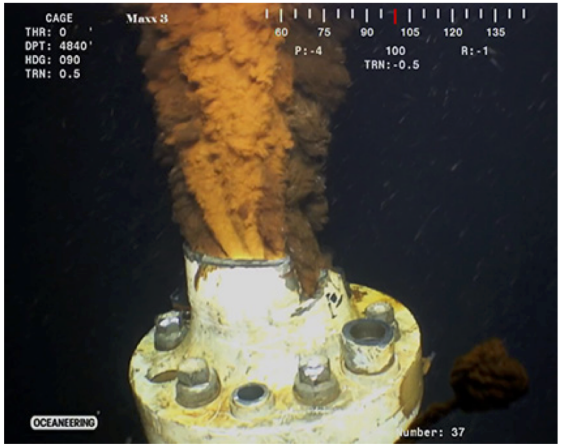


Probabilistic Programs are Simulator–Based Models



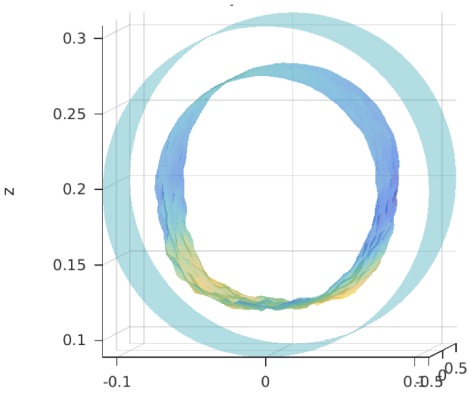
Example: Deep Sea Oil Pipes



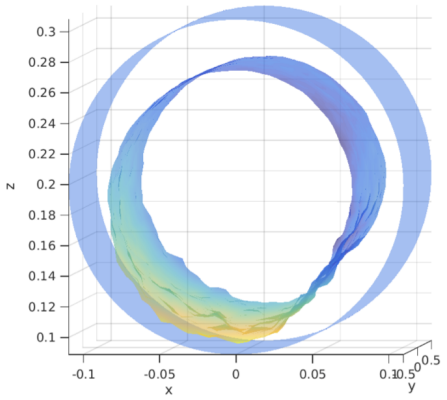


?

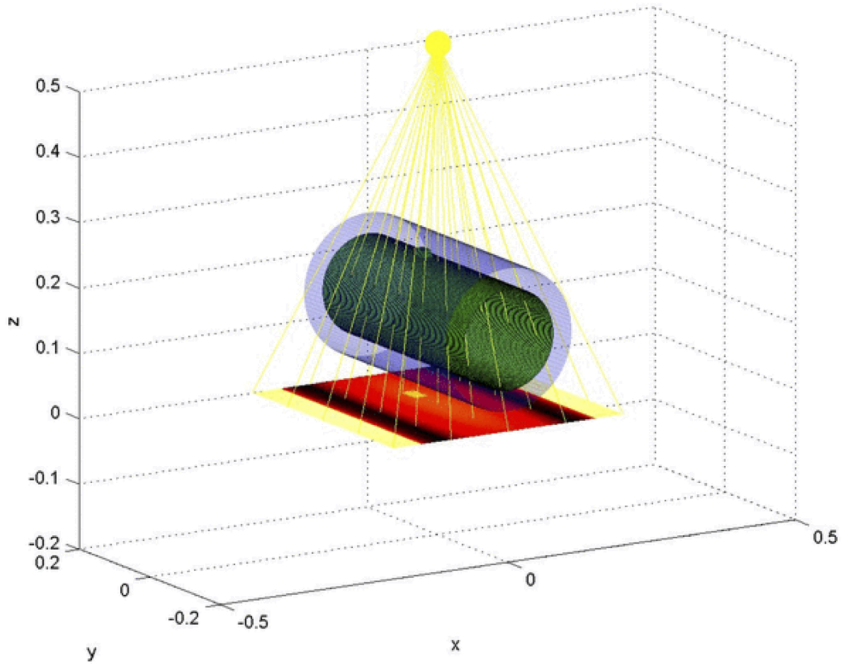
Example: Deep Sea Oil Pipes (2)



