# Machine Learning and Programming Languages: Challenges and Opportunities

**Ugo Dal Lago**

*First ALMA AI Workshop on Foundations of AI*

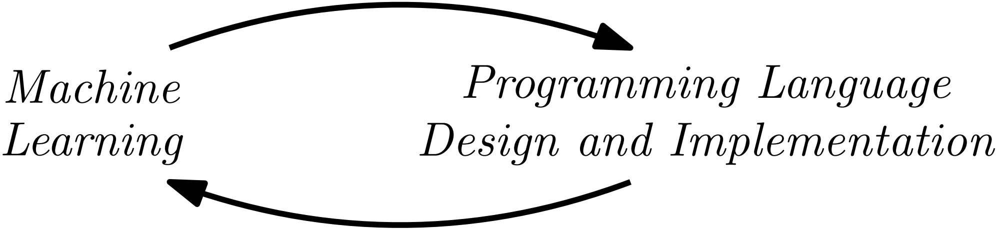
# ML vs. PL

*Machine Learning*

*Programming Language Design and Implementation*

# ML PL

- Synthesis, and automatic program completion;
- Prediction of program intentions;
- Code classification.
- ...



- Synthesis, and automatic program completion;
- Prediction of program intentions;
- Code classification.
- ...

*Machine Learning*

*Programming Language Design and Implementation*

- Tensor Programming;
- Differentiable Programming;
- Bayesian Programming;
- ...

- Synthesis, and automatic program completion;
- Prediction of program intentions;
- Code classification.
- ...

*Machine Learning*

*Programming Language Design and Implementation*

- Tensor Programming;
- Differentiable Programming;
- Bayesian Programming;
- ...

This talk

This talk

# Differentiable Programming

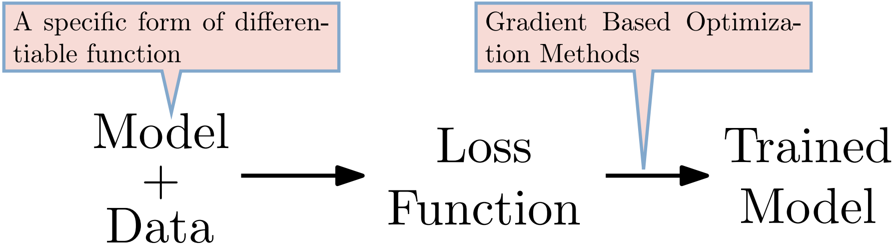
Model + Data $\longrightarrow$ Loss Function $\longrightarrow$ Trained Model

# Differentiable Programming

A specific form of differentiable function

Model + Data $\longrightarrow$ Loss Function $\longrightarrow$ Trained Model

# Differentiable Programming



# Differentiable Programming

A specific form of differentiable function

Gradient Based Optimization Methods

Model + Data $\longrightarrow$ Loss Function $\longrightarrow$ Trained Model

Differential Programming

=

"What if we take *any* program from a general purpose programming language as the underlying *model*?"

# DP - Applications

## Differentiable Programming for Image Processing and Deep Learning in Halide


TZU-MAO LI, MIT CSAIL
MICHAËL GHARBI, MIT CSAIL
ANDREW ADAMS, Facebook AI Research
FRÉDO DURAND, MIT CSAIL
JONATHAN RAGAN-KELLEY, UC Berkeley & Google

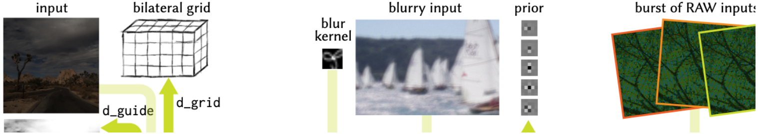




IMAGE PROCESSING

# DP - Applications

## Fast Greeks by Algorithmic Differentiation

Luca Capriotti*

*Global Modelling and Analytics Group, Investment Banking Division, Credit Suisse Group, Eleven Madison Avenue, New York City, NY 10010-3086, United States of America*

