Probabilistic Programming

Marius Popescu

popescunmarius@gmail.com

2020 - 2021

Introduction

What is Probabilistic Programming

Probabilistic programming languages aim to unify general purpose programming with probabilistic modeling; literally, users specify a probabilistic model in its entirety (e.g., by writing code that generates a sample from the joint distribution) and inference follows automatically given the specification

http://www.probabilistic-programming.org

Probabilistic programming is about performing Bayesian inference using the tools of computer science: programming language for model denotation and statistical inference algorithms for computing the conditional distribution of program inputs that could have given rise to the observed program output

A programming paradigm that provides a language based on random variables and stochastic control flow to construct a broad class of probabilistic models

(Bayesian) Probabilistic Modeling

Problem: given i.i.d. data $X = (x_1, x_2 \dots, x_n)$ from distribution $p(x|\theta)$ one needs to estimate θ

Frequentist framework: use maximum likelihood estimation (MLE)

$$\theta_{ML} = \operatorname{argmax} p(X|\theta) = \operatorname{argmax} \prod_{i=1}^{n} p(x_i|\theta) = \operatorname{argmax} \sum_{i=1}^{n} \log p(x_i|\theta)$$

Bayesian framework: encode uncertainty about θ in a *prior* $p(\theta)$ and apply Bayesian inference to find *the posterior*:

$$p(\theta|X) = \frac{\prod_{i=1}^{n} p(x_i|\theta)\, p(\theta)}{\int \prod_{i=1}^{n} p(x_i|\theta)\, p(\theta) d\theta}$$

Example: Coin Tossing

- We have a coin which may be fair or not
- The task is to estimate a probability θ of landing heads up
- Data: 2 tries with a result (H,H)

Example: Coin Tossing

- We have a coin which may be fair or not
- The task is to estimate a probability θ of landing heads up
- Data: 2 tries with a result (H,H)

Frequentist framework:

In all experiments the coin landed heads up
θ_{ML}=1

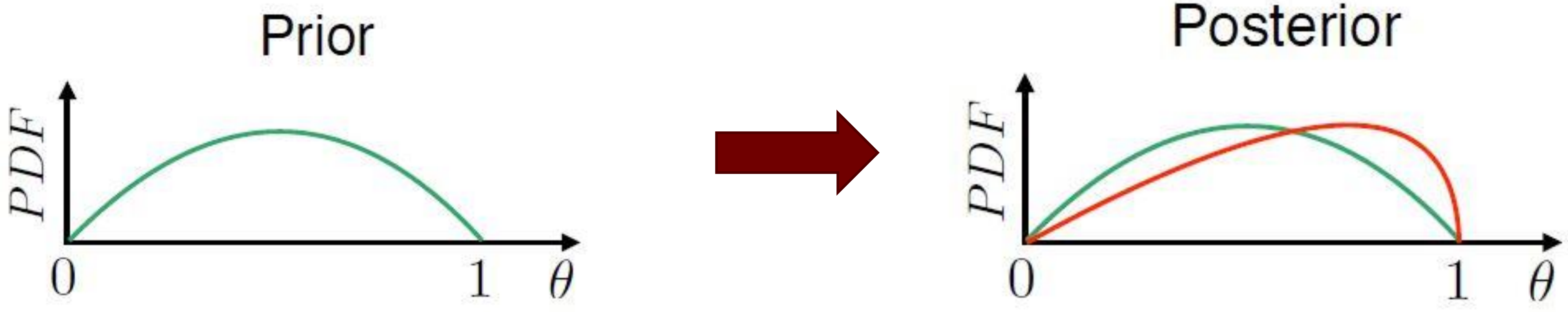
→

The coin is not fair and always lands heads up

Example: Coin Tossing

- We have a coin which may be fair or not
- The task is to estimate a probability θ of landing heads up
- Data: 2 tries with a result (H,H)

Bayesian framework:



Example: Coin Tossing

- We have a coin which may be fair or not
- The task is to estimate a probability θ of landing heads up
- Data: 1000 tries with a result of 489 tails and 511 heads

Both frameworks:

Sufficient amount of data matches our expectations → The coin is fair

Bayesian Modeling: The Problem

Bayesian framework: encode uncertainty about θ in a *prior* $p(\theta)$ and apply Bayesian inference to find *the posterior*:

$$p(\theta|X) = \frac{\prod_{i=1}^{n} p(x_i|\theta)\, p(\theta)}{\boxed{\int \prod_{i=1}^{n} p(x_i|\theta)\, p(\theta) d\theta}}$$

May be intractable

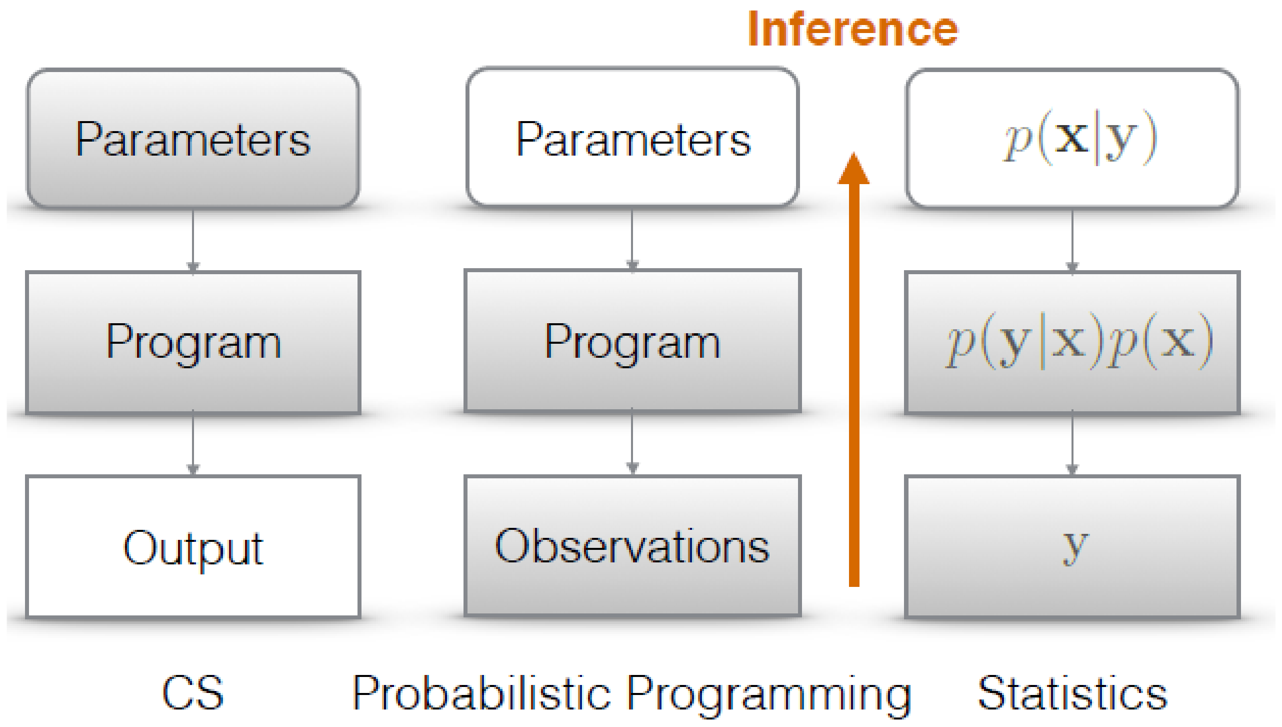
What is Probabilistic Programming

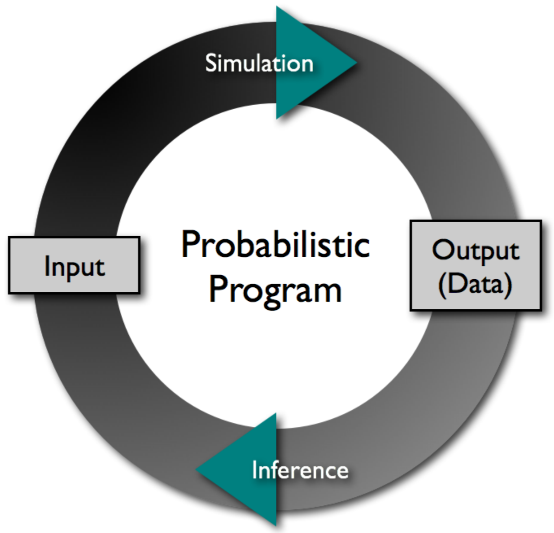
Probabilistic programming languages are to the *Bayesian or probabilistic machine learning* as automated differentiation tools are to the *deep learning*

The rapid grow of deep learning has been triggered largely by the emergence of programming language tools that automate the tedious and troublesome derivation and calculation of gradients for optimization