Pipe sample 1



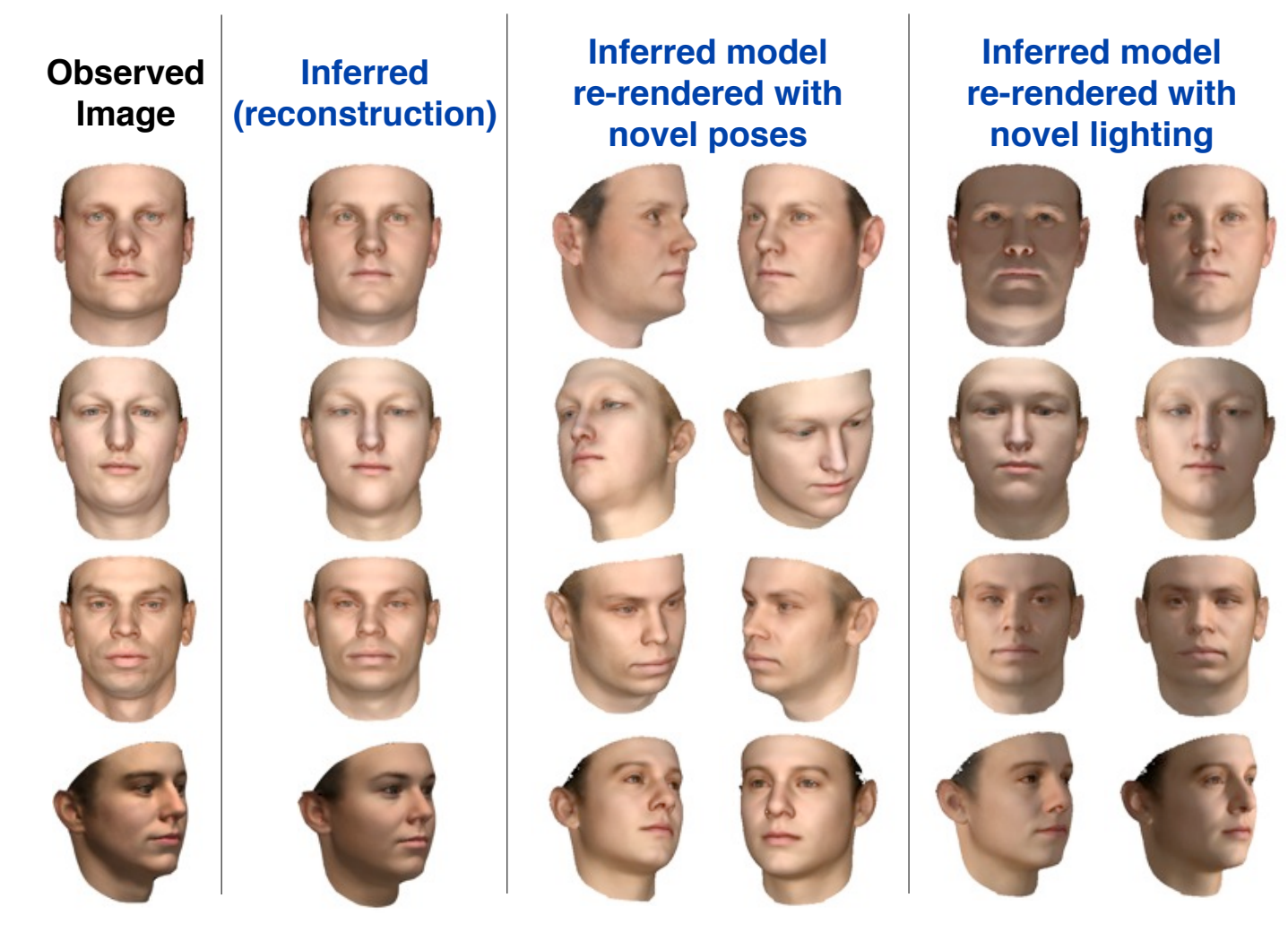
Pipe Sample 2



$$p(\theta)$$

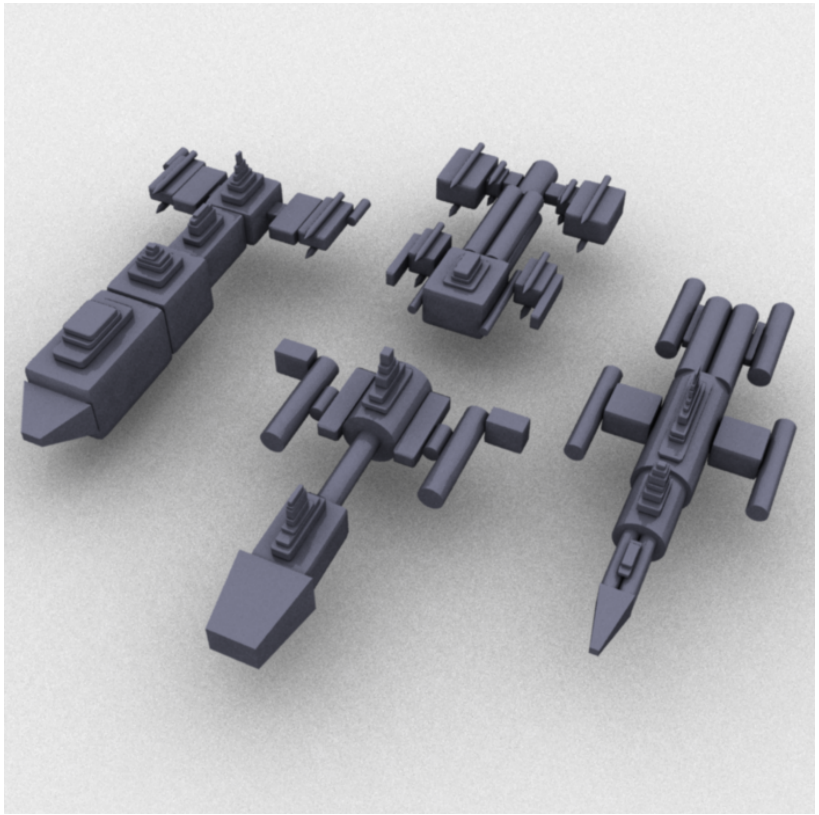
$$p(\mathcal{D}|\theta)$$

Application: Scene Perception

