Probabilistic Programming

Lecture #1: Introduction

Joost-Pieter Katoen

RWTH Lecture Series on Probabilistic Programming 2022-23

Overview

Overview

Zoubin Gharahmani, Uber & University of Cambridge


REVIEW

doi:10.1038/nature14541

Probabilistic machine learning and artificial intelligence

Zoubin Ghahramani[1]

Perspective

"There are several reasons why probabilistic programming could prove to be revolutionary for machine intelligence and scientific modelling."

[Ghahramani, Nature 2015]

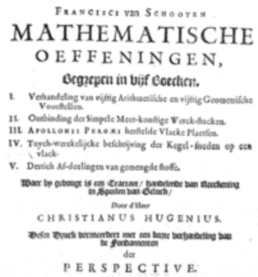
Why? Probabilistic programming

1. … obviates the need to manually provide inference methods
2. … enables rapid prototyping
3. … clearly separates the model and the inference procedures

Overview

The origin of probability theory

Christiaan Huygens



addendum to "Mathematische Oeffeningen" 1660[a]

[a] Kees Verduin, A Short History of Probability and Statistics

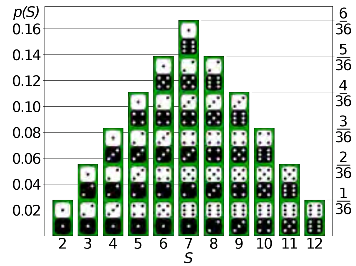
How “proofs” looked like

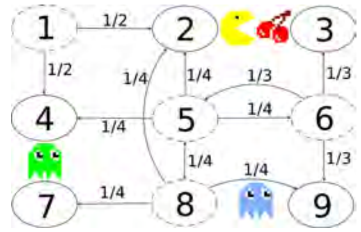
II. *VOORSTEL.*

Als ick gelijcke kans hebbe tot *a* of *b* of *c*, het is my soo veel weerdt als of ick $(a+b+c)/3$ hadde.

Om dit wederom te vinden, soo zy als vooren gestelt *x* voor de waerde van mijn kans. Soo moet ick dan *x* hebbende weder tot de selfde kansse konnen geraecken door rechtmaetig spel. Laet dit het spel zijn, dat ick [491] tegen 2 andere speele, insettende ieder van ons drien *x*, ende laet ick met den eenen dese voorwaerde maecken, dat soo hy het spel wint hy my sal geven *b*, ende ick *b* aen hem, soo ick het kome te winnen. Met den anderen laet ick dese voorwaerde maecken, dat hy het spel winnende my sal geven *c*, of ick aen hem *c* als ick het win. Het blijckt dat dit spel rechtmaetig is. Ende ick sal daer door gelijcke kans hebben, om *b* te hebben, te weeten, als het den eersten wint, of *c*, als het den tweeden wint, of $3x - b - c$ als ick het win; want dan treck ick $3x$ die ingeset zijn, en geve daer van aen den eenen *b*, aen den anderen *c*. Indien nu $3x - b - c$ gelijck waer aen a, so soude ick gelijcke kans hebben tot *a* of *b* of *c*. So stel ick dan $3x - b - c = a$ en komt $x=(a+b+c)/3$, voor de waerde van mijn kans. Op gelijcke manier werdt gevonden, dat als ick gelijcke kans hebbe tot *a* of *b* of *c* of *d*, dit soo veel weerdt is als $(a+b+c+d)/4$, ende soo voorts.

Development of probabilistic models

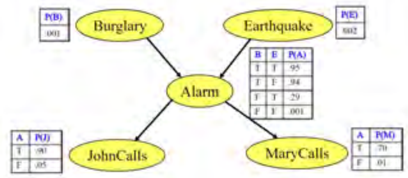




$$S \to \overset{0.1}{aSu} \mid \overset{0.4}{aS} \mid \overset{0.4}{Su} \mid \overset{0.1}{\epsilon}$$

$$S \overset{0.1}{\Rightarrow} aSu \overset{0.1}{\Rightarrow} aaSuu \overset{0.1}{\Rightarrow} aauu \qquad 0.001$$

$$S \overset{0.4}{\Rightarrow} aS \overset{0.4}{\Rightarrow} aaS \overset{0.4}{\Rightarrow} aaSu \overset{0.4}{\Rightarrow} aaSuu \overset{0.1}{\Rightarrow} aauu \qquad 0.00256$$



The origin of programming

Ada Lovelace

Sketch of
The Analytical Engine
Invented by Charles Babbage

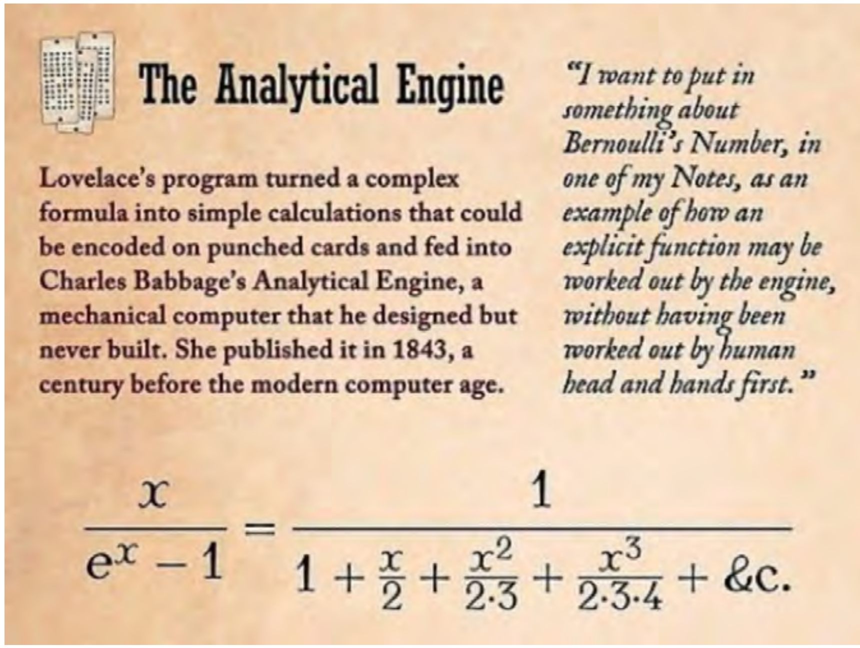
By L. F. MENABREA
of Turin, Officer of the Military Engineers