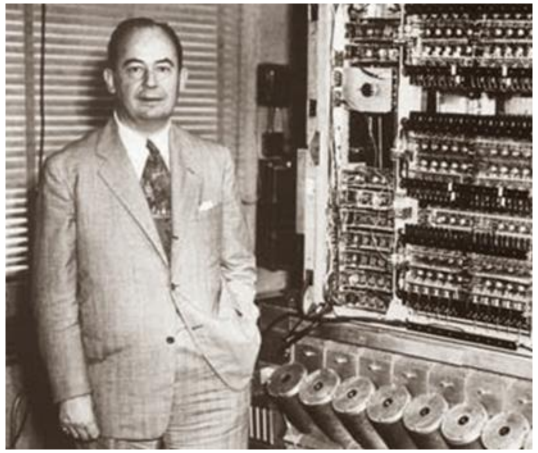
Probabilistic programming aims to build and deliver a toolchain that does the same for probabilistic machine learning

What is Probabilistic Programming


Inference
Parameters
Program
Output
Parameters
Program
Observations
$p(\mathbf{x}|\mathbf{y})$
$p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
$\mathbf{y}$
CS
Probabilistic Programming
Statistics



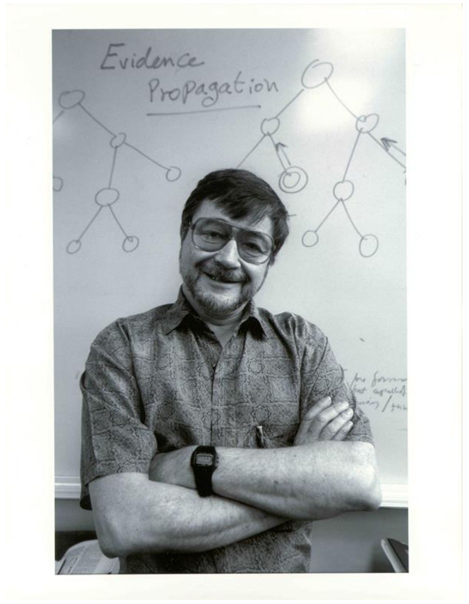
Some History

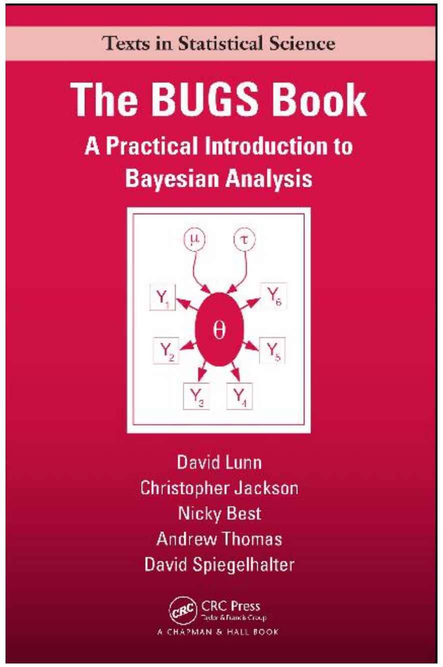


Von Neumann, J. (1956).
Probabilistic logics and the synthesis of reliable organisms from unreliable components

Shannon, C. E. (1958).
Von Neumann's contributions to automata theory.

Some History





Pearl, J. (1982).

“Reverend Bayes on inference engines: A distributed hierarchical approach,” Proceedings, AAAI-82

The BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling) project is concerned with flexible software for the Bayesian analysis of complex statistical models using Markov chain Monte Carlo (MCMC) methods. The project began in 1989 in the MRC Biostatistics Unit, Cambridge.

Is it useful? Is it hot?

Why to study probabilistic programming?

INSIGHT
Machine intelligence

nature

THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE

Back on its feet

Using an intelligent trial-and-error learning algorithm this robot adapts to injury in minutes

PAGES 426 & 503

COGNITION

WHY FISH NEED TO BE CLEVER

Social behaviours need plenty of brainpower

PAGE 412

ARTIFICIAL INTELLIGENCE

LIVING WITH ROBOTS

AI researchers' ethics prescriptions

PAGE 415

HUMAN EVOLUTION

ANOTHER FACE IN THE CROWD

A new hominin from Ethiopia's middle Pliocene

PAGES 432 & 483

NATURE.COM/NATURE

28 May 2015 £10

Vol. 521, No. 7553

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

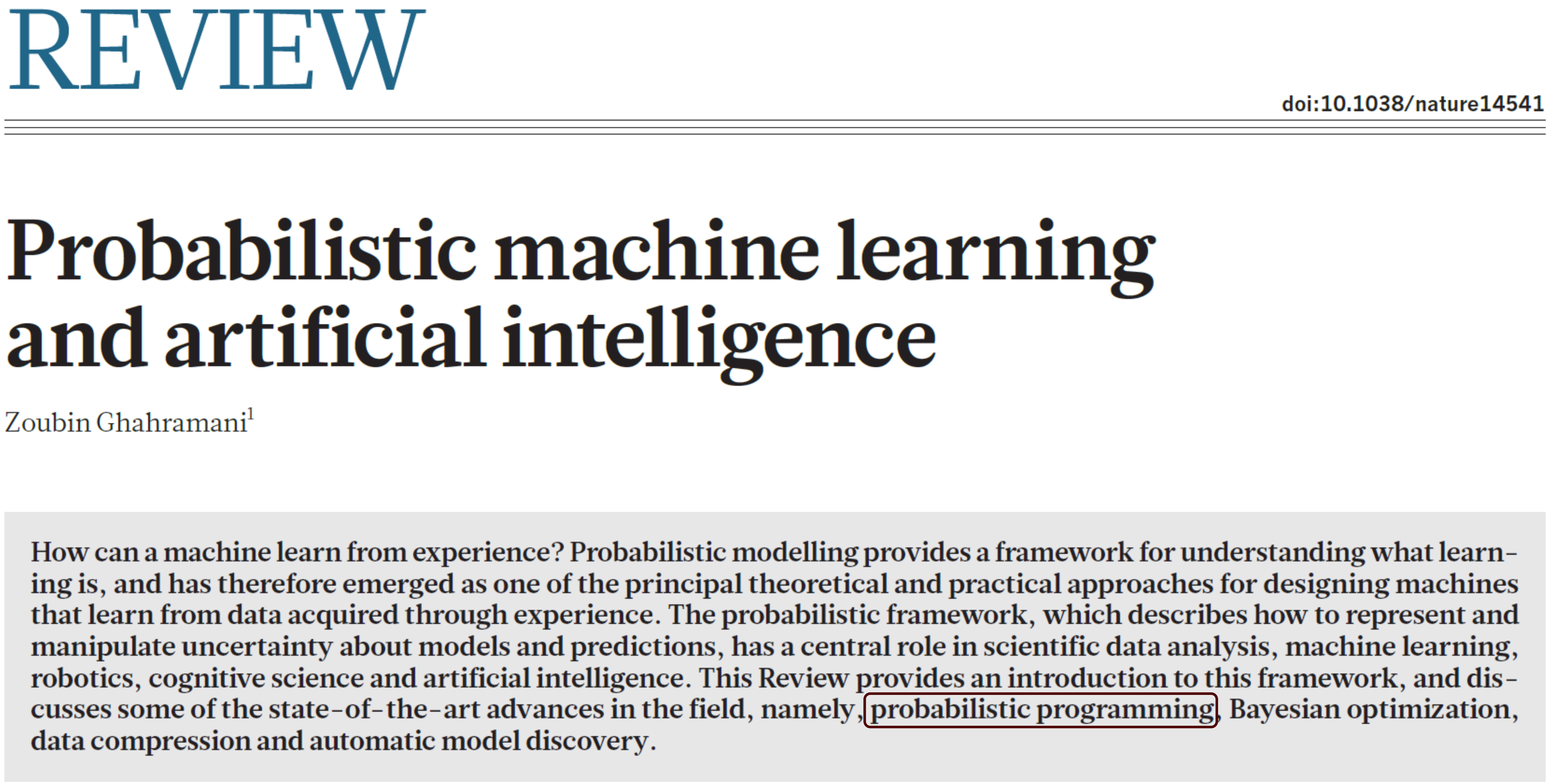
REVIEW

doi:10.1038/nature14540

Reinforcement learning improves behaviour from evaluative feedback

Michael L. Littman[1]

Reinforcement learning is a branch of machine learning concerned with using experience gained through interacting with the world and evaluative feedback to improve a system's ability to make behavioural decisions. It has been called the artificial intelligence problem in a microcosm because learning algorithms must act autonomously to perform well and achieve their goals. Partly driven by the increasing availability of rich data, recent years have seen exciting advances in the theory and practice of reinforcement learning, including developments in fundamental technical areas such as generalization, planning, exploration and empirical methodology, leading to increasing applicability to real-life problems.



