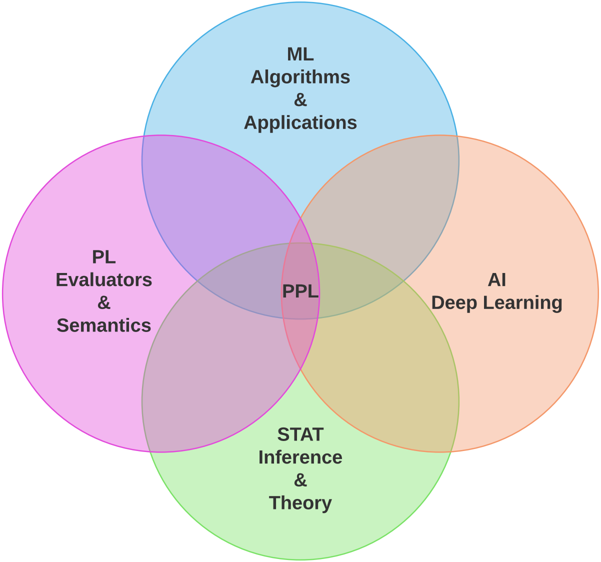
MultiVerse: Causal Reasoning Using Importance Sampling in Probabilistic Programming

Presented by:
Haluk Dogan
https://haluk.github.io/
hdogan@vivaldi.net
Some slides copied from
Frank Wood

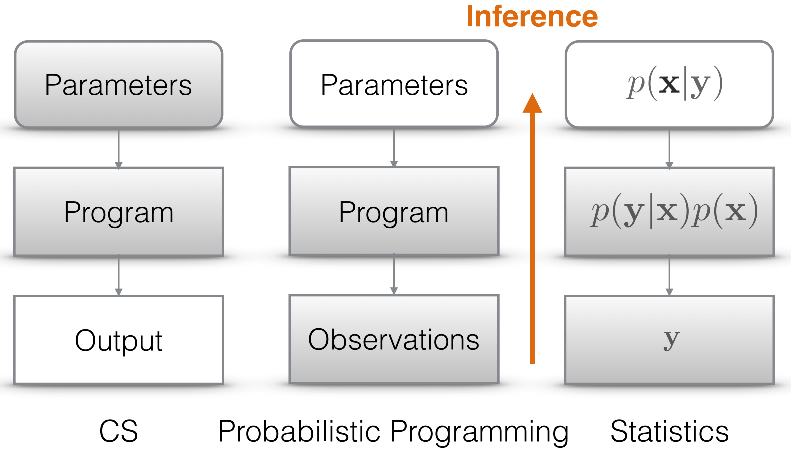
Department of Computer Science
University of Nebraska-Lincoln

February 17, 2020

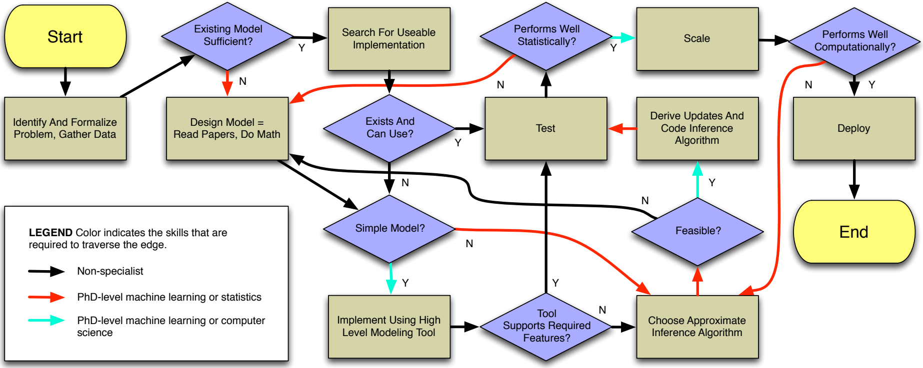
Probabilistic Programming Languages (PPL)