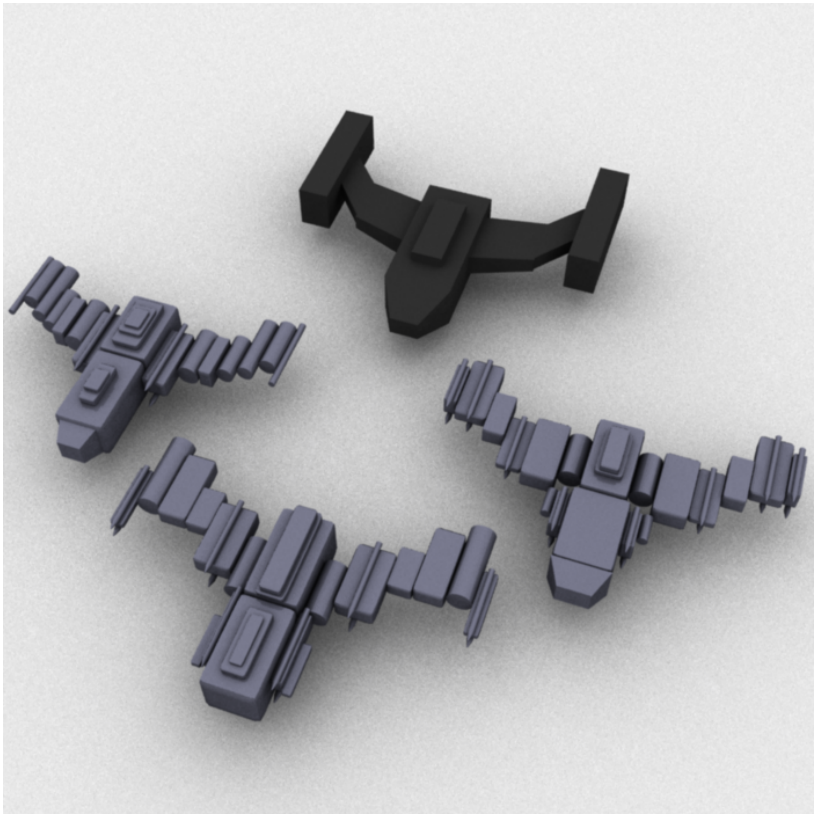


[1] T D Kulkarni et al. "Picture: A probabilistic programming language for scene perception". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Application: Guided Object Generation