(Dated: June 2, 2010)

We show how Algorithmic Differentiation can be used to implement efficiently the Pathwise Derivative method for the calculation of option sensitivities with Monte Carlo. The main practical difficulty of the Pathwise Derivative method is that it requires the differentiation of the payout function. For the type of structured options for which Monte Carlo simulations are usually employed, these derivatives are typically cumbersome to calculate analytically, and too time consuming to evaluate with standard finite-differences approaches. In this paper we address this problem and show how Algorithmic Differentiation can be employed to calculate very efficiently and with machine precision accuracy these derivatives. We illustrate the basic workings of this computational technique by means of simple examples, and we demonstrate with several numerical tests how the Pathwise Derivative method combined with Algorithmic Differentiation – especially in the adjoint mode – can provide speed-ups of several orders of magnitude with respect to standard methods.

Keywords: Algorithmic Differentiation, Monte Carlo Simulations, Derivatives Pricing

### I. INTRODUCTION

Monte Carlo (MC) simulations are becoming the main

is generally smaller than the one of Bum
limitation of the technique is that it invol
tiation of the payout function. These deri

COMPUTATIONAL
FINANCE

# DP - Challenges

```python
def I(x):
    if x==0:
        return 0
    else:
        return x
```

# DP - Challenges

```
def I(x):
    if x==0:
        return 0
    else:
        return x
```

$$\xrightarrow{\nabla}$$

```
def dI(x):
    if x==0:
        return 0
    else:
        return 1
```

# DP - Challenges

```
def I(x):
    if x==0:
        return 0
    else:
        return x
```

$$\xrightarrow{\nabla}$$

```
def dI(x):
    if x==0:
        return 0
    else:
        return 1
```

- **Soundness**: to which extent does the derivative program $\nabla$`P` compute the actual derivative of `P`?
- **Generality**: for which program constructs could $\nabla(\cdot)$ be defined?
- **Efficiency**: what if the program `P` gets complex? How long does it take to compute $\nabla$(`P`)?

# Bayesian Programming

Model
+
Data

$\longrightarrow$

Trained
Model

# Bayesian Programming

The joint distributions of some random variables, called the *prior*.

Model
+
Data → Trained Model

# Bayesian Programming

The joint distributions of some random variables, called the *prior*.

Bayesian Inference

Model
+
Data $\longrightarrow$ Trained Model

# Bayesian Programming

The joint distributions of some random variables, called the *prior*.

Bayesian Inference

Model + Data $\xrightarrow{\text{Bayesian Inference}}$ Trained Model

The prior conditioned to data, called the *posterior*.

# Bayesian Programming

The joint distributions of some random variables, called the *prior*.

Bayesian Inference

Model + Data $\longrightarrow$ Trained Model

The prior conditioned to data, called the *posterior*.

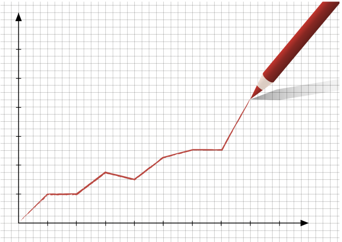
Bayesian Programming

=

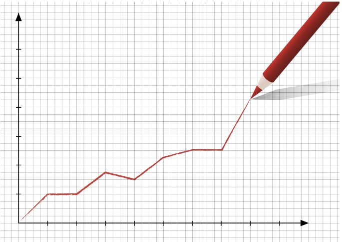
"What if we take *any* program from a general purpose randomized programming language as the underlying *prior*?"