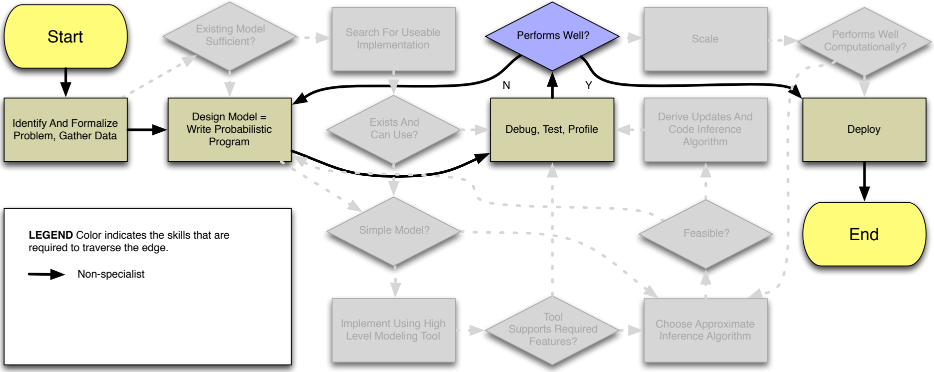
Intuition


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

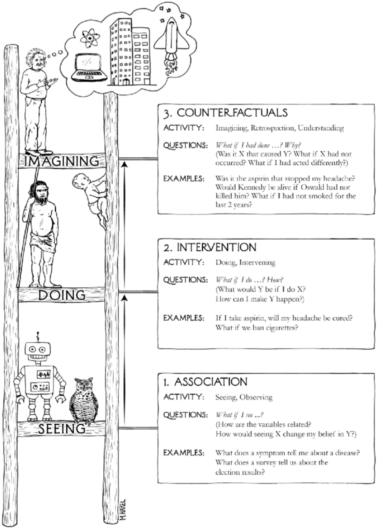
The Way Machine Learning Is



The Way Machine Learning Will Be



Causality



Ladder of Causation (Judea Pearl 2018)

Causal Model $M(X, Y, F)$:

- X: Exogenous variables (latent)
- Y: Endogenous variables (observed)
- F: Set of structural equations $\{F_K \mid K \in Y\}$

Counterfactual inference query:

$$P(Y' \mid Y = e; \text{do}(D = d))$$

Evaluating this query in 3 steps:

- Abduction (observational inference)
- Intervention (action)
- Prediction

Abduction and Counterfactual Inference

Abduction

- The hardest part of the counterfactual procedure
  - Exact inference is possible but either difficult or intractable
- $M' \leftarrow P(X \mid Y = e)$
  - It has the same structure as model M
  - X is replaced by $X \mid Y = e$
  - Y is not observed anymore

Counterfactual inference

- Twin-net approach, i.e., loopy belief propagation (Balke and Pearl 1994)
- Single-world intervention graphs (Richardson and Robins 2013)
- Matching (Li 2012)

Importance Sampling

- Approximate inference technique

$$\mathbb{E}[f(x)] = \int f(x)p(x)dx \approx \frac{1}{n}\sum_i f\left(x_i\right)$$

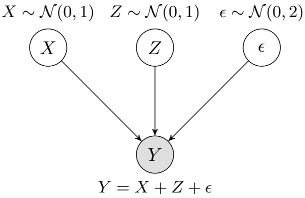
- If sampling from $p(x)$ is very hard

$$\mathbb{E}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)} q(x)dx \approx \frac{1}{n}\sum_i f\left(x_i\right)\frac{p\left(x_i\right)}{q\left(x_i\right)}$$

- $\sigma^2(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

Continuous Variable Example in MultiVerse

```python
#!/usr/bin/env python3

import timeit

from multiverse import (DeltaERP, NormalERP, ObservableNormalERP, do, observe,
                        predict, run_inference)
from utils import calculate_expectation

NUM_SAMPLES = 1000


def base_program():
    X = NormalERP(0, 1)
    Z = NormalERP(0, 1)
    Y = ObservableNormalERP(X.value + Z.value, 2, depends_on=[X, Z],)

    return X, Z, Y


def program_with_data():
    X, Z, Y = base_program()
    observe(Y, 1.2342)
    do(Z, -2.5236)

    predict(Y.value, predict_counterfactual=True)


start = timeit.default_timer()
results = run_inference(program_with_data, NUM_SAMPLES)
stop = timeit.default_timer()

print("Time:", stop - start)
result = calculate_expectation(results)
print("Prediction:", result)
```



$$\mathbb{E}(Y' \mid Y = 1.2342, \mathsf{do}(Z = -2.5236))$$

Continuous Variable Example in Pyro

```python
#!/usr/bin/env python3

import timeit

import numpy
import pyro
import torch

NUM_SAMPLES = 1000  # for both abduction and intervention/prediction
ROUND_DIGIT_APPR = (
    1  # discretisation to avoid 'infinite rejection sampling' for continuous variables
)
GUIDE_TO_USE = None
latent_procedure_sites = ["X", "Z", "Y_epsilon"]


def rounder(val):
    if ROUND_DIGIT_APPR is None:
        # Don't round:
        return torch.tensor(float(val))
    else:
        return torch.tensor(round(float(val), ROUND_DIGIT_APPR))


def extract_obs_if_any(data, var_name):
    if data is not None and var_name in data:
        return data[var_name]
    else:
        None