Forward Sampling



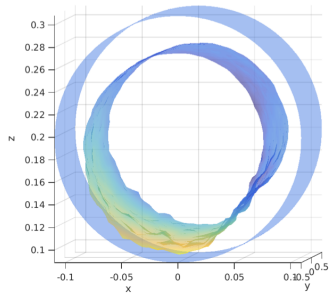
SOSMC-Controlled Sampling

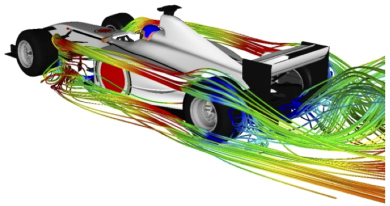
[2] D Ritchie et al. "Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo". In: *ACM Transactions on Graphics (TOG)* (2015).

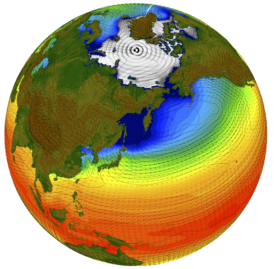
What is a Probabilistic Programming System?

- Probabilistic programming systems (PPSs) allow Bayesian models to be represented in the form of a stochastic simulator and statements for conditioning on data
- Informally, one can think of the generative model as the definition of a prior, the conditioning statements as the definition of a likelihood, and the output of the program as samples from a posterior distribution.
- Their core philosophy is to decouple model specification and inference
- Inference is automated for any model the user might write

Separating Models from Inference

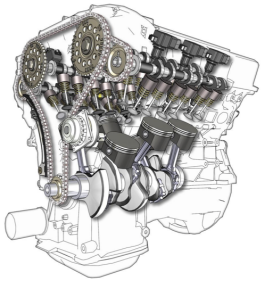






Simulator = User's Responsibility

Programming Language Representation / Abstraction Layer



Algorithm 1 Sequential Monte Carlo (all for $i = 1, \ldots, N$)

Input: data $y_{1:T}$, number of particles N, proposals q_t
$x_1^i \sim q_1(x_1)$
$w_1^i = \frac{g_1(y_1|x_1^i)\mu(x_1^i)}{q_1(x_1^i)}$
for $t = 2$ **to** T **do**
  $a_{t-1}^i \sim \text{Discrete}\left(\{\bar{w}_{t-1}^\ell\}_{\ell=1}^N\right)$
  $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i})$
  Set $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$
  $w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_{1:t}^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}$
end for

Algorithm 3 iPMCMC sampler

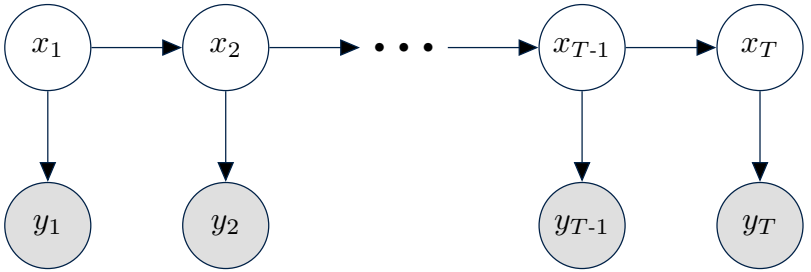
Input: number of nodes M, conditional nodes P and MCMC steps R, initial $\mathbf{x}'_{1:P}[0]$
for $r = 1$ **to** R **do**
  Workers $1 : M \backslash c_{1:P}$ run Algorithm 1 (SMC)
  Workers $c_{1:P}$ run Algorithm 2 (CSMC), conditional on $\mathbf{x}'_{1:P}[r-1]$ respectively.
  for $j = 1$ **to** P **do**
    Select a new conditional node by simulating c_j according to (5).
    Set new MCMC sample $\mathbf{x}'_j[r] = \mathbf{x}_{c_j}^{b_j}$ by simulating b_j according to (7)
  end for
end for