# BP - Applications

# BP - Applications



# BP - Applications



# BP - Applications

normalize(

$x \leftarrow$ sample($bern\left(\frac{5}{7}\right)$);

$r \leftarrow$ if $x$ *then* 10 *else* 3;

**observe** 4 from $poisson(r)$;

return($x$))

# BP - Applications

1. normalize(
2. $x \leftarrow$ sample($bern\left(\frac{5}{7}\right)$);
3. $r \leftarrow$ if $x$ *then* $10$ *else* $3$;
4. **observe** $4$ from $poisson(r)$;
5. return($x$))

- 

# BP - Applications

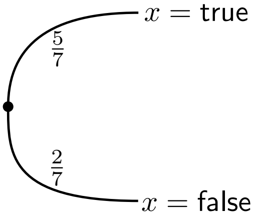
normalize(

$\quad x \leftarrow$ sample$(bern\left(\frac{5}{7}\right))$;

$\quad r \leftarrow$ if $x$ *then* $10$ *else* $3$;

$\quad$ **observe** $4$ from $poisson(r)$;

$\quad$ return$(x)$)



# BP - Applications

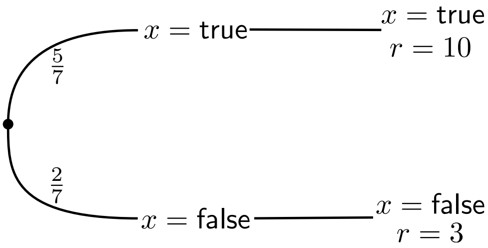
normalize(

  $x \leftarrow$ sample$\left(bern\left(\frac{5}{7}\right)\right)$;

  $r \leftarrow$ if $x$ *then* $10$ *else* $3$;

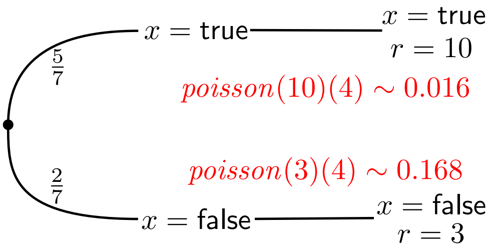
  **observe** $4$ from $poisson(r)$;

  return$(x)$)



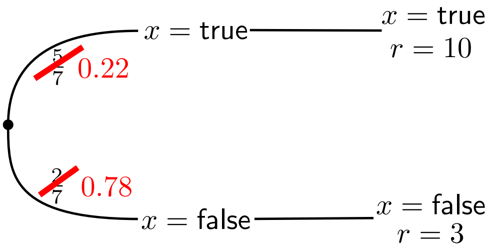
# BP - Applications

```
normalize(
    x ← sample(bern(5/7));
    r ← if x then 10 else 3;
    observe 4 from poisson(r);
    return(x))
```



# BP - Applications

normalize(
$\quad x \leftarrow$ sample$(bern\left(\frac{5}{7}\right))$;
$\quad r \leftarrow$ if $x$ *then* 10 *else* 3;
$\quad$ **observe** 4 from $poisson(r)$;
$\quad$ return$(x)$)



# BP - Applications

INJURY PREDICTION

RETURN TO ISSUE | < PREV **ARTICLE** NEXT >

## Predicting Drug-Induced Liver Injury with Bayesian Machine Learning

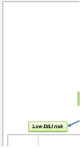
Dominic P. Williams*, Stanley E. Lazic, Alison J. Foster, Elizaveta Semenova, and Paul Morgan

**Cite this:** *Chem. Res. Toxicol.* 2020, 33, 1, 239–248
Publication Date: September 19, 2019
https://doi.org/10.1021/acs.chemrestox.9b00264


**SUBJECTS:** Anatomy, Assays, Bioactivation, Mathematical methods, Toxicity

## Abstract

Drug induced liver injury (DILI) can require significant risk management in drug development and on occasion can cause morbidity or mortality, leading to drug attrition. Optimizing candidates preclinically can minimize hepatotoxicity risk, but it is difficult to predict due to multiple etiologies encompassing DILI, often with multifactorial and overlapping mechanisms. In addition to epidemiological risk factors, physicochemical properties, dose, disposition, lipophilicity, and hepatic metabolic function are also relevant for DILI risk. Better human-relevant, predictive models are required to improve hepatotoxicity risk assessment in drug discovery. Our hypothesis is that integrating mechanistically relevant hepatic safety assays with Bayesian machine learning will improve hepatic safety risk prediction. We present a quantitative and mechanistic risk assessment for candidate nomination using data from *in vitro* assays (hepatic spheroids, BSEP, mitochondrial toxicity, and bioactivation),