```

Continuous Variable Example in Pyro (cont'd)

```python
def model(data=None, posterior_distribution=None):
    if posterior_distribution is not None and "X" in posterior_distribution:
        # If we are re-using a sample from a posterior,
        # we just should use the value for this variable
        # directly:
        X = posterior_distribution["X"]
    else:
        X = pyro.sample(
            "X", pyro.distributions.Normal(0, 1), obs=extract_obs_if_any(data, "X")
        )
    if posterior_distribution is not None and "Z" in posterior_distribution:
        Z = posterior_distribution["Z"]
    else:
        Z = pyro.sample(
            "Z", pyro.distributions.Normal(0, 1), obs=extract_obs_if_any(data, "Z")
        )
    if posterior_distribution is not None and "Y_epsilon" in posterior_distribution:
        Y_epsilon = posterior_distribution["Y_epsilon"]
    else:
        Y_epsilon = pyro.sample("Y_epsilon", pyro.distributions.Normal(0, 2))
    discrete_Y = rounder(X + Z + Y_epsilon)
    # We must (re-)evaluate deterministic variables in any case:
    Y = pyro.sample(
        "Y",
        pyro.distributions.Delta(torch.tensor(discrete_Y)),
        obs=extract_obs_if_any(data, "Y"),
    )
    return X, Z, Y_epsilon, Y
```

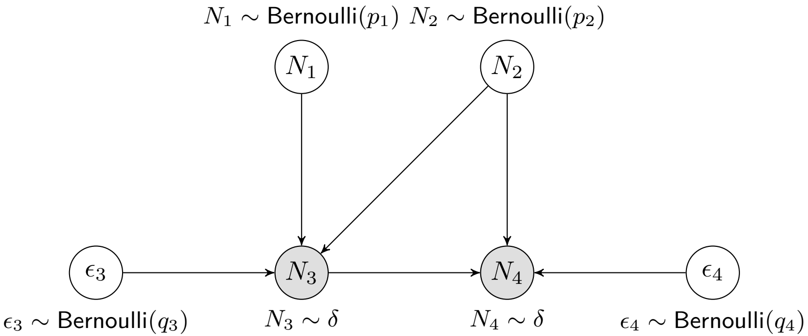
Continuous Variable Example in Pyro (cont'd)

```python
data = {"Y": rounder(1.2342), "X": None, "Z": None}
start = timeit.default_timer()

# 1. Abduction
posterior = pyro.infer.Importance(
    model, guide=GUIDE_TO_USE, num_samples=NUM_SAMPLES
).run(data=data)
print("AbductionESS:", posterior.get_ESS())
posterior = pyro.infer.EmpiricalMarginal(posterior, sites=latent_procedure_sites)

# 2. Intervention
intervention = {"Z": -2.5236}
intervened_posterior = pyro.do(model, intervention)

# 3. Prediction
predictions_Y = []
for sample_index in range(NUM_SAMPLES):
    # We are drawing a sample from the posterior world:
    posterior_sample_vector = posterior.sample()
    # We drew that sample in a vector form;
    # now we need to transform
    # it to a dictionary of variables.
    posterior_sample = {}
    for index, var_name in enumerate(latent_procedure_sites):
        if var_name in intervention:
            # We must ensure that we don't
            # use intervened variables
            # from its posterior:
            pass
        else:
            posterior_sample[var_name] = posterior_sample_vector[index]
    X, Z, Y_epsilon, Y = intervened_posterior(posterior_distribution=posterior_sample)
    predictions_Y.append(Y)

stop = timeit.default_timer()

print("Time:", stop - start)
expected_Y = numpy.mean(predictions_Y)
print("Prediction:", expected_Y)
```