Inference Engine(s) = Developer's Responsibility

Using Information in the Code

Inputs: Transition std-dev σ, output shape α, output rate β, data $y_{1:T}$

1: $x_0 \leftarrow 0$
2: `tr-dist` $\leftarrow$ **`normal`** $(0, \sigma)$
3: `obs-dist` $\leftarrow$ **`gamma`** (α, β)
4: **for** $t = 1, \ldots, T$ **do**
5: $\quad x_t \leftarrow x_{t-1} +$ **`sample`** (`tr-dist`)
6: $\quad$ **`observe`** (`obs-dist`, $y_t - x_t$)
7: $\quad z_t \leftarrow \mathbb{I}(x_t > 4)$
8: **end for**
9: **return** $z_{1:T}$



$$p(x_{1:T}, y_{1:T} | \sigma, \alpha, \beta) =$$
$$\mathcal{N}(x_1; 0, \sigma^2)\ \textsc{Gamma}(y_1 - x_1; \alpha, \beta)$$
$$\prod_{t=2}^{T} \mathcal{N}(x_t - x_{t-1}; 0, \sigma^2)\ \textsc{Gamma}(y_t - x_t; \alpha, \beta)$$

Worked Example: Bayesian Linear Regression

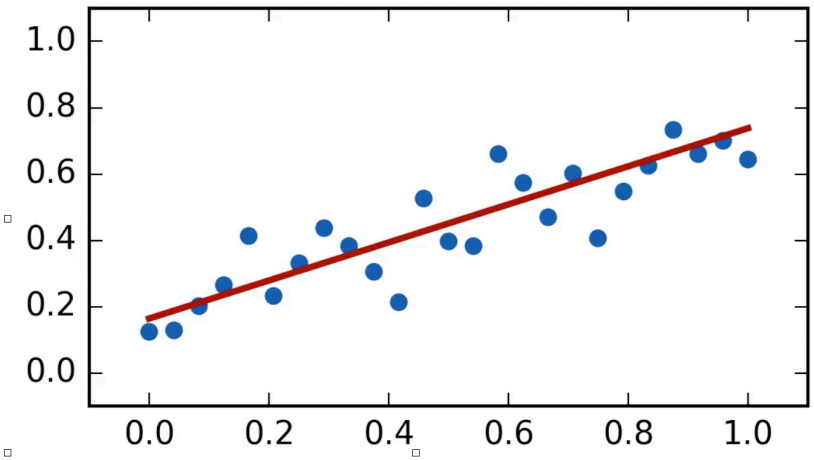
Inputs: Input points $u_{1:N}$

1: $m \leftarrow$ **sample** (**normal** (0,1))
2: $c \leftarrow$ **sample** (**normal** (0,1))
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad \mu_n \leftarrow m u_n + c$
5: $\quad v_n \leftarrow$ **sample** (**normal** $(\mu_n, 0.1)$)
6: **end for**
7: **return** $m,\ c,\ v_{1:N}$

Worked Example: Bayesian Linear Regression

Inputs: Data $u_{1:N}$, $v_{1:N}$

1: $m \leftarrow$ **sample** (**normal** (0,1))
2: $c \leftarrow$ **sample** (**normal** (0,1))
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad \mu_n \leftarrow m u_n + c$
5: $\quad$ **observe** (**normal** $(\mu_n, 0.1)$, v_n)
6: **end for**
7: **return** $m,\ c$