REVIEW

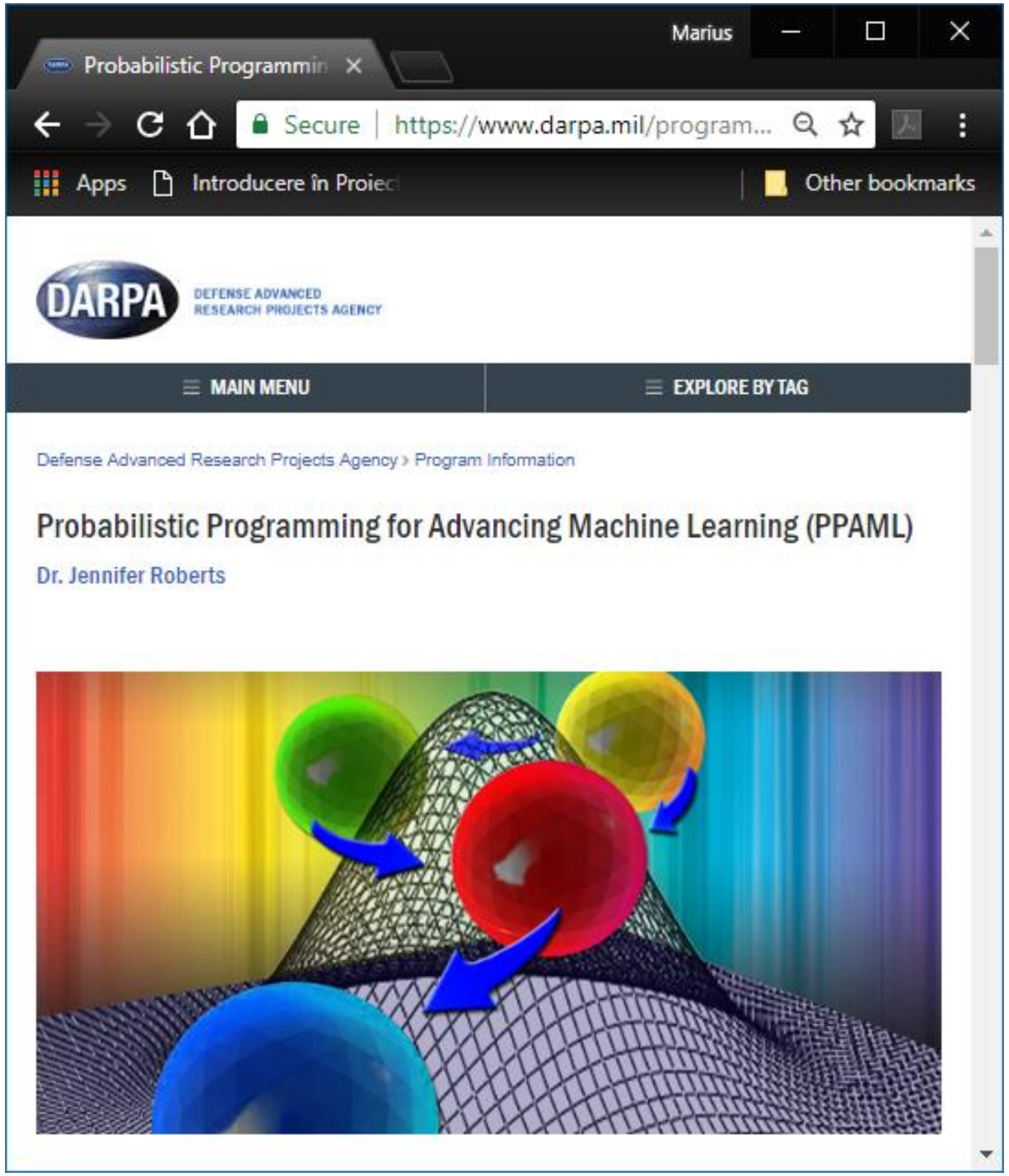
doi:10.1038/nature14541

Probabilistic machine learning and artificial intelligence

Zoubin Ghahramani[1]

How can a machine learn from experience? Probabilistic modelling provides a framework for understanding what learning is, and has therefore emerged as one of the principal theoretical and practical approaches for designing machines that learn from data acquired through experience. The probabilistic framework, which describes how to represent and manipulate uncertainty about models and predictions, has a central role in scientific data analysis, machine learning, robotics, cognitive science and artificial intelligence. This Review provides an introduction to this framework, and discusses some of the state-of-the-art advances in the field, namely, probabilistic programming, Bayesian optimization, data compression and automatic model discovery.

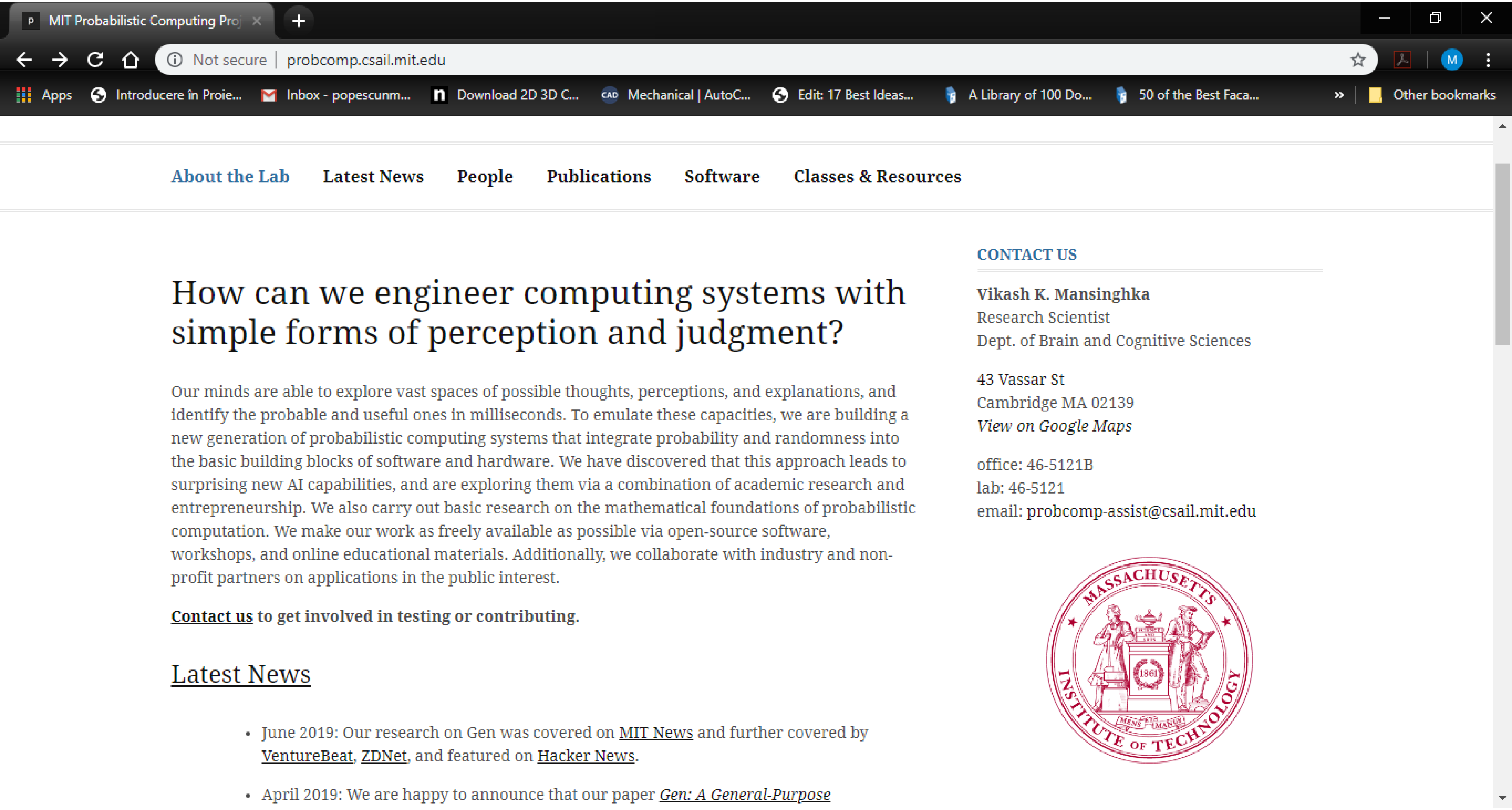
PPAML started in March 2013 and is scheduled to run 46 months, with three phases of activity through 2017



Defense Advanced Research Projects Agency › Program Information

Probabilistic Programming for Advancing Machine Learning (PPAML)

Dr. Jennifer Roberts



About the Lab   Latest News   People   Publications   Software   Classes & Resources

How can we engineer computing systems with simple forms of perception and judgment?

Our minds are able to explore vast spaces of possible thoughts, perceptions, and explanations, and identify the probable and useful ones in milliseconds. To emulate these capacities, we are building a new generation of probabilistic computing systems that integrate probability and randomness into the basic building blocks of software and hardware. We have discovered that this approach leads to surprising new AI capabilities, and are exploring them via a combination of academic research and entrepreneurship. We also carry out basic research on the mathematical foundations of probabilistic computation. We make our work as freely available as possible via open-source software, workshops, and online educational materials. Additionally, we collaborate with industry and non-profit partners on applications in the public interest.