# BP - Challenges

normalize(

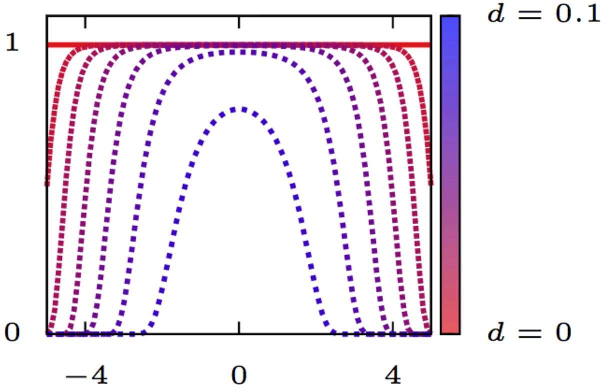
  $x \leftarrow$ sample($gauss\,(0, 1)$) in

  **observe** $d$ from $exp(1/f\,(x))$;

  return($x$))

# BP - Challenges

1. normalize(
2. $\quad x \leftarrow$ sample($gauss\ (0, 1)$) in
4. $\quad$ **observe** $d$ from $exp(1/f(x))$;
5. $\quad$ return($x$))



# BP - Challenges

```
def whm():
    h=sample(Normal(1.7,0.5))
    if sample(Bernoulli(0.5)):
        observe(Normal(h,0.1),2.0)
    return h
```

Outputs: 1.812, 1.814, 1.823, 1.813, 1.806

# BP - Challenges

```
def whm():
    h=sample(Normal(1.7,0.5))
    if sample(Bernoulli(0.5)):
        observe(Normal(h,0.1),2.0)
    return h
```

Outputs: 1.812, 1.814, 1.823, 1.813, 1.806

```
def whcm():
    h=sample(Normal(170,50))
    if sample(Bernoulli(0.5)):
        observe(Normal(h,10),200)
    return h
```

Outputs: 170.1, 170.4, 171.5, 170.2, 169.4

# BP - Challenges

```
def whm():
    h=sample(Normal(1.7,0.5))
    if sample(Bernoulli(0.5)):
        observe(Normal(h,0.1),2.0)
    return h
```

Outputs: 1.812, 1.814, 1.823, 1.813, 1.806

```
def whcm():
    h=sample(Normal(170,50))
    if sample(Bernoulli(0.5)):
        observe(Normal(h,10),200)
    return h
```

Outputs: 170.1, 170.4, 171.5, 170.2, 169.4

- **Semantics**: can we give a satisfactory semantics to BP programs sampling from continuous distributions?
- **Sound Inference**: can we prove inference algorithms correct, or even *formulate* their correctness?

# Wrapping Up

⋮

# Wrapping Up

**The Simple Essence of Automatic Differentiation**

CONAL ELLIOTT, Target, USA

**Bayesian Synthesis of Probabilistic Programs for Automatic Data Modeling**

FERAS A. SAAD, Massachusetts Institute of Technology, USA
MARCO F. CUSUMANO-TOWNER, Massachusetts Institute of Technology, USA
ULRICH SCHAECHTLE, Massachusetts Institute of Technology, USA
MARTIN C. RINARD, Massachusetts Institute of Technology, USA
VIKASH K. MANSINGHKA, Massachusetts Institute of Technology, USA