$$\theta = \{m, c\} \qquad \mathcal{D} = \{u_n, v_n\}_{n=1}^{N}$$

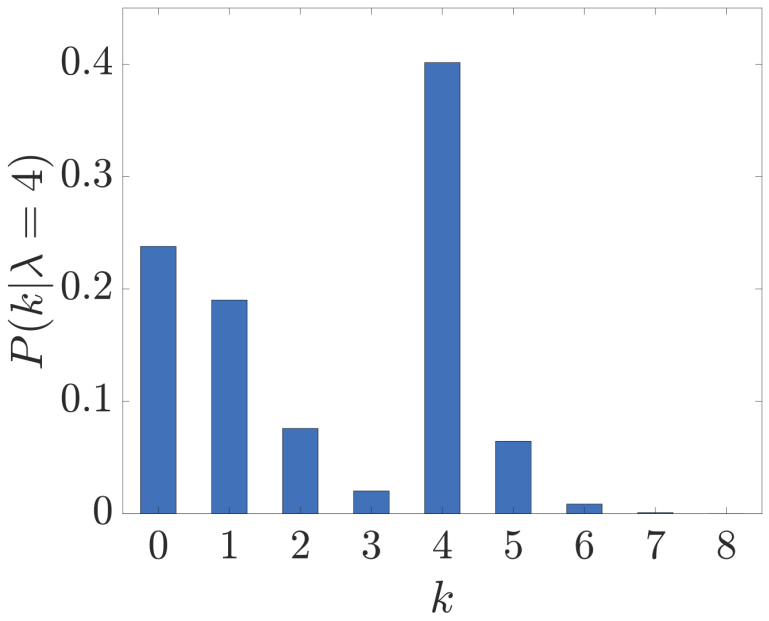
Two Approaches to System Design

- Probabilistic programming is more of an umbrella term than an exactly defined approach
- There are two philosophies systems are built around:
  - **Inference driven** PPSs start with a specific inference method(s) and build a language around making it as easy as possible to write models suitable for that method(s)
  - **Model driven** PPSs start with language capable of defining any computable distribution and try and construct inference engines capable of working on any program
    - These are sometimes based on Turing–complete languages, in which case they are known as **universal** PPSs

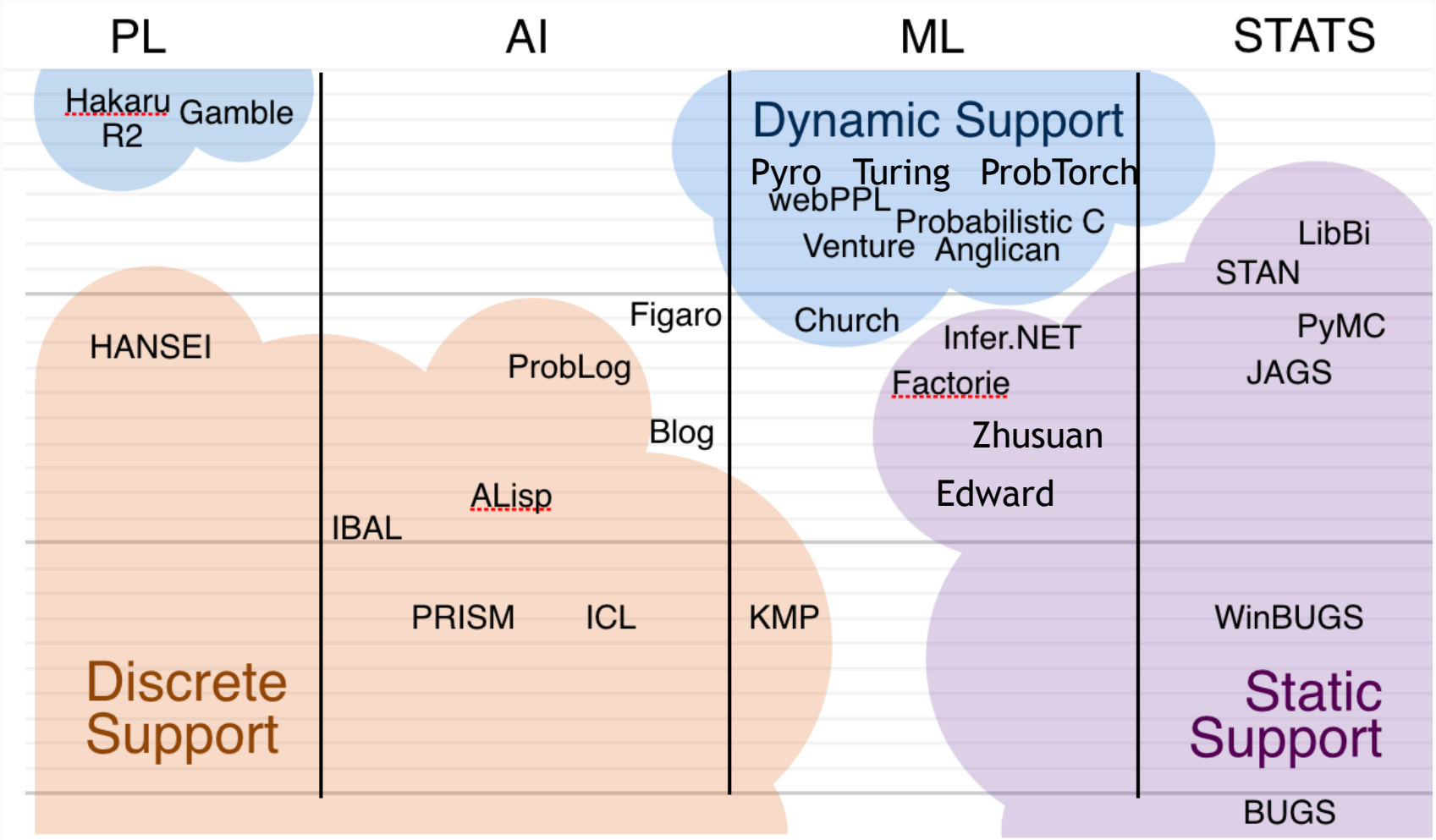
Universal PPSs allow Stochastic Control Flow