Contact us to get involved in testing or contributing.

Latest News

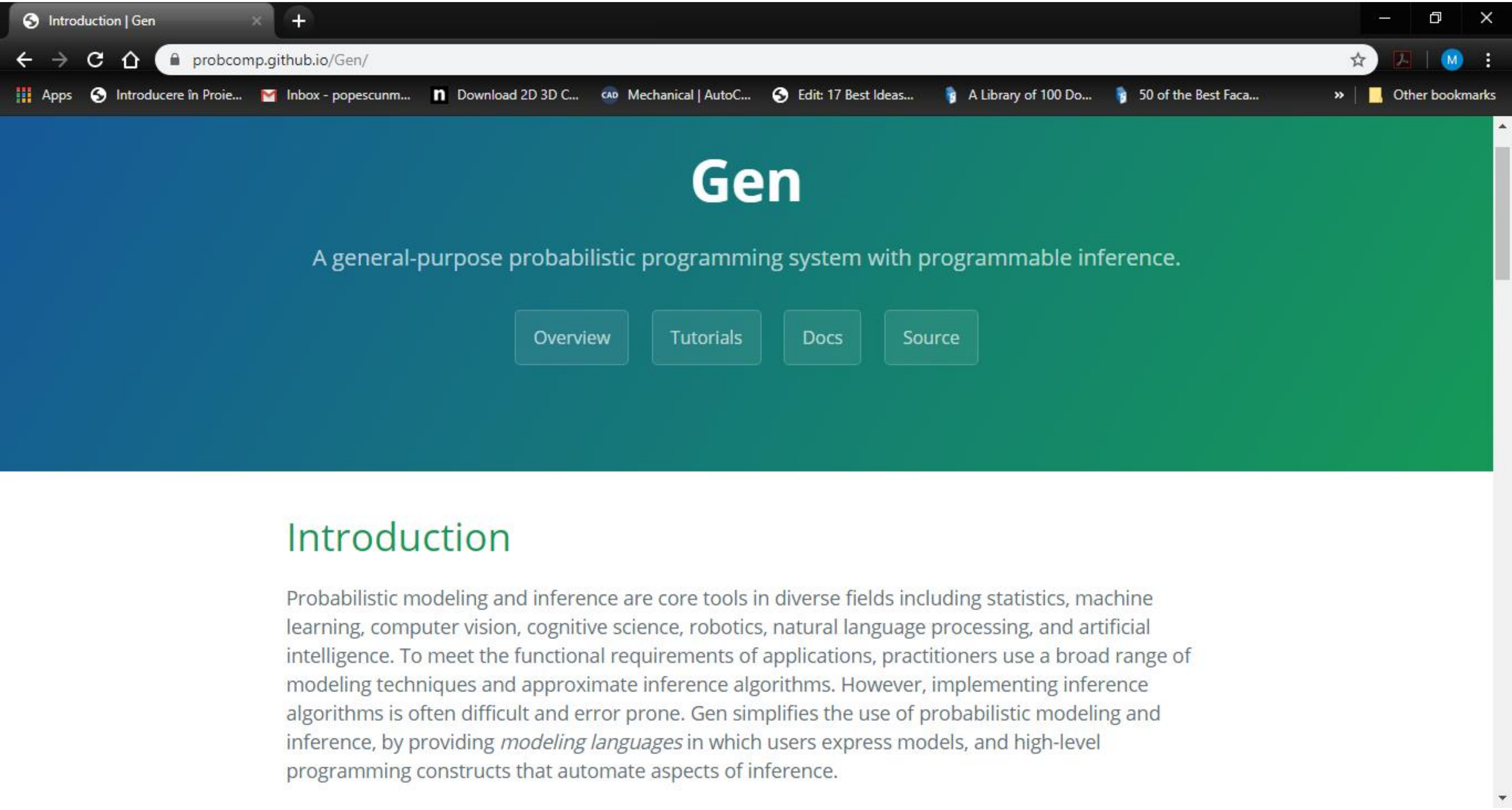
- June 2019: Our research on Gen was covered on MIT News and further covered by VentureBeat, ZDNet, and featured on Hacker News.
- April 2019: We are happy to announce that our paper *Gen: A General-Purpose*

CONTACT US

Vikash K. Mansinghka
Research Scientist
Dept. of Brain and Cognitive Sciences

43 Vassar St
Cambridge MA 02139
View on Google Maps

office: 46-5121B
lab: 46-5121
email: probcomp-assist@csail.mit.edu


Introduction | Gen

probcomp.github.io/Gen/

Apps · Introducere în Proie... · Inbox - popescunm... · Download 2D 3D C... · Mechanical | AutoC... · Edit: 17 Best Ideas... · A Library of 100 Do... · 50 of the Best Faca... · Other bookmarks

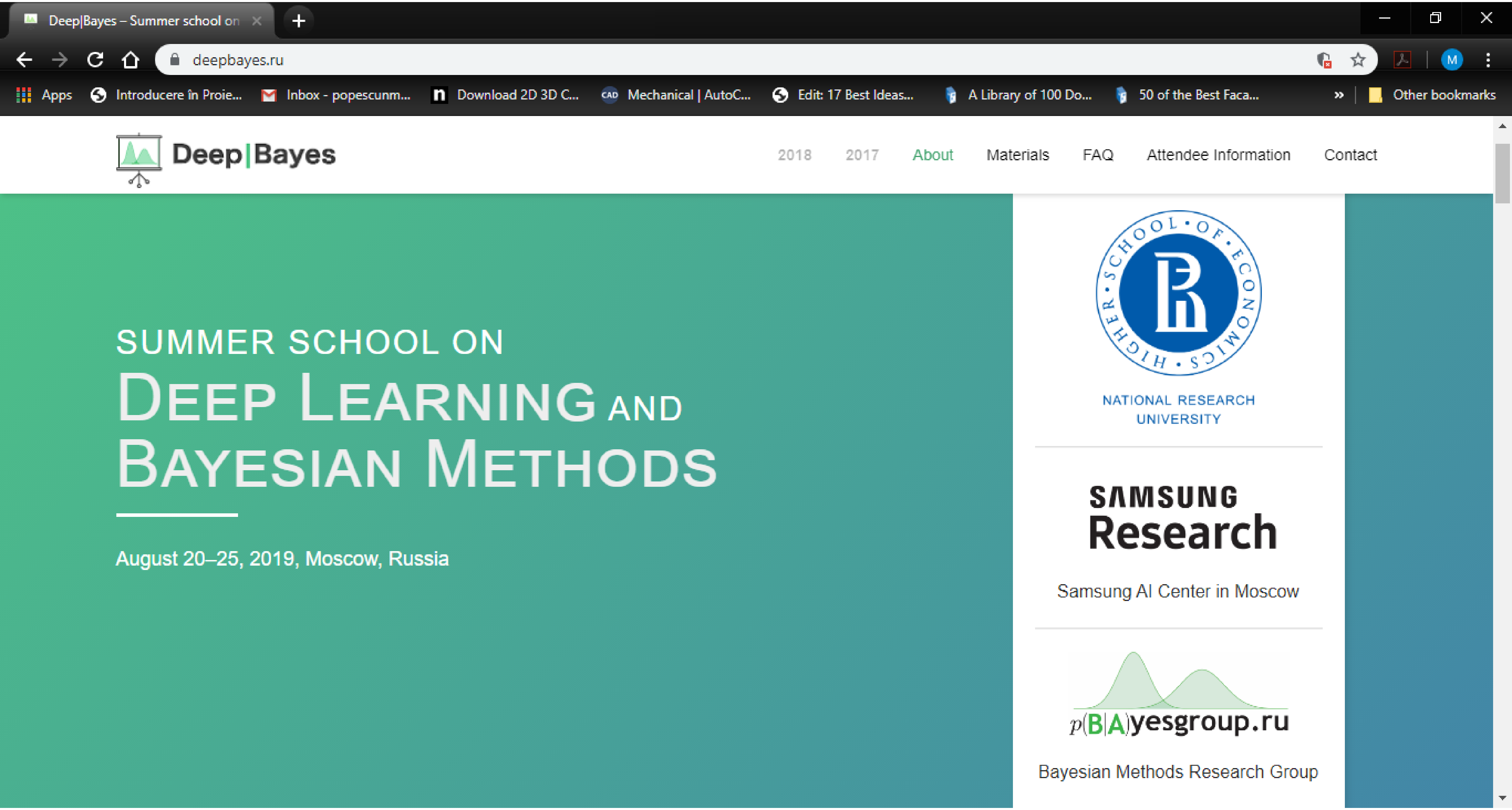
Gen

A general-purpose probabilistic programming system with programmable inference.

Overview · Tutorials · Docs · Source

Introduction

Probabilistic modeling and inference are core tools in diverse fields including statistics, machine learning, computer vision, cognitive science, robotics, natural language processing, and artificial intelligence. To meet the functional requirements of applications, practitioners use a broad range of modeling techniques and approximate inference algorithms. However, implementing inference algorithms is often difficult and error prone. Gen simplifies the use of probabilistic modeling and inference, by providing *modeling languages* in which users express models, and high-level programming constructs that automate aspects of inference.