from the *Bibliothèque Universelle de Genève*, October, 1842, No. 82

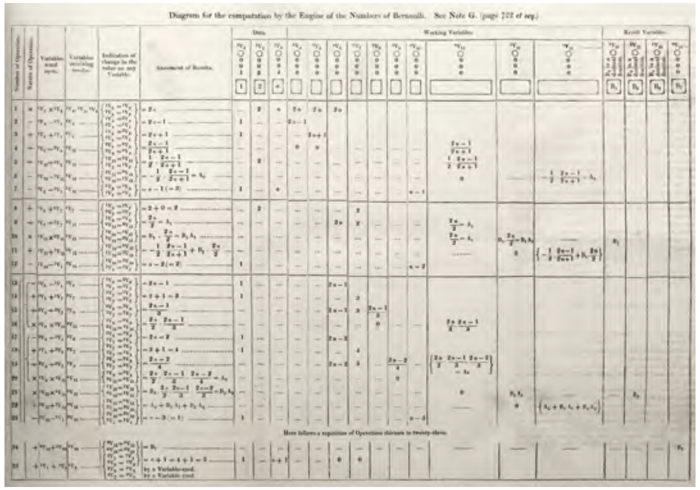
With notes upon the Memoir by the Translator
ADA AUGUSTA, COUNTESS OF LOVELACE

Translated by Ada Lovelace, 1836[a]

[a] Andrew L. Russell and Robin Hammerman, Ada'a Legacy: Cultures of Computing from the Victorian to the Digital Age, 2015.

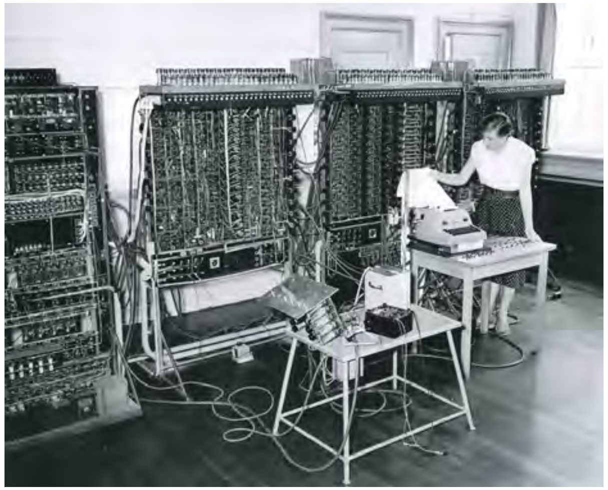


How "programs" looked like



Lovelace's program to compute Bernoulli numbers

The first Dutch computer



Automatische Relais Rekenmachine Amsterdam-1 (ARRA-1), 1952

Its first program

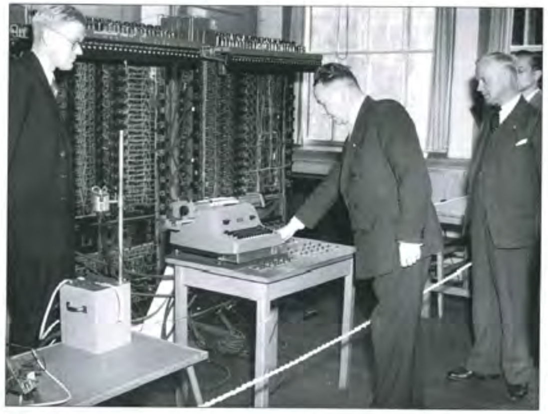
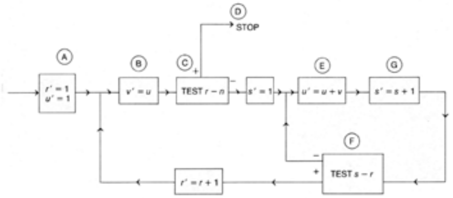


Foto van het ANP van de officiële ingebruikname van de ARRA op 21 juni 1952. Links B.J. (Bram Jan) Loopstra, in het midden minister van Onderwijs, Kunsten en Wetenschap, professor F.J.Th. Rutten, en rechts de burgemeester van Amsterdam, mr. A.J. d'Ailly (fotocollectie E. Verhagen)

. . . . . . was a probabilistic program, 1952[1]

[1] Erik Verhagen. Geheugentrommels. 2016

Historical perspective




```
         00000000 00000100 0000000000000000
01011110 00001100 11000010 0000000000000010
         11101111 00010110 0000000000000101
         11101111 10011110 0000000000001011
11111000 10101101 11011111 0000000000010010
         01100010 11011111 0000000000010101
11101111 00000010 11111011 0000000000010111
11110100 10101101 11011111 0000000000011110
00000011 10100010 11011111 0000000000100001
11101111 00000010 11111011 0000000000100100
01111110 11110100 10101101
11111000 10101110 11000101 0000000000101011
00000110 10100010 11111011 0000000000110001
11101111 00000010 11111011 0000000000110100
         01010000 11010100 0000000000111011
                  00000100 0000000000111101
```