Inputs: Event rate λ

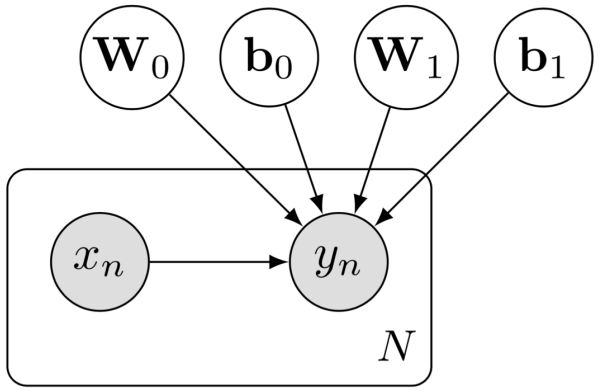
1: $L \leftarrow \exp(-\lambda),\ k \leftarrow 0,\ p \leftarrow 1$
2: **while** $p > L$ **do**
3: $\quad u \leftarrow$ `sample` (`uniform` (0,1))
4: $\quad p \leftarrow pu$
5: $\quad$ **if** $p \leq L$ **then**; **break while**; **end if**
6: $\quad$ `observe` (`bernoulli` (0.2), 1)
7: $\quad k \leftarrow k + 1$
8: **end while**
9: `observe` (`bernoulli` (0.99), $\mathbb{I}(k{>}3)$)
10: **return** k



Example PPSs



Deep PPSs



Edward

PROB TORCH

PYRO

ZHUSUAN

```
1  W_0 = Normal(mu=tf.zeros([D, H]), sigma=tf.ones([D, H]))
2  W_1 = Normal(mu=tf.zeros([H, 1]), sigma=tf.ones([H, 1]))
3  b_0 = Normal(mu=tf.zeros(H), sigma=tf.ones(L))
4  b_1 = Normal(mu=tf.zeros(1), sigma=tf.ones(1))
5
6  x = tf.placeholder(tf.float32, [N, D])
7  y = Bernoulli(logits=tf.matmul(tf.nn.tanh(tf.matmul(x, W_0) + b_0), W_1) + b_1)
```

[3] D Tran et al. "Edward: A library for probabilistic modeling, inference, and criticism". In: *arXiv preprint arXiv:1610.09787* (2016).

Recap

- Conditional independences are a very important modeling assumption
- Graphical models, and in particular DAGs, allow us to construct complex generative models by piecing together components of the model
- Observing nodes in graph allow us to condition on observations and thus perform Bayesian reasoning
- Probabilistic programming gives us a means of encoding Bayesian models as code
- It further provides automated inference engines to calculate the posterior

Next time: Bayesian inference—actually calculating posteriors

Further Reading

- The lecture notes provide a more thorough and technical introduction to probabilistic programming
- Eric Xing's course on Probabilistic Graphical Model:

  `http://www.cs.cmu.edu/~epxing/Class/10708-14/lecture.html`
- Pieter Abbeel and Dan Klein on Bayes Nets (Lectures 16 to 19 in the recommended list): `http://ai.berkeley.edu./lecture_videos.html`
- Bishop, *Pattern recognition and machine learning*, Chapters 8
- D Barber. *Bayesian reasoning and machine learning*. 2012, Chapters 2-4
- Video tutorial on probabilistic programming by Frank Wood:

  `https://www.youtube.com/watch?v=Te7A5JEm5UI&t=500s`
- Full conference of talks on probabilistic programming:

  `https://www.youtube.com/channel/UCTFDb7aQY1ewBYwJJrpKp6Q`
- Websites for individual PPSs (Pyro is a good place to start)