SUMMER SCHOOL ON

DEEP LEARNING AND BAYESIAN METHODS

August 20–25, 2019, Moscow, Russia

SCHOOL OF ECONOMICS HIGHER
NATIONAL RESEARCH UNIVERSITY
SAMSUNG Research
Samsung AI Center in Moscow
p(B|A)yesgroup.ru
Bayesian Methods Research Group

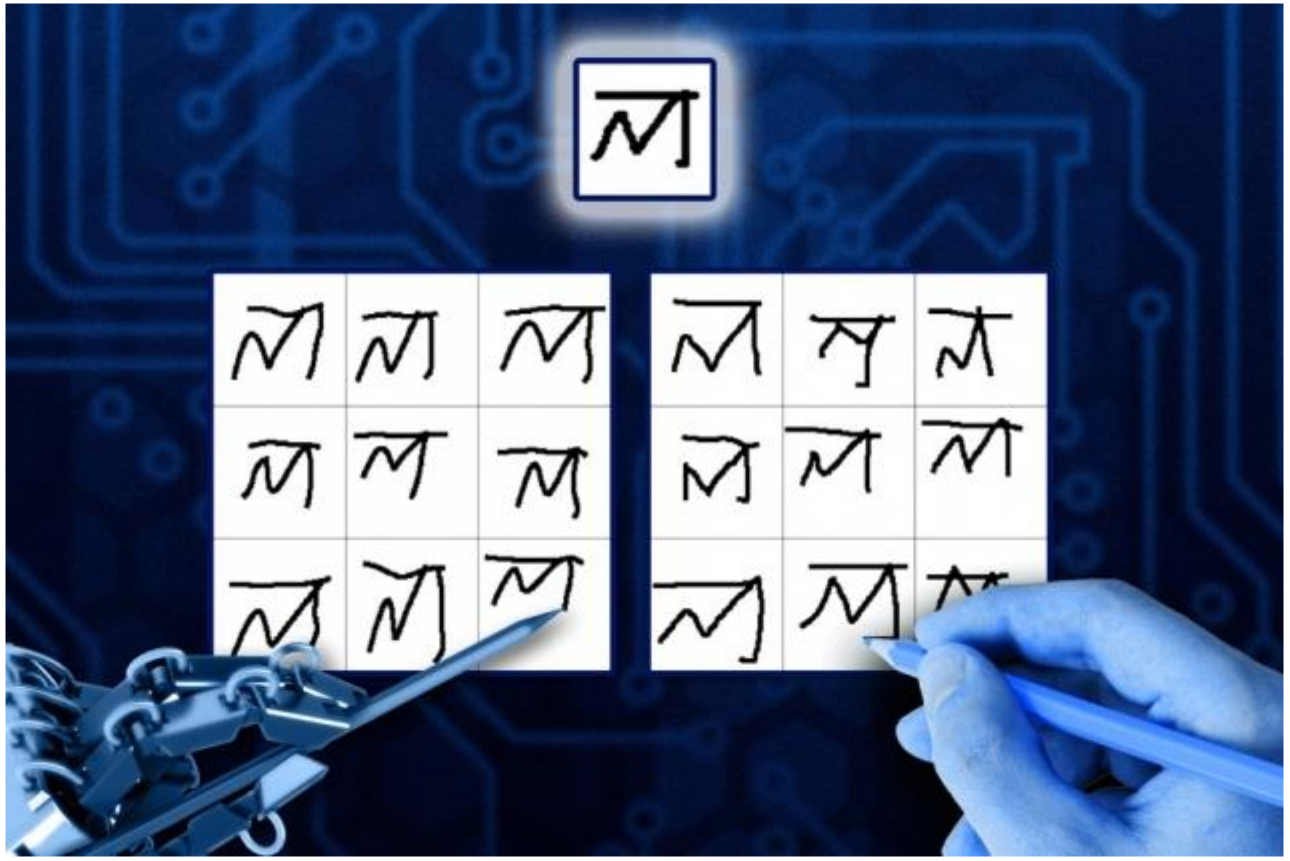
Program of the Summer School on Deep Learning and Bayesian Methods 2019

August 20, Tue	August 21, Wed	August 22, Thu	August 23, Fri	August 24, Sat	August 25, Sun
10:00 - 10:15 Welcome notes	🟩 10:00 - 11:30 Stochastic variational inference and variational autoencoders Dmitry Vetrov	🟩 10:00 - 11:30 Generative adversarial networks Egor Zakharov	🟩 10:00 - 11:30 Gaussian processes and Bayesian optimization Evgeny Burnaev	🟩 10:00 - 11:30 Markov Chain Monte Carlo Dmitry Kropotov	🟩 10:00 - 11:30 Bayesian neural networks Dmitry Molchanov
🟩 10:15 - 11:15 Introduction to Bayesian methods Dmitry Vetrov	11:30 - 12:00 Coffee break	11:30 - 12:00 Coffee break	11:30 - 12:00 Coffee break	11:30 - 12:00 Coffee break	11:30 - 12:00 Coffee break
11:15 - 11:45 Coffee break	🟨 12:00 - 13:00 Variational autoencoders Kirill Struminsky	🟨 12:00 - 13:30 Generative adversarial networks Egor Zakharov	🟨 12:00 - 13:30 Gaussian processes and Bayesian optimization Yermek Kapushev	🟨 12:00 - 13:30 Markov Chain Monte Carlo Viktor Yanush	🟨 12:00 - 13:45 Sparsification of deep neural networks Arsenii Ashukha Dmitry Molchanov
🟨 11:45 - 12:30 Bayesian reasoning Ekaterina Lobacheva	13:00 - 13:15 Break	13:30 - 14:30 Lunch	13:30 - 14:30 Lunch	13:30 - 14:30 Lunch	13:45 - 14:45 Lunch
12:30 - 12:45 Break	🟩 13:15 - 14:15 Discrete variable models Artem Sobolev	🟩 14:30 - 15:30 Normalizing flows Arsenii Ashukha	🟪 14:30 - 16:00 Deep Gaussian processes Maurizio Filippone	🟩 14:30 - 16:00 Langevin dynamics for sampling and global optimization Kirill Neklyudov	🟪 14:45 - 16:15 Uncertainty estimation in supervised learning Andrey Malinin
🟩 12:45 - 13:45 Variational inference Dmitry Vetrov	14:15 - 15:15 Lunch	15:30 - 15:45 Break	16:00 - 16:15 Break	16:00 - 16:30 Sponsor talk	16:15 - 16:30 Break
13:45 - 14:45 Lunch	🟨 15:15 - 16:15 Discrete variable models Kirill Struminsky	🟨 15:45 - 17:15 Normalizing flows Arsenii Ashukha Kirill Struminsky	🟪 16:15 - 17:15 Adaptive skip-gram model Sergey Bartunov	16:30 - 17:00 Coffee break	🟩 16:30 - 17:30 Loss surfaces Dmitry Molchanov
🟩 14:45 - 15:45 Latent variable models and EM-algorithm Dmitry Vetrov	16:15 - 16:30 Break	17:15 - 19:15 Poster session	17:15 - 17:45 Coffee break	🟪 17:00 - 18:30 Variational inference with implicit and semi-implicit models Francisco Ruiz	18:00 - 23:00 Closing reception
15:45 - 16:00 Break	🟪 16:30 - 18:00 Fair machine learning Novi Quadrianto		🟨 17:45 - 19:15 Adaptive skip-gram model Sergey Bartunov		
🟨 16:00 - 17:15 Approximate Bayesian inference Ekaterina Lobacheva	18:00 - 22:00 Social event				
17:15 - 19:15 Poster session					