Code

```
add KEY_ZL r3
adc r2
add KEY_ZH r2

; Z2 = (y + sum)

mov Y_HIGH r0
mov Y_LOW r1

add SUM_L r1
adc r0
add SUM_H r0

; Z4 = Z1 Xor Z2

xor r0 r2
xor r1 r3

; Z3 = ((y>>5) + key[3])

mov Y_HIGH r0
mov Y_LOW r1

; right shift = sign bit is replicated

ashc #-5 r0

; use BIC to clear unwanted replication
bic #174000 r0

add KEY_3L r1
```

```java
package com.beginnersbook;
import java.util.Scanner;
class JavaExample {

    public static double squareRoot(int number) {
        double temp;

        double sr = number / 2;

        do {
            temp = sr;
            sr = (temp + (number / temp)) / 2;
        } while ((temp - sr) != 0);

        return sr;
    }

    public static void main(String[] args)
    {
        System.out.print("Enter any number:");
        Scanner scanner = new Scanner(System.in);
        int num = scanner.nextInt();
        scanner.close();
```

Rethinking the Bayesian approach

"In particular, the graphical model formalism that ushered in an era of rapid progress in AI has proven inadequate in the face of [these] new challenges.

A promising new approach that aims to bridge this gap is probabilistic programming, which marries probability theory, statistics and programming languages"

Daniel Roy, Computability, Inference and Modeling in Probabilistic Programming. 2011[2]

[2] MIT/EECS George M. Sprowls Doctoral Dissertation Award

Overview

1. Introduction
2. Historical perspective
3. What is probabilistic programming?
4. Two striking examples
5. Why are probabilistic programs intricate?
6. What are we going to do in this course?
7. What do we expect from you?

What is probabilistic programming good for?

"The goal of probabilistic programming is
to enable probabilistic modeling and machine learning
to be accessible to the working programmer"

Andy Gordon

Tom Henzinger

Aditya Nori

Sriram Rajamani

What is probabilistic programming good for?

"Why probabilistic programs?
To provide a clear and high-level,
but complete, language
for specifying complex [probabilistic] models"

UBER

The PPL landscape

Python	PyMC, Edward, TFP
Javascript	webPPL
C++	STAN
Scala	Figaro
C	Probabilistic-C
Excel	Fabular
R	R2

Every programming language has a probabilistic variant

Probabilistic programming

Probabilistic program:

random sampling, conditioning, and usual control-flow constructs

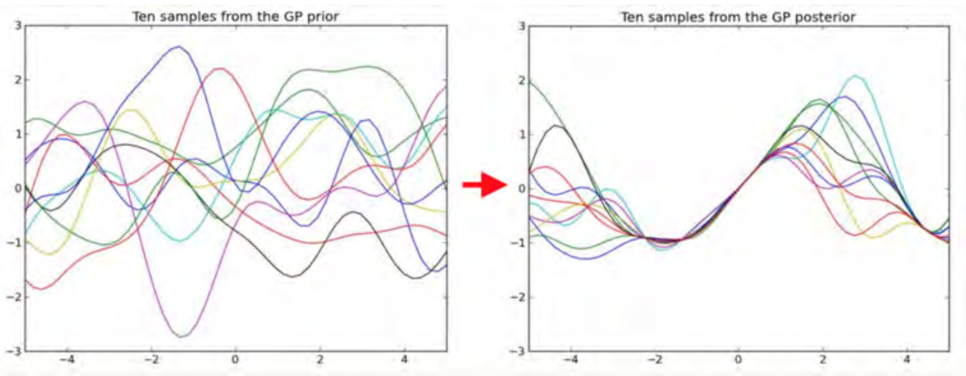
Analysis:

sophisticated Markov chain Monte Carlo simulation approaches[3]

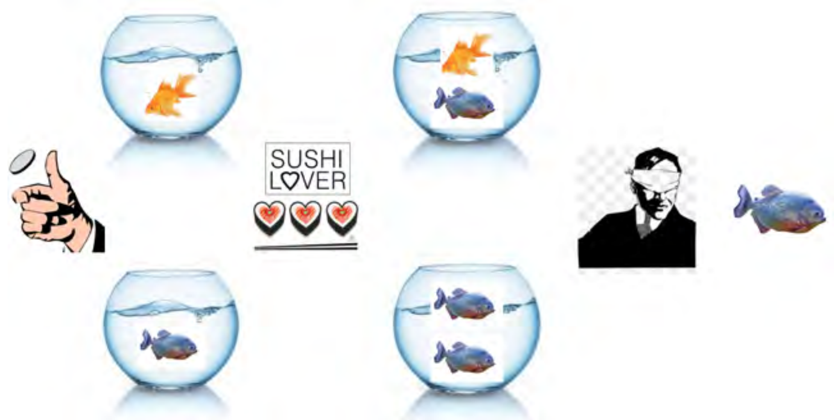
It is a young field: first PROBPROG conference in 2018