**Denotational Validation of Higher-Order Bayesian Inference**

ADAM ŚCIBIOR, University of Cambridge, England, UK and Max Planck Institute for Intelligent Systems, Germany
OHAD KAMMAR, University of Oxford, England, UK
MATTHIJS VÁKÁR, University of Oxford, England, UK
SAM STATON, University of Oxford, England, UK
HONGSEOK YANG, KAIST, South Korea
YUFEI CAI, Universität Tübingen, Germany
KLAUS OSTERMANN, Universität Tübingen, Germany
SEAN K. MOSS, University of Cambridge, England and University of Oxford, England, UK
CHRIS HEUNEN, University of Edinburgh, Scotland, UK
ZOUBIN GHAHRAMANI, University of Cambridge, England, UK and Uber AI Labs, California, USA

**LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models**

Yuan Zhou, Bradley J. Gram-Hansen, Tobias Kohn, Tom Rainforth, Hongseok Yang, Frank Wood
University of Oxford, University of Cambridge, KAIST, University of British Columbia

**Automating Involutive MCMC using Probabilistic and Differentiable Programming**

Marco Cusumano-Towner, Alexander K. Lew, Vikash K. Mansinghka
Massachusetts Institute of Technology

**Backpropagation in the Simply Typed Lambda-Calculus with Linear Negation**

ALOÏS BRUNEL, Deepomatic, France
DAMIANO MAZZA, CNRS, France
MICHELE PAGANI, Université de Paris, France

**On the Versatility of Open Logical Relations***
**Continuity, Automatic Differentiation, and a Containment Theorem**

Gilles Barthe[1,4], Raphaëlle Crubillé[4], Ugo Dal Lago[2,3], and Francesco Gavazzo[2,3,4]

⋮

⋮

# Wrapping Up

PPX

julia

The Simple Essence of Automatic Differentiation

CONAL ELLIOTT, Target, USA

Bayesian Synthesis of Probabilistic Programs for Automatic Data Modeling

FERAS A. SAAD, Massachusetts Institute of Technology, USA
MARCO F. CUSUMANO-TOWNER, Massachusetts Institute of Technology, USA
ULRICH SCHAECHTLE, Massachusetts Institute of Technology, USA
MARTIN C. RINARD, Massachusetts Institute of Technology, USA
VIKASH K. MANSINGHKA, Massachusetts Institute of Technology, USA

Denotational Validation of Higher-Order Bayesian Inference

ADAM ŚCIBIOR, University of Cambridge, England, UK and Max Planck Institute for Intelligent Systems, Germany
OHAD KAMMAR, University of Oxford, England, UK
MATTHIJS VÁKÁR, University of Oxford, England, UK
SAM STATON, University of Oxford, England, UK
HONGSEOK YANG, KAIST, South Korea
YUFEI CAI, Universität Tübingen, Germany
KLAUS OSTERMANN, Universität Tübingen, Germany
SEAN K. MOSS, University of Cambridge, England and University of Oxford, England, UK
CHRIS HEUNEN, University of Edinburgh, Scotland, UK
ZOUBIN GHAHRAMANI, University of Cambridge, England, UK and Uber AI Labs, California, USA

LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models

Yuan Zhou, Bradley J. Gram-Hansen, Tobias Kohn, Tom Rainforth, Hongseok Yang, Frank Wood
University of Oxford, University of Cambridge, KAIST, University of British Columbia

Automating Involutive MCMC using Probabilistic and Differentiable Programming

Marco Cusumano-Towner, Alexander K. Lew, Vikash K. Mansinghka
Massachusetts Institute of Technology

Backpropagation in the Simply Typed Lambda-Calculus with Linear Negation

ALOÏS BRUNEL, Deepomatic, France
DAMIANO MAZZA, CNRS, France
MICHELE PAGANI, Université de Paris, France

On the Versatility of Open Logical Relations*

Continuity, Automatic Differentiation, and a Containment Theorem

Gilles Barthe[1,4], Raphaëlle Crubillé[4], Ugo Dal Lago[2,3], and Francesco Gavazzo[2,3,4]

⋮

⋮

# Thank You!

Questions?