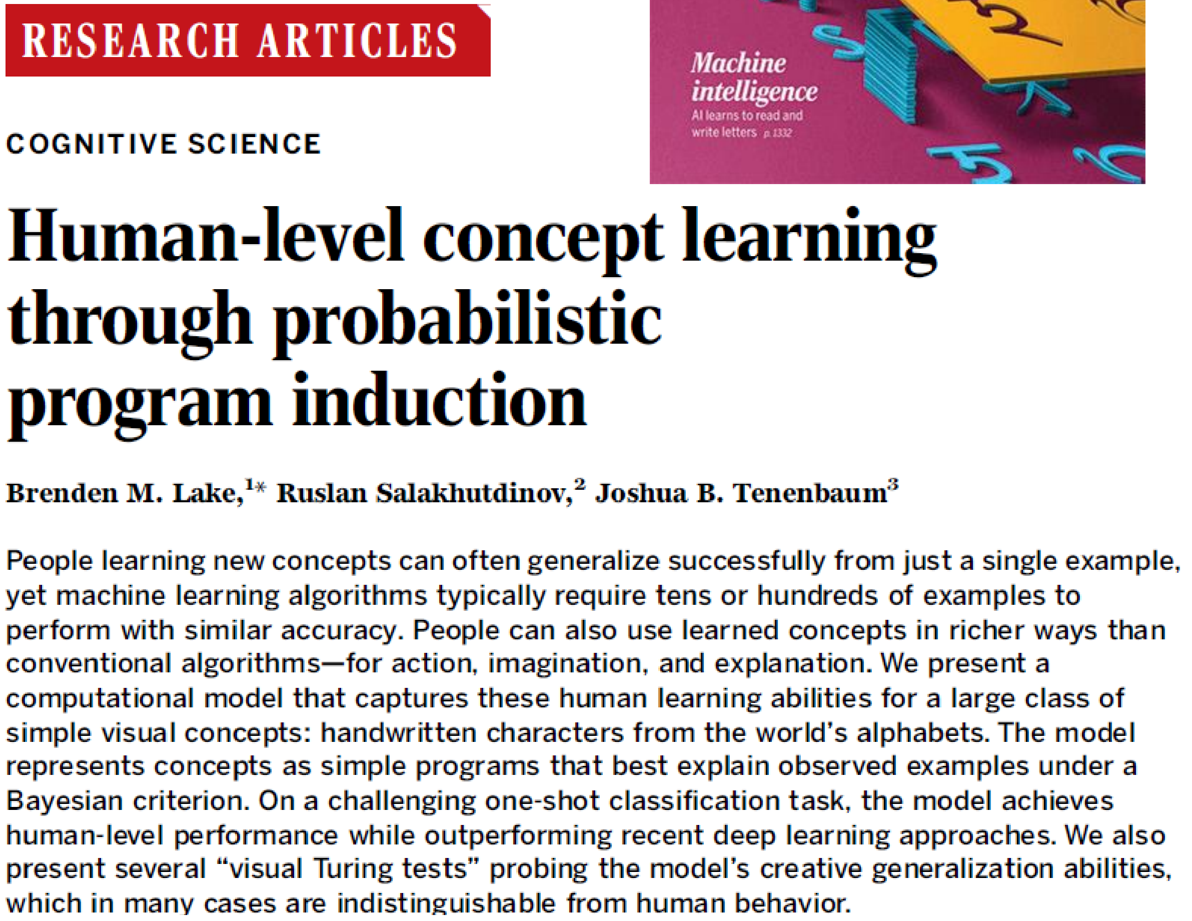
🟩 Lecture	🟪 Keynote Lecture	🟨 Practical Session

Some Accomplishments

Humans and machines were given an image of a novel character (top) and asked to produce new versions.



Can your phone improve your health? p. 1306
Vitamin C and cancer cell metabolism pp. 1317 & 1391
In search of stem cells p. 1319
Science
11 DECEMBER 2015
AAAS
Machine intelligence
AI learns to read and write letters p. 1332



RESEARCH ARTICLES

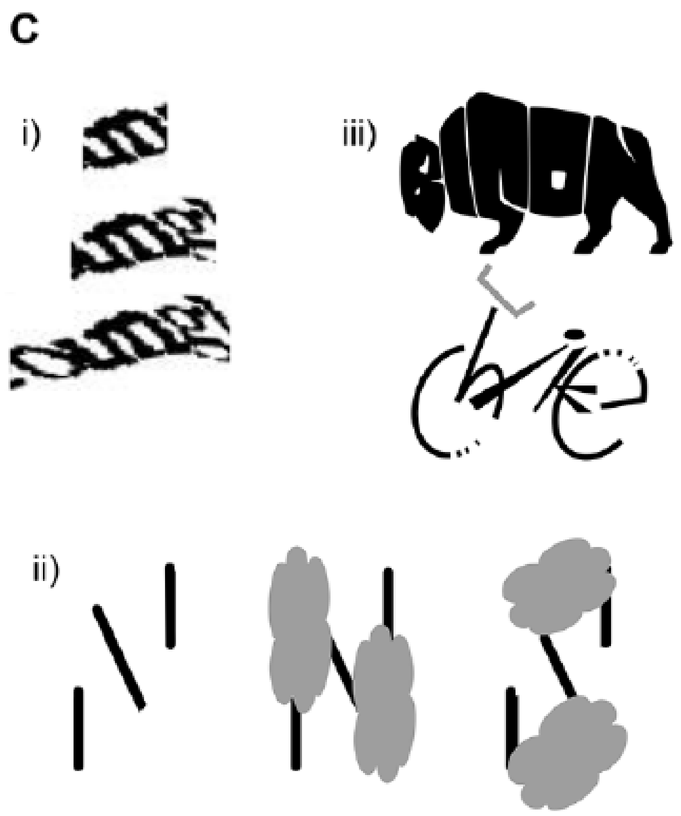
COGNITIVE SCIENCE

Human-level concept learning through probabilistic program induction

Brenden M. Lake,[1*] Ruslan Salakhutdinov,[2] Joshua B. Tenenbaum[3]

People learning new concepts can often generalize successfully from just a single example, yet machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy. People can also use learned concepts in richer ways than conventional algorithms—for action, imagination, and explanation. We present a computational model that captures these human learning abilities for a large class of simple visual concepts: handwritten characters from the world's alphabets. The model represents concepts as simple programs that best explain observed examples under a Bayesian criterion. On a challenging one-shot classification task, the model achieves human-level performance while outperforming recent deep learning approaches. We also present several "visual Turing tests" probing the model's creative generalization abilities, which in many cases are indistinguishable from human behavior.

Common sense and context affect letter form perception: (i) m vs u and n. (ii) the same line segments are interpreted as N or S depending on occluder positions (iii) perception of the shapes aids the recognition .



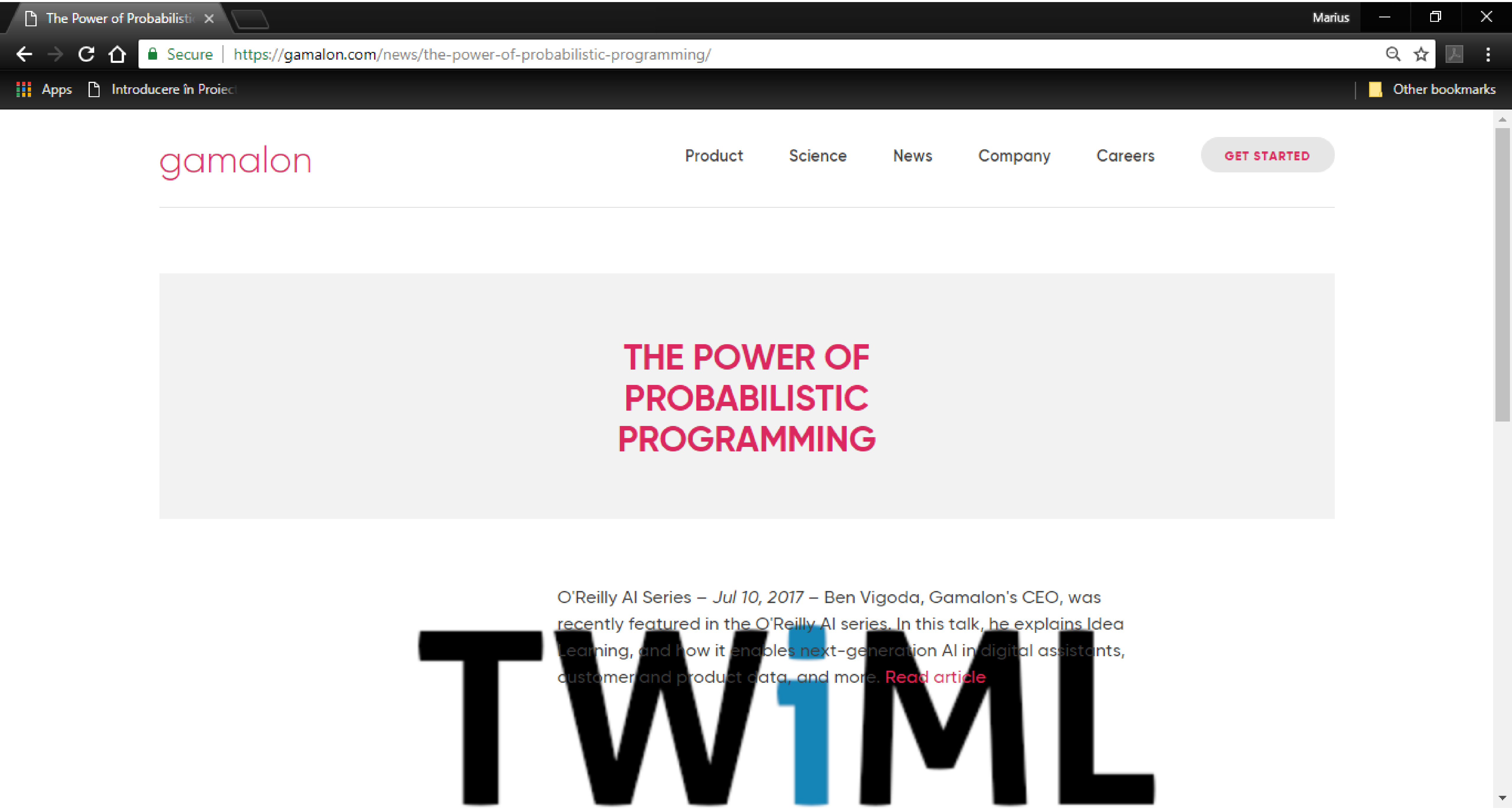
A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs

D. George,* W. Lehrach, K. Kansky, M. Lázaro-Gredilla,* C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, D. S. Phoenix

Vicarious AI, 2 Union Square, Union City, CA 94587, USA.