[3] typically based on rejection sampling and random walks

Conditioning = learning

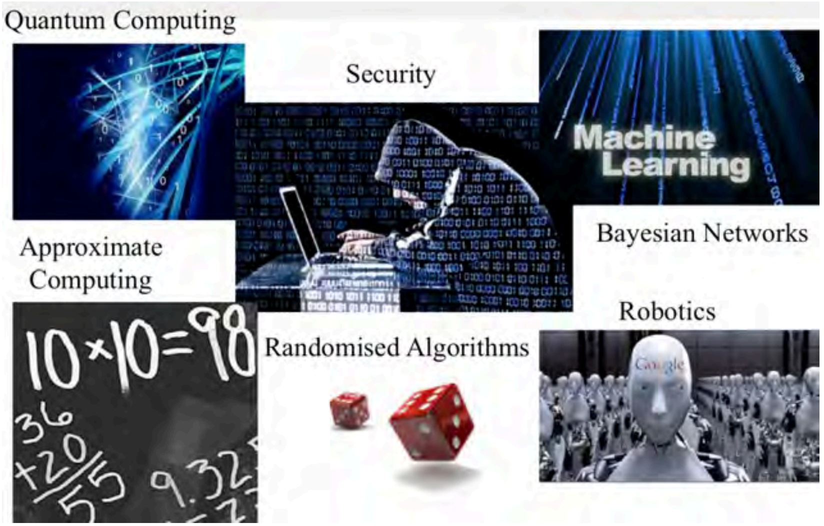


Bayesian learning



Q: What is the probability that initially there was a piranha? A: 2/3

Applications



Randomised algorithms

- What? Some decisions are based on coin flips
- Why?
  - Their conceptual simplicity
  - Their speed
    - mostly faster than their deterministic counterpart
    - no particular input elicits worst-case behaviour
  - Their existence
    - many solve problems that have no deterministic solution
- Types
  1. Las Vegas: always produces correct results, random runtime
  2. Monte Carlo: may produce wrong results, deterministic runtime

Famous randomised algorithms

- Randomised Quicksort (196X)
- Rabin-Miller's Primality Test (1980)
- Freivald's Matrix Multiplication (1977)
- Lehmann-Rabin's Randomised Mutual Exclusion (1981)
- Herman's Randomised Self-Stabilising Algorithm (1990)
- The Coupon's Collector Algorithm
- Skip Lists (1990) and Bloom Filters (1970)
- ......

Las Vegas: Sorting by flipping coins

Quicksort:

```
QS(A) =
  if |A| <= 1 { return A; }
  i := ceil(|A|/2);
  A< := {a in A | a < A[i]};
  A> := {a in A | a > A[i]};
  return QS(A<) ++ A[i] ++ QS(A>)
```

Worst case complexity:
$O(N^2)$ comparisons

Randomised Quicksort:

```
rQS(A) =
  if |A| <= 1 { return A; }
  i := Unif[1...|A|];
  A< := {a in A | a < A[i]};
  A> := {a in A | a > A[i]};
  return rQS(A<) ++ A[i] ++ rQS(A>)
```

Worst case complexity:
$O(N \log N)$ expected comparisons

Monte Carlo: Matrix multiplication

Input: three N^2 square matrices A, B, and C

Output: yes, if $A \cdot B = C$; no, otherwise

Time complexity over the years:

- until end 1960s: cubic (= 3)
- 1969: 2.808
- 1978: 2.796
- 1979: 2.780
- 1981: 2.522
- 1984: 2.496
- 1989: 2.376
- 2014: 2.373
- 2100: ......

Monte Carlo: Freivald's matrix multiplication

Input: three $\mathcal{O}(N^2)$ square matrices A, B, and C

Output: yes, if $A \times B = C$; no, otherwise

Deterministic: compute $A \times B$ and compare with C

Complexity: in $\mathcal{O}(N^3)$, best known complexity $\mathcal{O}(N^{2.37})$

Randomised:
1. take a random bit-vector $\vec{x}$ of size N
2. compute $A \times (B\,\vec{x}) - C\,\vec{x}$
3. output yes if this yields the null vector; no otherwise
4. repeat these steps k times

Complexity: in $\mathcal{O}(k \cdot N^2)$, with false positive with probability $\leqslant 2^{-k}$

Security

Turing Award Winners 2013

"Goldwasser and Micali proved (1982) that encryption schemes must be **random** rather than deterministic [...] an insight that revolutionised the study of encryption and laid the foundation for the theory of cryptographic security."

used in almost all communication protocols, Internet transactions and cloud computing

A Birthday Attack

Let crypto-hash function $h : \{0,1\}^m \to \{0,1\}^n$ with $Pr[h(x) = y] = 2^{-n}$

```
b := true;
i := 1;
while (b) {
  for j = 1 to m do // sample m bits uniformly
     xi[j] := uniform(1) // the j-th bit of xi
  for k = 1 to i-1 do
     b := b and h(xi) != h(xk)
  od;
  i++
}
```

How many iterations are needed to terminate with probability > 1/2?

[Katz & Lindall, Intro to Modern Cryptography, 2014]

The famous RSA-OAEP protocol

Oracle $\mathsf{Enc}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $f_{pk}(s \| t)$

Oracle $\mathsf{Dec}_{sk}(c)$:
$(s,t) \leftarrow f_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $[s \oplus G(r)]_{k_1} = 0^{k_1}$ then return $[s \oplus G(r)]^n$
else return $\perp$

Oracle $G(x)$:
if $x \notin \mathrm{dom}(L_G)$ then $L_G[x] \xleftarrow{\$} \{0,1\}^{n+k_1}$;
return $L_G[x]$

Oracle $H(x)$:
if $x \notin \mathrm{dom}(L_H)$ then $L_H[x] \xleftarrow{\$} \{0,1\}^{k_0}$;
return $L_H[x]$

Game IND-CCA2 :
$(sk, pk) \leftarrow \mathcal{KG}(\,)$;
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^* \leftarrow \mathsf{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma)$;
return $b = b'$

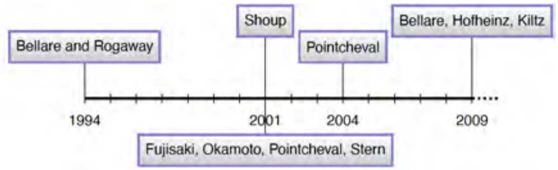
Game POW :
$(sk, pk) \leftarrow \mathcal{KG}()$;
$y \xleftarrow{\$} \{0,1\}^{n+k_1}$;
$z \xleftarrow{\$} \{0,1\}^{k_0}$;
$y' \leftarrow \mathcal{I}(f_{pk}(y \| z))$;
return $y = y'$

