Introduction to probabilistic programming

David Duvenaud and James Lloyd

University of Cambridge

Thanks to
Daniel M Roy (Cambridge)
Roger Grosse (MIT)

How to write a Bayesian modeling paper

1. Write down a generative model in an afternoon

2. Get 2 grad students to implement inference for a month

3. Use technical details of inference to pad half of the paper

CAN WE DO BETTER?

Example: Graphical Models

Application Papers

1. Write down a graphical model
2. Perform inference using general-purpose software
3. Apply to some new problem

Inference papers

1. Identify common structures in graphical models (e.g. chains)
2. Develop efficient inference method
3. Implement in a general-purpose software package

Modeling and inference have been disentangled

Not all models are graphical models

What is the largest class of models available?

Probabilistic Programs

1. A probabilistic program (PP) is any program that can depend on random choices. Can be written in any language that has a (P)RNG.
2. You can specify any (computable) prior by simply writing down a PP that generates samples
3. Any PP implicitly defines a distribution over execution traces

TODO: show a program and a histogram of its output

Probabilistic Programs vs Probabilistic Programming

Once we've defined a prior, what do we want to do with it?

The PP defines $P(D, N, H)$, we choose D to be the subset of variables we observe, H the set of variables we're interested in, and N the set of variables that we're not interested in, so we'll integrate them out. We want to get to $P(H|D)$

Probabilistic Programming

1. Usually refers to doing inference when a PP specifies your prior.
2.

TODO: Show the two possibilities of conditioning in th eprevious program

Can we develop generic inference for all PPs?

Yes - rejection sampling. But can we be more efficient whilst being generic?
Yes. MCMC over execution traces.

PP VIA MCMC

Following Wingate et alia we represent the unconditioned PP as a parameterless function f

Evaluating f results in random choices which are denoted as

$$x_k = f_{k|x_1,\ldots,x_{k-1}} \sim p_{t_k}(.|\theta_k, x_1, \ldots, x_{k-1}).$$

The density / probability of a particular evaluation is then

$$p(x) = \prod_{k=1}^{K} p_{t_k}(x_k|\theta_k, x_1, \ldots, x_{k-1}).$$

We then perform MCMC over the x_k i.e. the execution trace.

MCMC OVER EXECUTION TRACES

1. Select a random $x_k = f_k$ in the execution trace
2. Propose a new value $x'_k \sim K_{t_k}(.|x_k, \theta_k)$
3. Run the program to determine all subsequent choices $(x'_l : l > k)$, reusing current choices where possible
4. Propose moving from the state $(x_1, \ldots, x_K)$ to $(x_1, \ldots, x_{k-1}, x'_k, \ldots, x'_{K'})$
5. Accept the change with the appropriate reversible jump MCMC acceptance probability, this includes terms like
   - 5.1 $K_{t_k}(x'_k|x_k, \theta_k),\ K_{t_k}(x_k|x'_k, \theta_k),\ p_{t_k}(x_k|\theta_k, x_1, \ldots, x_{k-1})$
   - 5.2 $\prod_{i=k}^{K} p_{t_i}(x_i|\theta_i, x_1, \ldots, x_{i-1}),\ \prod_{i=k}^{K'} p_{t'_i}(x'_i|\theta'_i, x_1, \ldots, x_{k-1}, x'_k, \ldots, x'_{i-1})$
   - 5.3 i.e. $\frac{K_{t_k}(x_k|x'_k,\theta_k) \prod_{i=k}^{K'} p_{t'_i}(x'_i|\theta'_i,x_1,\ldots,x_{k-1},x'_k,\ldots,x'_{i-1})}{K_{t_k}(x'_k|x_k,\theta_k) \prod_{i=k}^{K} p_{t_i}(x_i|\theta_i,x_1,\ldots,x_{i-1})}$

FURTHER GENERI INFERENCE METHODS

e.g. HMC, parallel tempering, etc. Remember graphical models (fancy algorithms that work in certain model classes)

PP TIMELINE

Infer.net?

-

Generative model

$$
\begin{aligned}
(\mu_i)_{i=1\ldots k} &\sim_{\text{iid}} \mathcal{N}(0,1) \\
(\pi_i)_{i=1\ldots k} &\sim \text{Dir}(\alpha) \\
\Theta &:= \sum_{i=1}^{k} \pi_i \delta_{\mu_i} \\
(\theta_i)_{i=1\ldots n} &\sim_{\text{iid}} \Theta \\
(x_i)_{i=1\ldots n} &\sim_{\text{iid}} \mathcal{N}(\theta_i, 1)
\end{aligned}
$$

(Pseudo) MATLAB code

```
mu = randn(k,1);
pi = dirichlet(k, alpha);

for i = 1:n
  theta = mu(mnrnd(1,pi));
  x(i)  = theta + randn;
end
```

Change to generative model

$$\Theta := \sum_{i=1}^{k} \pi_i \delta_{\mu_i} \rightarrow \Theta \sim \mathrm{DP}(\alpha, \mathcal{N}(0, 1))$$

(Pseudo) MATLAB code - stick breaking construction

```
sticks = []; atoms = [];
for i = 1:n
  p = rand;
  while p > sum(sticks)
    sticks(end+1) = (1-sum(sticks)) * betarnd(1, alpha);
    atoms(end+1)  = randn;
  end
  theta(i) = atoms(find(cumsum(sticks)>=p, 1, 'first'));
end
x = theta' + randn(n, 1);
```

STOCHASTIC MEMOISATION

1. The stick breaking construction can be applied to any base measure

2. Church provides the function `DPmem` that takes any base measure sampling function and returns a function that samples from a sample from the corresponding Dirichlet process

3. This allows easy specification of many nonparametric models e.g. HDP based models