Learning Probabilistic Programs

Xin Zhang
Peking University
Recap of Last Lecture

• Evaluation-based inference
  • Dynamic
  • Can deal with programs with unbounded loops
Likelihood Weighting

• A form of importance sampling where the proposal is the prior

\[ E_{q(X)} \left[ \frac{p(X|Y)}{q(X)} r(X) \right] = \frac{1}{p(Y)} E_{q(X)} \left[ \frac{p(Y, X)}{q(X)} r(X) \right] \]

\[ \approx \frac{1}{p(Y)} \frac{1}{L} \sum_{l=1}^{L} W^l r(X^l), \]

\[ W^l = \frac{p(Y, X^l)}{q(X^l)} = \frac{p(Y|X^l)p(X^l)}{p(X^l)} = p(Y|X^l) \]  

If we use \( p(X^l) \) as the proposal distribution

Y are observed/conditioned variables
Likelihood Weighting: Variants

• Naïve Metropolis Hasting (draw random traces)

• Single-site proposal (try to only change one variable at a time)
Sequential Monte Carlo

• In probabilistic programming, sample a high-dimensional distribution by sampling a sequence of lower dimensional distributions

• Also called particle filters

• Used in signal processing and probabilistic inference
SMC: Problem Statement

Given

$p(x_0)$ and
$p(x_t \mid x_{t-1})$ and
$p(y_t \mid x_t)$ and
Observations $y_{1:t}$

Estimate

$p(x_{0:t} \mid y_{1:t})$ or
$p(x_t \mid y_{1:t})$ or
$I(f_t) = E_{p(x_{0:t} \mid y_{1:t})}[f_t(x_{0:t})] = \int f_t(x_{0:t}) p(x_{0:t} \mid y_{1:t}) dx_{0:t}$
SMC: Main Ideas

• Sample on the Markov chain:

\[ \pi(x_{0:t} \mid y_{1:t}) = \pi(x_{0:t-1} \mid y_{1:t-1}) \pi(x_t \mid x_{0:t-1}, y_{1:t}) . \]

• Reweight the samples using importance sampling

• Throw away the samples (particles) with low probabilities
i=1,...,N=10 particles

From “An Introduction to Sequential Monte Carlo Methods” by Arnaud Doucet, Nando De Freitas, and Neil Gordon
SMC: Bootstrap Filter

Assume the proposal distribution is $p(x_{1:t})$

1. Initialization. $T = 0$
   - For $i = 1, \ldots, N$, sample $x_0^{(i)} \sim p(x_0)$ and set $t = 1$

2. Importance sampling step.
   - For sample $\tilde{x}_t^{(i)} \sim p(x_t | \tilde{x}_{t-1})$ and set ($\tilde{x}_{0:t-1}^{(i)}, \tilde{x}_t^{(i)}$).
   - For $i = 1, \ldots, N$, evaluate the importance weights.
   - Normalize the importance weights

3. Selection step
   - Resample with replacement $N$ particles from the current particles according to importance weights
   - Set $t \to t + 1$
Question 1

• In evaluation-based method, if the sampled trace doesn’t terminate, what would you do in practice?
Question 2

• Consider the program \( x = \text{uniform}(0, 1); y = \text{gaussian}(x, 1) \). Suppose the current trace is \( x = 0.5, y = 1 \). Now we want to change \( y \), what is \( p(y) \) that we're sampling from?

• What if we want to change \( x \)?
Question 3

• Consider the program

\[
\begin{align*}
x &= 0; \\
\text{while}(\text{bernoulli}(0.5)); \\
x &= x+1 \\
\text{condition}(x > 2)
\end{align*}
\]

• Describe an algorithm to sample traces from it.
Question 4

• Sequential Monte Carlo can be see as a variant of importance sampling. Is the statement right?
Question 5

• What would happen if we don’t throw away particles in sequential Monte Carlo?
This Lecture

• Learning in probabilistic programming
  • Still an active research area
  • Not a solved problem
Question

• Can you define inference and learning?
Inference vs. Learning

• Inference: given $f|\theta$, run $f|\theta$ to output data

• Learning: given $f|\theta$, and data $D$, figure out $\theta$
Inference vs. Learning

• Inference is often a part of learning
  • Example: perform inference with different parameters
Inference vs. Learning

- Inference is often a part of learning

\[
p = \text{bernoulli()}
\]
\[
D = [...]
\]
\[
\text{if } p == 1:
\]
\[
    m = \text{model1}
\]
\[
\text{else:}
\]
\[
    m = \text{model2}
\]
\[
\text{for } (x,y) \text{ in } D;
\]
\[
    \text{condition}(m(x) + \text{N}(0,0.1) == y)
\]
\[
\text{output } m
\]
Learning in Probabilistic Programming

• Parameter learning

\[
x = \text{uniform}(p1, p2) \\
y = \text{gaussian}(x, p3) \\
\text{if(bernoulli(p4))} \\
    \quad z = x \\
\text{else} \\
    \quad z = y \\
\text{condition(z > 100)}
\]

What are p1, p2, p3, p4?
Learning in Probabilistic Programming

• Structure learning

\[
x = \text{uniform}(p1, p2) \\
y = \text{gaussian}(x, p3) \\
\text{if bernoulli}(p4) \\
    z = x \\
\text{else} \\
    z = y \\
\text{condition}(z > 100)
\]
More on Structure Learning

• How to synthesize (deterministic) programs is an active field

• Program synthesis
  • Started early
  • Still under development
  • Works well in specific settings
Program Synthesis