42

Correctness

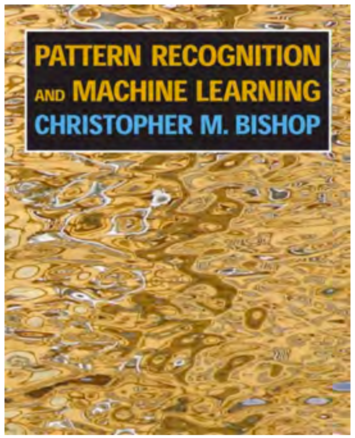
For every IND-CCA2 adversary A there exists an inverter I for which

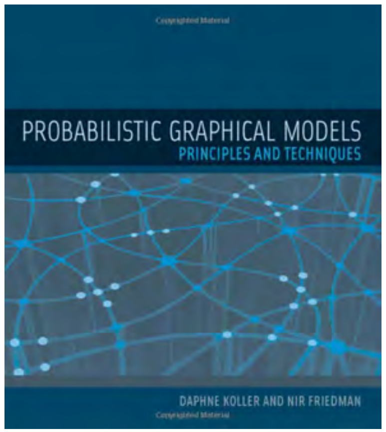
$$\left| \Pr_{\text{IND-CCA2}}[b = b'] - 1/2 \right| \leq \underbrace{\Pr_{\text{POW}}[y = y']}_{\text{depends on } I} + \ldots$$



It took about 20 years to prove a tight upper bound
Proof-checked by Barthe *et. al.* (2012)

Probabilistic graphical models



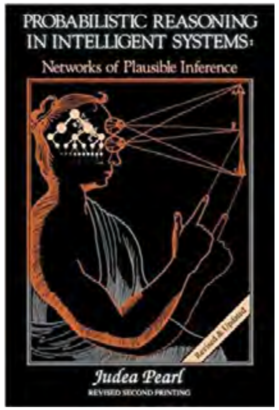


The importance of Bayesian networks

"Bayesian networks are as important to AI and machine learning as Boolean circuits are to computer science."

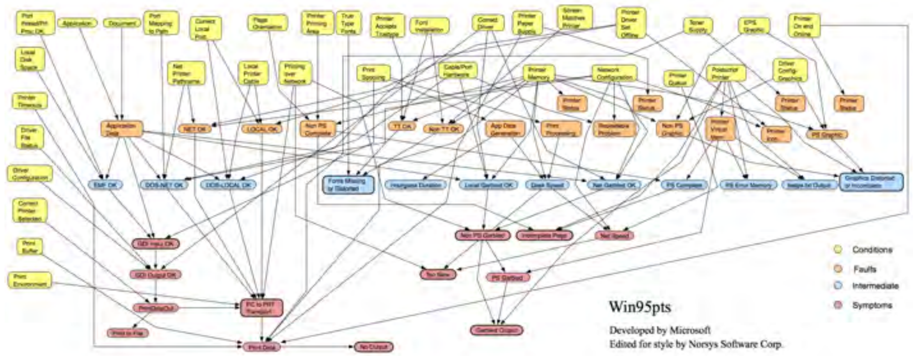
[Stuart Russell (Univ. of California, Berkeley), 2009]

Judea Pearl: The father of Bayesian networks



Turing Award 2011: "for fundamental contributions to AI through the development of a calculus for probabilistic and causal reasoning".

Printer troubleshooting in Windows 95



How likely is it that your print is garbled **given that** the ps-file is not and the page orientation is portrait?

[Ramanna *et al.*, Emerging Paradigms in Machine Learning, 2013]

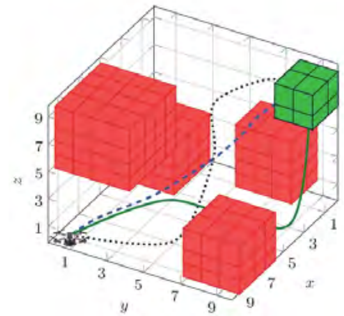
Cognitive science

Computational models that perform probabilistic inference over hierarchies of flexibly structured representations can address some of the deepest questions about the nature and origins of human thought.

[Tenenbaum, Kemp, Griffiths, Goodman, Science, 2011]

Probabilistic programming provides such modeling and inference

Planning in AI: Robot navigation



Uncertainty: noisy sensors and actuators, unknown environment[4]

[4]Evans *et al.*, Modeling Agents with Probabilistic Programs, 2019

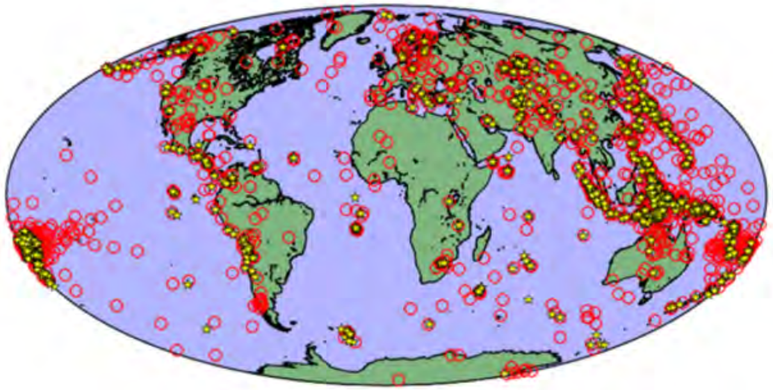
Overview

The seismic localization problem

1996: Comprehensive Nuclear-Test-Ban Treaty (by 2019: 184 countries signed)

International Monitoring System

Event location UN system



It is not that easy

Problems:

- UN system misses nearly one-third of all seismic events of interest
- About 50% of the reported events are spurious
- Large team of experts analyse automatic bulletins to improve accuracy