*Corresponding author. Email: dileep@vicarious.com (D.G.); miguel@vicarious.com (M.L.-G.)

Learning from few examples and generalizing to dramatically different situations are capabilities of human visual intelligence that are yet to be matched by leading machine learning models. By drawing inspiration from systems neuroscience, we introduce a probabilistic generative model for vision in which message-passing based inference handles recognition, segmentation and reasoning in a unified way. The model demonstrates excellent generalization and occlusion-reasoning capabilities, and outperforms deep neural networks on a challenging scene text recognition benchmark while being 300-fold more data efficient. In addition, the model fundamentally breaks the defense of modern text-based CAPTCHAs by generatively segmenting characters without CAPTCHA-specific heuristics. Our model emphasizes aspects like data efficiency and compositionality that may be important in the path toward general artificial intelligence.



THE POWER OF PROBABILISTIC PROGRAMMING

O'Reilly AI Series – *Jul 10, 2017* – Ben Vigoda, Gamalon's CEO, was recently featured in the O'Reilly AI series. In this talk, he explains Idea Learning, and how it enables next-generation AI in digital assistants, customer and product data, and more. Read article

Course Goal & Objectives

- Understanding of the basic concepts and mechanisms of the probabilistic programming
- Ability of using a probabilistic programming framework (PyMC, Edward)
- Using probabilistic programming for building a probabilistic model of a phenomenon, inferring the model parameters given the model and the data, criticizing the model

Tools

PyMC (https://github.com/pymc-devs/pymc)

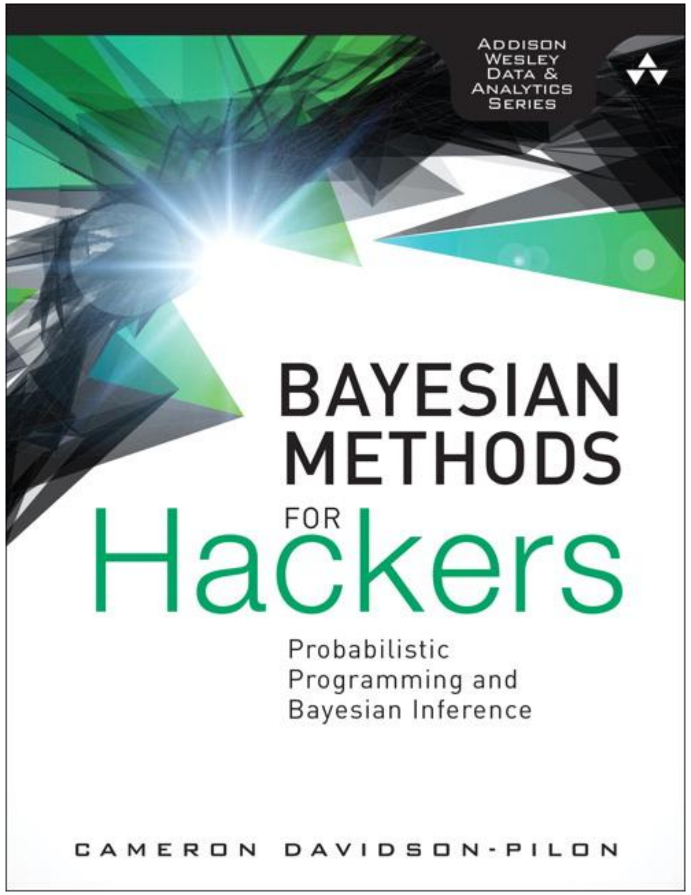
Edward: A library for probabilistic modeling, inference, and criticism (http://edwardlib.org/)

TensorFlow Probability Edward2: A probabilistic programming language (https://www.tensorflow.org/probability)

PYRO: Deep Universal Probabilistic Programming (https://pyro.ai/)

?

Bibliography



Addison Wesley Data & Analytics Series

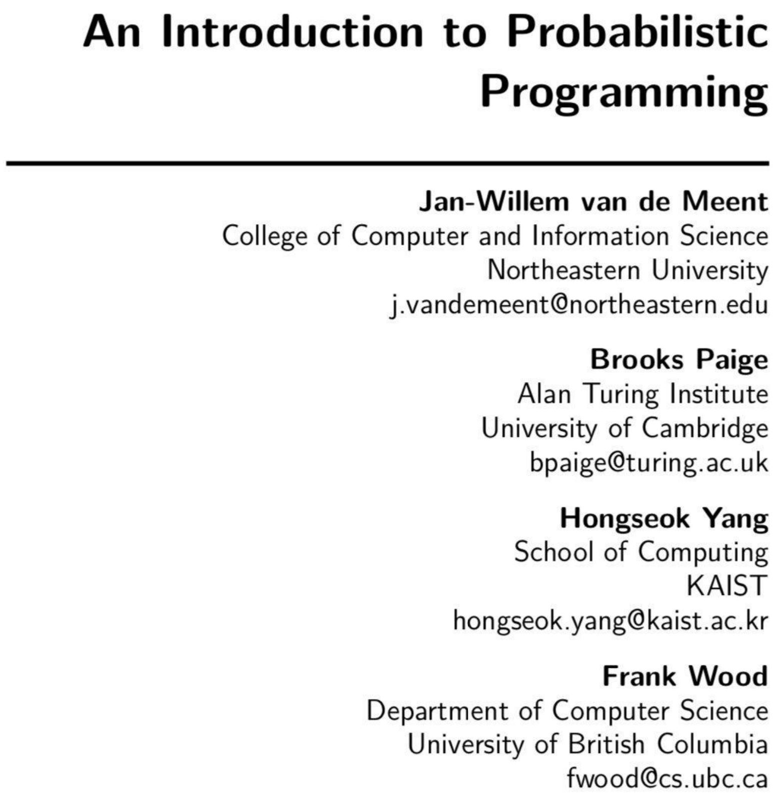
BAYESIAN METHODS FOR Hackers

Probabilistic Programming and Bayesian Inference

CAMERON DAVIDSON-PILON

https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

https://arxiv.org/abs/1809.10756



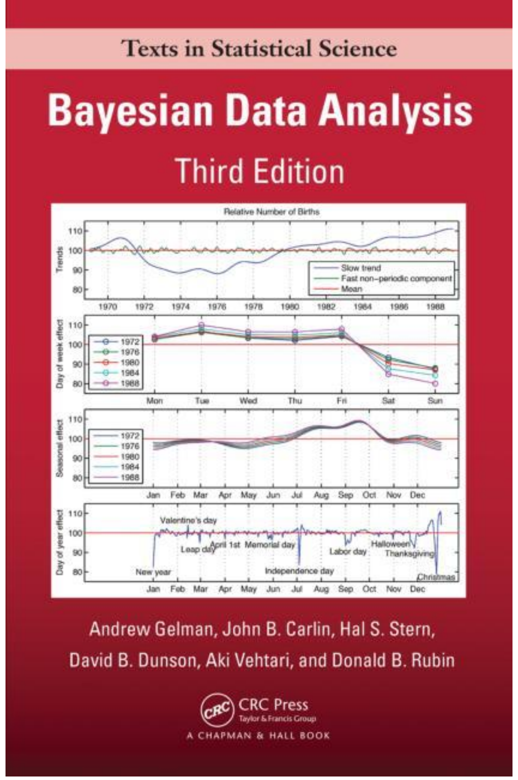
An Introduction to Probabilistic Programming

Jan-Willem van de Meent
College of Computer and Information Science
Northeastern University
j.vandemeent@northeastern.edu

Brooks Paige
Alan Turing Institute
University of Cambridge
bpaige@turing.ac.uk

Hongseok Yang
School of Computing
KAIST
hongseok.yang@kaist.ac.kr

Frank Wood
Department of Computer Science
University of British Columbia
fwood@cs.ubc.ca



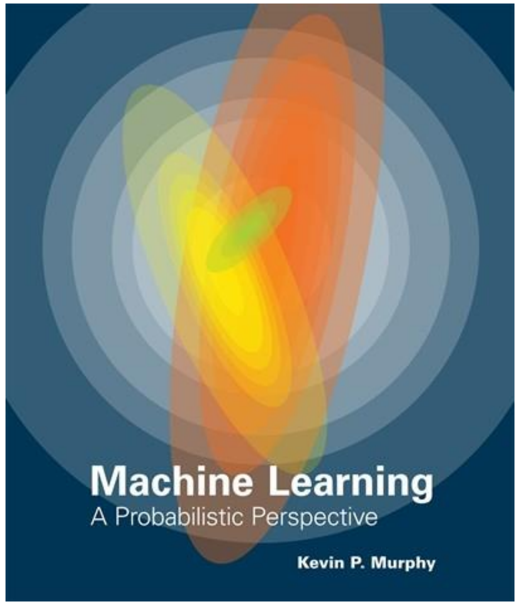
Texts in Statistical Science

Bayesian Data Analysis

Third Edition

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin

CRC Press, Taylor & Francis Group, A Chapman & Hall Book



Machine Learning

A Probabilistic Perspective

Kevin P. Murphy

"Hello World!"