• Given a specification, generates a program that satisfies the specification

• Main challenge: intractable search space

• Various approach to cut the search space
  • Sketch
  • SyGuS (Syntax-Guided Synthesis)
Program Synthesis: Sketch

if (x > ??)
    y = 100
else
    y = ??
output x*x+y*y

x = 1, o = 100
x = 10, o = 1000
Program Synthesis SyGuS

Syntax Constraints:

\[ e := \text{input} | e + e | e \times e | e - e | e / e \]

Semantic Constraints:

\[ e(2) = 100 \]
\[ e(5) = 700 \]

\[ \ldots \]

The semantics constraints can be more high-level than input-out examples. For example, the output of a sorting algorithm is sorted.
More on Program Synthesis

• https://people.csail.mit.edu/asolar/SynthesisCourse/TOC.htm

• https://xiongyingfei.github.io/SA/2020/main.htm
A Possible Pipeline to Synthesize Probabilistic Programs

specification → Structure Learning → Parameter Learning → Program
Two Typical Approaches

• Non-Bayesian method (Maximum Likelihood)

• Bayesian method
  • Feras A. Saad, Marco F. Cusumano-Towner, Ulrich Schaechtle, Martin C. Rinard, Vikash K. Mansinghka: Bayesian Synthesis of Probabilistic Programs for Automatic Data Modeling. POPL 19.
Ellis et al., 2015: Motivations

• Goal: unsupervised learning
  • Induce good latent representations of a data set

• Programs are a natural knowledge representation for many domains
  • Compression: find smallest representation
  • Infer both programs and inputs

• General solution is head
  • Encode domain-specific parts using a DSL
Key Ideas

• Using PCFG to limit the program space

• Symbolic search: SMT
**Problem Formalization**

Minimize

\[
- \log P_f (f) + \sum_{i=1}^{N} \left( - \log P_{x|z} (x_i | f(I_i)) \right) - \log P_I (I_i)
\]

- **Program length**
- **Data reconstruction error**
- **Data encoding length**

\( f \) is drawn from a prior determined by the sketch

\( I \) is drawn from a domain-dependent description length prior \( P_I \), which leads to \( z_i = f(I_i) \).

\( P_{x|z}(x_i|z_i) \) estimates the error between predictions and observations.

Program is largely deterministic, but inputs are random. Also, going from \( z \) to \( x \) is a random process (manually specified)
Defining a Program Space

• Probabilistic context-free grammar (PCFG)
  • Place probabilities on production rules

\[
E \rightarrow E + E \mid \mathbb{R} \mid x
\]

• Define denotations for each rule using SMT

\[
[E_1 + E_2](I) = [E_1](I) + [E_2](I) \quad [r \in \mathbb{R}](I) = r \quad [x](I) = I
\]

• We can use SMT expression to denote the synthesis problem
Solution

• Construct an SMT that
  • Defines the space of programs
  • Computes the description length
  • Computes the output given an input and a program

• Use SMT to perform linear search on the loss function
More on SMT

• Satisfiability modulo theories
  • Generalizes SAT such that each clause can contain real numbers, integers, strings, quantifiers …

• Highly expressive, but its solvers only scale under well-defined scenarios

• Representative solver: z3 from Microsoft
Example: Visual Concept Learning

- Space of programs: simple graphic programs that control a turtle
  - Rotations
  - Forward movement
  - Rescaling of shapes
  ...

- Program outputs: image parses
  - A list of shapes \(<id, scale, x, y>\)
  - A list of containment relationships \((i, j)\)
  - A list of reflexive borders relations borders \((i, j)\)

\[
s_1 = \text{Shape}(id = 1, \text{scale} = 1, x = 10, y = 15)\\
s_2 = \text{Shape}(id = 2, \text{scale} = 1, x = 27, y = 54)\\
borders(s_1, s_2)
\]
Example: Visual Concept Learning

• Program inputs:
  • Shapes
  • Positions
  • Movement lengths and angles
  • Scales

• A noise model $P_{x|z}(\ast \mid \ast)$ that specifies how an output $z$ produces a parse $x$
  • Positions (add uniform random noise)
  • Optional borders and contains relations are erased with half chance
  • The indices/orders of shapes are randomly permuted
Example: Visual Concept Learning

Example Program

teleport(position[0], initialOrientation)
draw(shape[0], scale = 1)
move(distance[0], 0deg)
draw(shape[0], scale = scale[0])
move(distance[0], 0deg)
draw(shape[0], scale = scale[0])
Conclusion on Ellis et al., 2015

• Manually separated the deterministic part from the probabilistic part

• Convert the problem into an optimization problem by maximizing likelihood and minimizing encoding lengths
Overview: Saad et al., 2019

• Feras A. Saad, Marco F. Cusumano-Towner, Ulrich Schaechtle, Martin C. Rinard, Vikash K. Mansinghka: Bayesian Synthesis of Probabilistic Programs for Automatic Data Modeling. POPL 19.

• Usage: generate probabilistic programs as generative models of data

• A prior over distribution of programs; conditioning on the observed data, to infer the posterior distribution of the program
Overview of the Framework

From the Original Paper
Details of the Approach

• https://www.youtube.com/watch?v=T5fdUmYJsjM
More on Gaussian Process

• A distribution over functions (from x to y)

• Non-parametric model
  • With infinite many parameters

• The function can be seen as vector which is drawn from a big correlated Gaussian distribution
  • Specified by covariance functions
How to Sample Programs?

• MCMC (Metropolis-Hasting)

• Prior distribution: specified by the PCFG

• Accepting probability: correlates to likelihood
Conclusion on Saad et al., 2019

• A general Bayesian framework to handle different types of synthesis problems
  • Parameterized by the DSL

• Synthesize full programs in Bayesian manner
  • Scalability might be a problem
  • Choosing DSLs and priors are the key
Next Lecture

• Probabilistic Logic Programming