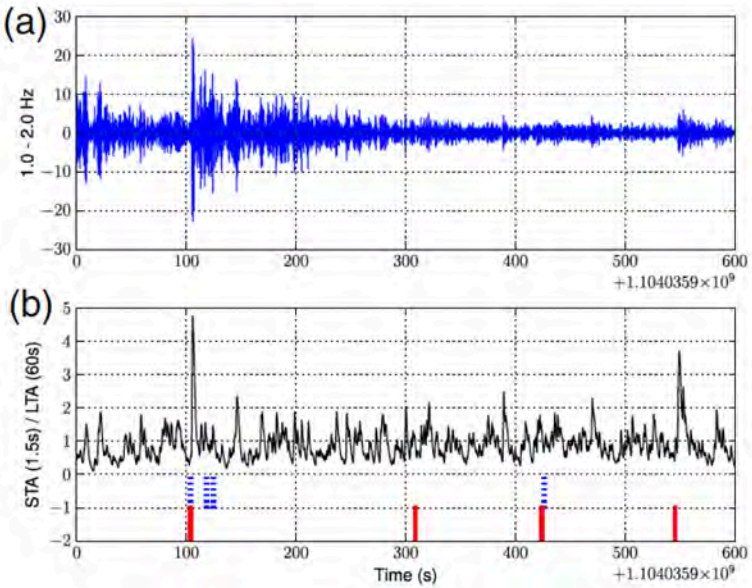
Sources of difficulty: Many sources of uncertainty

- Travel time, attenuation of frequencies on Earth not accurately known
- Detectors are noisy (up to 90% of detections are false)
- Huge number of detections recorded per day

Use a probabilistic generative model. In fact: a probabilistic program.

An infrasound station

Measurements at one station



Key question

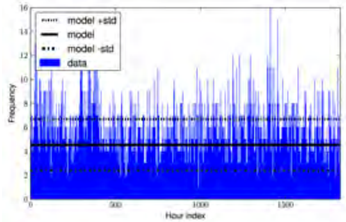
Recall Bayes' rule: $Pr(X \mid Y = y) \;=\; \dfrac{Pr_\theta(X) \cdot Pr_\phi(Y = y \mid X)}{Pr(Y = y)}$

To determine:

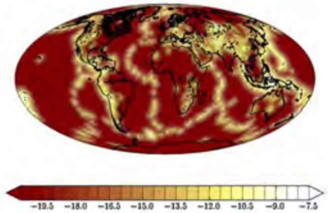
$$x^* \;=\; \arg\max_x \; \underbrace{Pr_\theta(X = x)}_{\text{prior over events}} \;\cdot\; \underbrace{Pr_\phi(Y = y \mid X = x)}_{\text{how events propagate and are measured}}$$

What is the most likely explanation for all sensor readings?

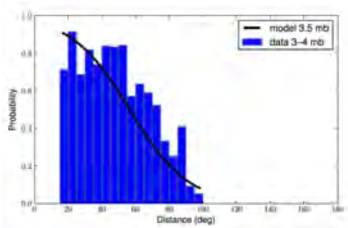
The priors



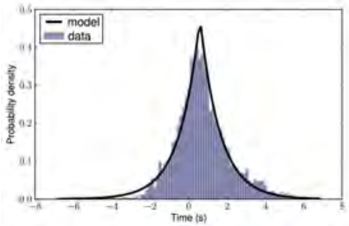
Event rate distribution (λ_e = 4.6 per hour)



Event location log density



Detection probability



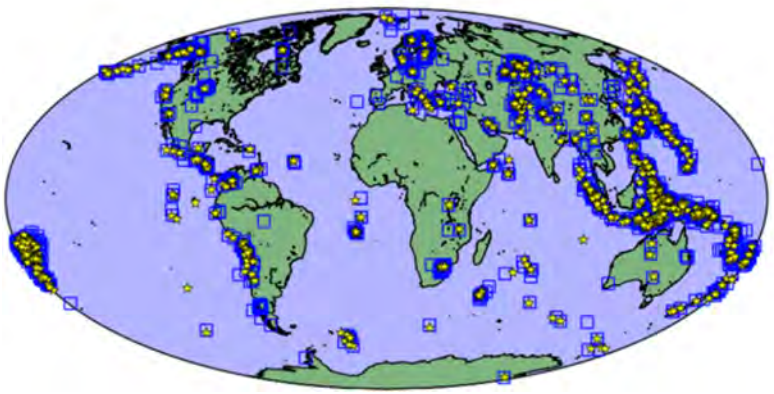
Arrival times of P-phase arrivals

A probabilistic program snippet in BLOG

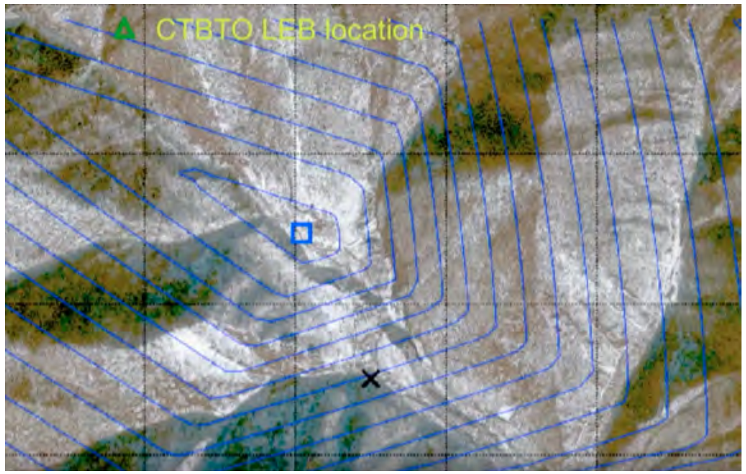
```
#SeismicEvents ~ Poisson[T*λe];
Time(e) ~ Uniform(0,T)
IsEarthQuake(e) ~ Bernoulli(.999);
Location(e) ~ if IsEarthQuake(e) then SpatialPrior() else UniformEarthDistribution();
Depth(e) ~ if IsEarthQuake(e) then Uniform[0,700] else 0;
Magnitude(e) ~ Exponential(log(10));
IsDetected(e,p,s) ~ Logistic[weights(s,p)](Magnitude(e), Depth(e), Distance(e,s));
#Detections(site = s) ~ Poisson[T*λf(s)];
#Detections(event=e, phase=p, station=s) = if IsDetected(e,p,s) then 1 else 0;
OnsetTime(a,s) ~ if (event(a) = null) then Uniform[0,T] else
  Time(event(a)) + GeoTravelTime(Distance(event(a),s),Depth(event(a)),phase(a))
                 + Laplace(μt(s), σt(s))
Amplitude(a,s) ~ If (event(a) = null) then NoiseAmplitudeDistribution(s)
    else AmplitudeModel(Magnitude(event(a)), Distance(event(a),s),Depth(event(a)),phase(a))
Azimuth(a,s) ~ If (event(a) = null) then Uniform(0, 360)
    else GeoAzimuth(Location(event(a)),Depth(event(a)),phase(a),Site(s)) + Laplace(0,σa(s))
Slowness(a,s) ~ If (event(a) = null) then Uniform(0,20)
    else GeoSlowness(Location(event(a)),Depth(event(a)),phase(a),Site(s)) + Laplace(0,σa(s))
ObservedPhase(a,s) ~ CategoricalPhaseModel(phase(a))
```