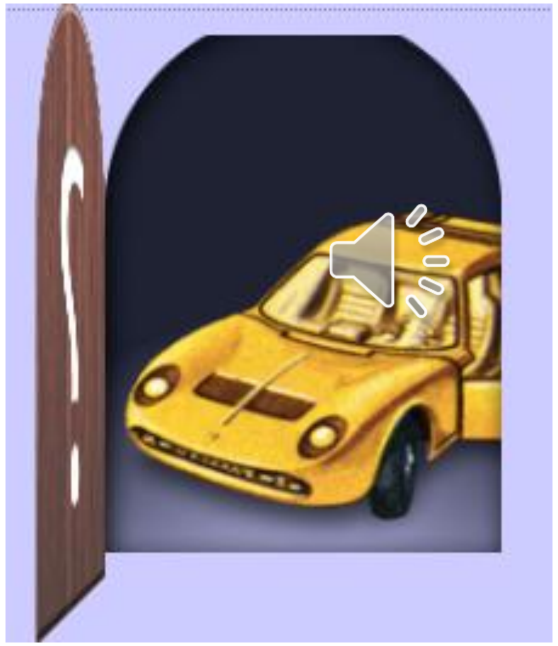
Monty Hall Problem

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

Monty Hall Problem

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

Monty Hall Problem



Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

PyMC Solution

```
import pymc as pm

car_door = pm.DiscreteUniform("car_door", lower = 1, upper = 3)
picked_door = pm.DiscreteUniform("picked_door", lower = 1, upper = 3)
preference = pm.DiscreteUniform("preference", lower = 0, upper = 1)
```

PyMC Solution

```
@pm.deterministic
def host_choice(car_door = car_door, picked_door = picked_door, preference = preference):
    if car_door != picked_door: return 6 - car_door - picked_door
    if car_door == 1:
      left = 2
      right = 3
    else:
      left = 1
      if car_door == 2:
        right = 3
      else:
        right = 2
    out = right if preference else left
    return out
```

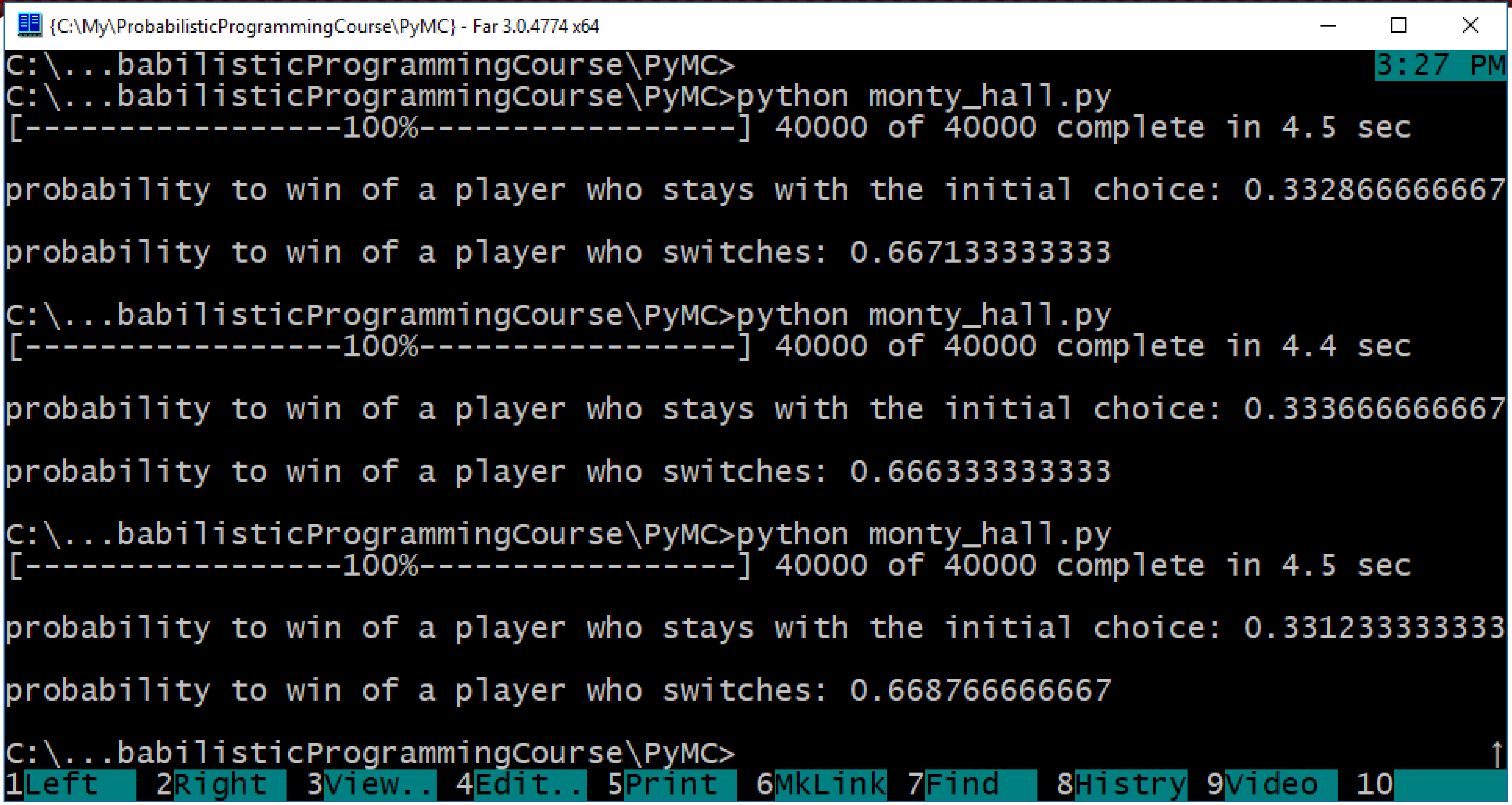
PyMC Solution

```
@pm.deterministic
def changed_door(picked_door = picked_door, host_choice = host_choice):
    return 6 - host_choice - picked_door
```

PyMC Solution

```
model = pm.Model([car_door, picked_door, preference, host_choice, changed_door])


mcmc = pm.MCMC(model)

mcmc.sample(40000, 10000, 1)

car_door_samples = mcmc.trace('car_door')[:]

picked_door_samples = mcmc.trace('picked_door')[:]

changed_door_samples = mcmc.trace('changed_door')[:]


print()

print()

print("probability to win of a player who stays with the initial choice:",

      (car_door_samples == picked_door_samples).mean())

print("probability to win of a player who switches:",

      (car_door_samples == changed_door_samples).mean())
```

PyMC Solution



{C:\My\ProbabilisticProgrammingCourse\PyMC} - Far 3.0.4774 x64

```
C:\...babilisticProgrammingCourse\PyMC>                                        3:27 PM
C:\...babilisticProgrammingCourse\PyMC>python monty_hall.py
[-----------------100%-----------------] 40000 of 40000 complete in 4.5 sec

probability to win of a player who stays with the initial choice: 0.332866666667

probability to win of a player who switches: 0.667133333333

C:\...babilisticProgrammingCourse\PyMC>python monty_hall.py
[-----------------100%-----------------] 40000 of 40000 complete in 4.4 sec

probability to win of a player who stays with the initial choice: 0.333666666667

probability to win of a player who switches: 0.666333333333

C:\...babilisticProgrammingCourse\PyMC>python monty_hall.py
[-----------------100%-----------------] 40000 of 40000 complete in 4.5 sec

probability to win of a player who stays with the initial choice: 0.331233333333

probability to win of a player who switches: 0.668766666667

C:\...babilisticProgrammingCourse\PyMC>
1Left   2Right   3View..  4Edit..  5Print   6MkLink  7Find    8Histry  9Video   10
```

Administrative Details

Two Project Assignments

	Release date	Due date
Project 1	Week 7	Week 10
Project 2	Week 11	Week 14

- The two assignments will be graded from 0 to 10
- Collaboration: All students must work individually
- Any late submissions are penalized at a rate of 20% per week

Grading

- Project 1 – ~~35%~~
- Project 2 – ~~35%~~
- ~~Exam - 30%~~

Grading

- Project 1 – 50%
- Project 2 – 50%