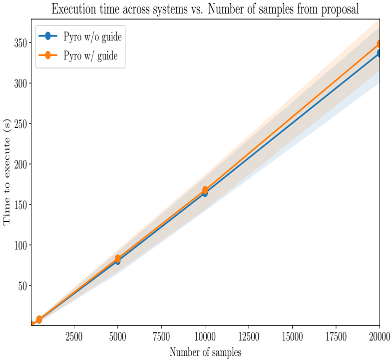
Experiments

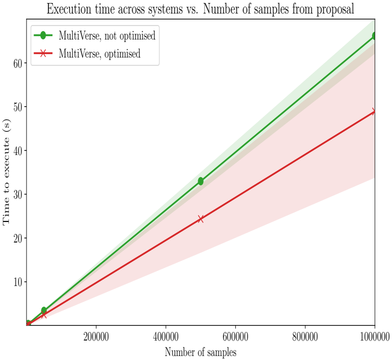
- 16-core EC2 instance m4.4xlarge
- 1000 SCM with 15 probabilistic procedures
  - BNs in the form of probabilistic programs
- MultiVerse experiments
  - produce the same number of samples in less time than Pyro
  - have better inference convergence

Test Models

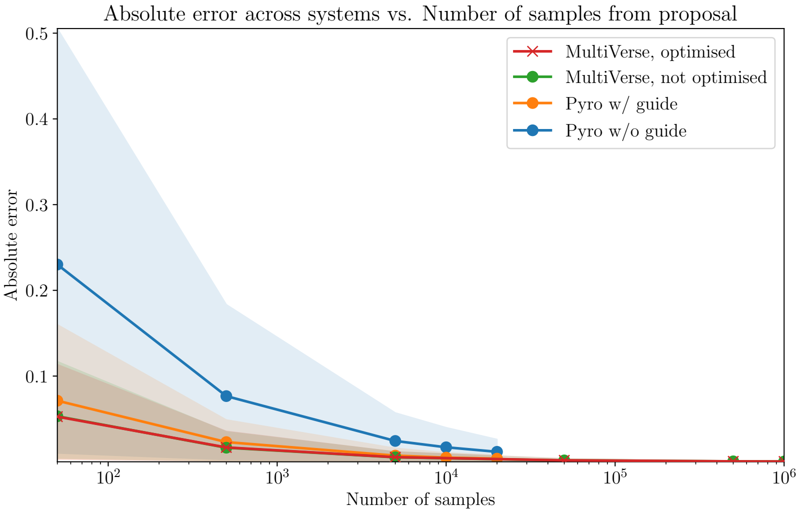

$N_1 \sim$ Bernoulli(p_1)
$N_2 \sim$ Bernoulli(p_2)
N_1
N_2
ϵ_3
N_3
N_4
ϵ_4
$\epsilon_3 \sim$ Bernoulli(q_3)
$N_3 \sim \delta$
$N_4 \sim \delta$
$\epsilon_4 \sim$ Bernoulli(q_4)

Time Performance Comparison





Convergence Test Performance



PPL Should Be Pure Functional

Anglican

```
[assume (a (normal 5 10))]
[assume (b (normal a 2))]
[assume (a (normal b 7))]
=> Error
```

Probabilistic-C

```
int a = normal(5, 10);
int b = normal(a, 2);
int a = normal(b, 7);
```

Birthday Paradox

Approximately, what's the probability that in a room filled with 23 people at least one pair of people have the same birthday?

```
[assume birthday (mem (lambda (i) (uniform-discrete 1 366)))]

[assume N 23]

[assume pair-equal
  (lambda (i j)
    (if (> i N)
      false
      (if (> j N)
        (pair-equal (+ i 1) (+ i 2))
        (if (= (birthday i) (birthday j))
          true
          (pair-equal i (+ j 1))))))]

[predict (pair-equal 1 2)]
```

Questions

Questions?

Judea Pearl, winner of Turing Award (2011)

References I

Balke, A. and J. Pearl (1994). "Probabilistic Evaluation of Counterfactual Queries". In: *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1*. Ed. by B. Hayes-Roth and R. E. Korf. AAAI Press / The MIT Press, pp. 230–237. URL: http://www.aaai.org/Library/AAAI/1994/aaai94-035.php (cit. on p. 7).

Judea Pearl, D. M. (May 15, 2018). *The Book of Why*. Hachette Book Group USA. 432 pp. ISBN: 046509760X. URL: https://www.ebook.de/de/product/30501615/judea_pearl_dana_mackenzie_the_book_of_why.html (cit. on p. 6).

Li, M. (June 2012). "Using the Propensity Score Method to Estimate Causal Effects". In: *Organizational Research Methods* 16.2, pp. 188–226. DOI: 10.1177/1094428112447816 (cit. on p. 7).

References II

Richardson, T. S. and J. M. Robins (2013). "Single world intervention graphs (SWIGs): A unification of the counterfactual and graphical approaches to causality". In: *Center for the Statistics and the Social Sciences, University of Washington Series. Working Paper* 128.30, p. 2013 (cit. on p. 7).