Arora, Russell, and Sudderth, Bull. of Seismological Society of America, 2013

Event location revamped



A  nuclear test



February 12, 2013

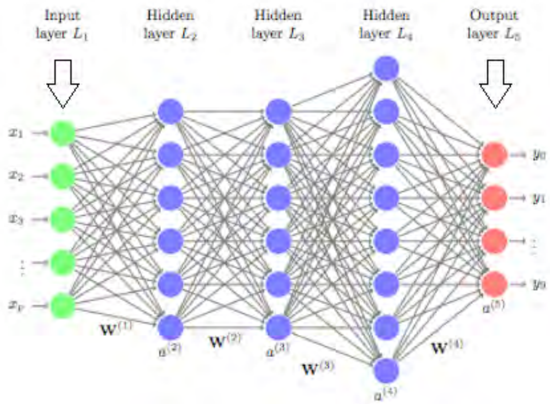
Main findings

- 60% reduction of missed events compared to UN system
  No increase of the number of false alarms
- Found events missed by human analysis of UN system output
- 2014: UN announces to use the NETVISA[5] system
- 2018: UN replaces their system by NETVISA

Since 2018, earthquakes and explosions (conventional and nuclear) are located and diagnosed by a probabilistic program[6]

[5] Network Processing Vertically Integrated Seismic Analysis
[6] Mitchell Prize of American Statistical Assoc. & Int. Society for Bayesian Analysis

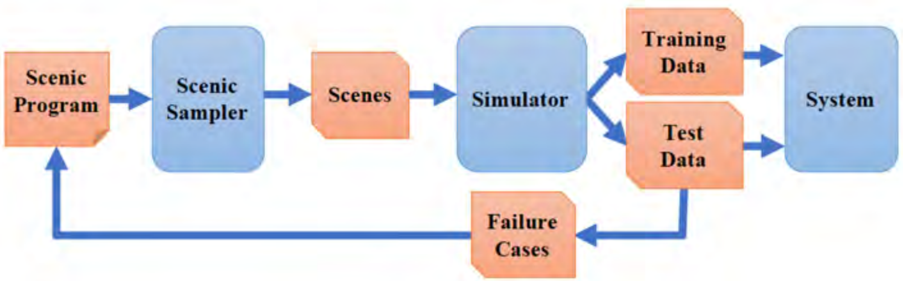
Synthetic training data for deep learning



- Collecting real-world data is slow and costly: preprocessing+labelling
- Hard to observe and reproduce rare corner cases (e.g., a car accident)

The SCENIC language

Aim: Use probabilistic programs to generate synthetic training data.[7]



[7] Fremont, Sangiovanni-Vincentelli, Seshia, *et al.*, PLDI 2019

SCENIC example: A badly parked car

Figure 4: A scene of a badly parked car.

```
spot = OrientedPoint on visible curb
badAngle = Uniform(1.0, -1.0) * (10, 20) deg
Car left of (spot offset by -0.5 @ 0), \
    facing badAngle relative to roadDirection
```

A snippet of SCENIC's syntax

```
    scenario := (import file)* (statement)*
     boolean := True | False | booleanOperator
      scalar := number | distribution | scalarOperator
distribution := baseDist | resample(distribution)
      vector := scalar @ scalar | Point | vectorOperator
     heading := scalar | OrientedPoint | headingOperator
   direction := heading | vectorField
       value := boolean | scalar | vector | direction
              | region | instance | instance.property
   classDefn := class class[(superclass)]:
                    (property: defaultValue)*
    instance := class specifier, ...
   specifier := with property value | posSpec | headSpec
```

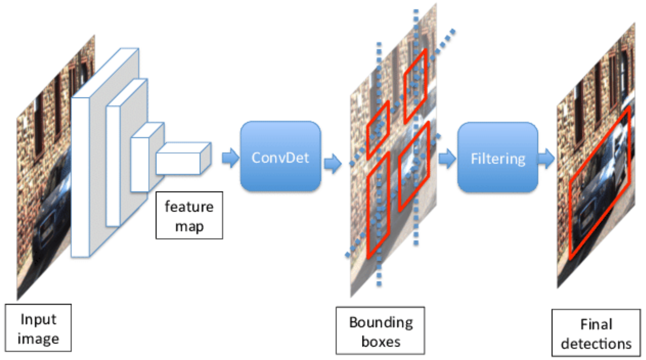
Syntax	Distribution
(*low*, *high*)	uniform on interval of $\mathbb{R}$
Uniform(*value*, ...)	uniform over given values
Discrete({*value*: *wt*, ...})	discrete with given weights
Normal(*mean*, *stdDev*)	normal with given μ, σ

Plus conditionals, loops, functions, methods and so on.

Example scenes

Three scenarios generated from a ± 20-line SCENIC program

SqueezeDet: Small, fast, and energy efficient



Convolutional NN for real-time object detection in autonomous driving[8]

[8] Wu, Iandola, Jin & Keutzer, CVPR 2017

Baseline: Grand Theft Auto V

Hard SCENIC test case: Background error

Hard SCENIC test case: Missed object

Precision and recall

Suppose a neural network for recognizing dogs in photographs identifies 8 dogs in a picture containing 12 dogs and some cats.

Of the 8 identified as dogs, 5 actually are dogs (true positives), while the rest are cats (false positives).

The program's precision is 5/8 while its recall is 5/12.

Precision is "how useful the vision recognition results are", and

Recall is "how complete the results are".

High precision: substantially more relevant results than irrelevant ones.

High recall: algorithm returned most of the relevant results.

Main findings

SCENIC enables generating more effective training sets

$$\text{precision} = \frac{\textit{true pos}}{\textit{true pos} + \textit{false pos}} \quad \text{and} \quad \text{recall} = \frac{\textit{true pos}}{\textit{true pos} + \textit{false neg}}$$

Training set	Baseline test-set[9]		"Hard" SCENIC test-set	
	precision	recall	precision	recall
100/0	72.9 ± 3.7	37.1 ± 2.1	62.8 ± 6.1	65.7 ± 4.0
95/5	73.1 ± 2.3	37.0 ± 0.6	68.9 ± 3.2	67.3 ± 2.4

Test set results trained on 5,000 images from
100% screen shots or mixed with 5% "hard" SCENIC cases

[9] Screen dumps from playing the video game Grand Theft Auto V

Overview

Three elementary issues

1. Program correctness
2. Termination
3. The runtime of a program

Issue 1: Program correctness

- Classical programs:
  - A program is correct with respect to a (formal) specification

    "for input array A, the output array B is sorted and contains all elements contained in A"
  - Defines a deterministic input-output relation
  - Partial correctness: if an output is produced, it is correct
  - Total correctness: in addition, the program terminates
- Probabilistic programs:
  - They do not always generate the same output
  - They generate a probability distribution over possible outputs

Issue 2: Termination

- Classical programs:
  - They terminate (on a given/all inputs), or they do not
  - If they terminate, they take finitely many steps to do so
  - Showing program termination is undecidable (halting problem)
- Probabilistic programs:
  - They terminate (or not) with a certain likelihood
  - They may have diverging runs whose likelihood is zero
  - They may take infinitely many steps (on average) to terminate

    even if they terminate with probability one!
  - Showing "probability-one" termination is "more" undecidable
    - and showing they do in finite time on average, even more!

Issue 3: The program's runtime

- Classical programs:
  - They have a deterministic, fixed run-time for a given input
  - Runtimes of terminating programs in sequence are compositional:

    if P and Q terminate in n and k steps, then P;Q halts in $n+k$ steps
  - Analysis techniques: recurrence equations, tree analysis, etc.
- Probabilistic programs:
  - Every runtime has a probability; their runtime is a distribution
  - Runtimes of "probability-one" terminating programs may not sum up

    if P and Q terminate in n and k steps on average,
    then P;Q may need infinitely many steps on average
  - Analysis techniques: involve reasoning about expected values etc.

Overview

This course's topics

- The `webPPL` Language
- Markov Chains
- Markov Chain Monte Carlo
- pGCL: Syntax and Semantics
- Fixed Point Theory
- Weakest Preconditions
- Probabilistic WP
- Bounds and Invariants
- Conditioning
- Recursion Theory
- Termination
- Proving AST
- Expected Runtimes
- Bayesian Networks
- Probabilistic Databases
- Automated Program Analysis

Course material

- Lecture material = the slides + the lectures + recent papers
- Noah Goodman and Andreas Stuhlmüller:
  The Design and Implementation of Probabilistic Programming Languages, 2016. On-line available `dippl.org`
- Benjamin Kaminski:
  Advanced Weakest Precondition Calculi for Probabilistic Programs, 2019.

  Gilles Barthe, Joost-Pieter Katoen and Alexandra Silva (editors):
  Foundations of Probabilistic Programming, 2020.

  Annabelle McIver and Carroll Morgan:
  Abstraction, Refinement and Proof for Probabilistic Systems, 2005.

Lectures

- Tue 16:30–18:00 (AH3)
- Thu 16:30–18:00 (AH2)
- Start: Oct 11, 2022
- Two lectures per week
- End: early January 2023
- Available material per lecture:
  - Slides prior to the lecture
  - Annotated slides after the lecture

Website

https://moves.rwth-aachen.de/teaching/ws-22-23 and RWTH Moodle

Overview

Exercise classes

- You are expected to hand in homework exercises
  - mostly on a weekly basis
  - working in groups of three students
  - a few practical exercises; mostly theoretical exercises
  - solutions discussed at the exercise classes
- Start:
  - First exercise series: Fri Oct 15
  - First exercise class: Fri Oct 22 (AH1)
- Assistants:
  - **Lutz Klinkenberg**, **Philipp Schroer**, and **Tobias Winkler**

Examination

- Form: written exam
- Exam qualification: ≥ 50% of points in exercise series
- Registration for exam: Nov 15–Jan 15
- Exam dates:
  - **Feb 16, 2023, 15:00–17:00**
  - **Mar 18, 2023, 12:30–14:30**
- A hand-written “cheat sheet” of 1 double-sided A4 allowed
- How to raise your chances to pass the exam?
  - Exercise. Exercise. Exercise.

Next lecture

Thursday Oct 13, 16:30