Introduction to Probabilistic Programming

Johannes Varga

Institute of Logic and Computation, TU Wien, Österreich
Research Unit for Algorithms and Complexity

December 18, 2023

Table of contents

Bayesian Learning

Learn, which model parameters are probable, given data.

Advantages

- Measure for uncertainty
- Use prior knowledge about model and parameters

$\rightarrow$ Sample efficient

Disadvantages

- Can be resource-heavy
- Probabilistic model required

Setting and Terminology

Given:

- Data x
- Probabilisic model $p(x \mid \theta)$
- Prior probability distribution $p(\theta)$

Compute: probability distribution $p(\theta \mid x)$

Bayes theorem

$$\underbrace{p(\theta \mid x)}_{\text{Posterior}} = \frac{\overbrace{p(x \mid \theta)}^{\text{Likelihood}}\ \overbrace{p(\theta)}^{\text{Prior}}}{\underbrace{p(x)}_{\text{Evidence}}}$$

$p(x)$ usually hard to compute, but x is known $\rightarrow$ $p(x)$ is constant

Therefore

$$p(\theta \mid x) \sim f(\theta) = p(x \mid \theta) p(\theta)$$

$f(\theta)$: Joint probability

How to specify f?
How to represent and compute posterior?

Running example: Coinflip

Flip (fair or unfair) coin n times
$\rightarrow$ k times head, $n-k$ times tail
Predict probability θ for head

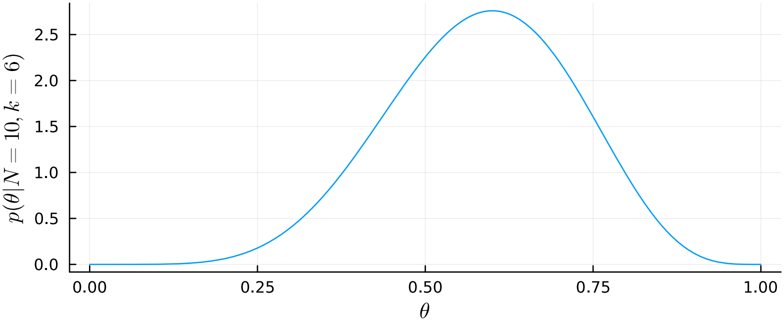


Figure: Analytical solution for $n=10$ and $k=6$

Probabilistic Programs

General framework to specify model via $f(\theta)$

Arbitrary program enriched by

$$\underbrace{A}_{\substack{\text{Name of}\\\text{random variable}}} \sim \underbrace{\mathrm{Normal}}_{\text{Distribution}} (\underbrace{\mu, \sigma^2}_{\text{Parameters}})$$

and the observed value for some random variables

Example: Coinflip in Turing.jl

```
@model function coinflip(heads::AbstractVector{Bool})
    θ ~ Uniform(0, 1)
    for i in eachindex(heads)
        heads[i] ~ Bernoulli(θ)
    end
end
```

Example: Coinflip in Turing.jl

```
@model function coinflip(heads::AbstractVector{Bool})
    θ ~ Uniform(0, 1)
    for i in eachindex(heads)
        heads[i] ~ Bernoulli(θ)
    end
end
```

Prior:

$$p(\theta) = \mathrm{PDF}_{\mathrm{Uniform}}(\theta) = \begin{cases} 1 & \text{, if } 0 \leq \theta < 1 \\ 0 & \text{otherwise} \end{cases}$$

Example: Coinflip in Turing.jl

```
@model function coinflip(heads::AbstractVector{Bool})
    θ ~ Uniform(0, 1)
    for i in eachindex(heads)
        heads[i] ~ Bernoulli(θ)
    end
end
```

Prior:

$$p(\theta) = \mathrm{PDF}_{\mathrm{Uniform}}(\theta) = \begin{cases} 1 & \text{, if } 0 \leq \theta < 1 \\ 0 & \text{otherwise} \end{cases}$$

Likelihood:

$$p(\mathrm{heads}|\theta) = \prod_{i=1}^{n} \mathrm{PDF}_{\mathrm{Bernoulli}}(\mathrm{heads}_i) = p^k p^{n-k}$$

where $n = |\mathrm{heads}|$ and $k = |\mathrm{heads}|_1$

→ Joint Probability is product of PDFs

Joint Probability, Trace

For each $\mathcal{V} \sim \mathcal{D}$, a function is called that

- determines and returns a value v for $\mathcal{V}$,
- appends $(\mathcal{V}, v)$ to the trace, and
- updates joint probability $f(\theta)$ (multiply with PDF).

Determine v:

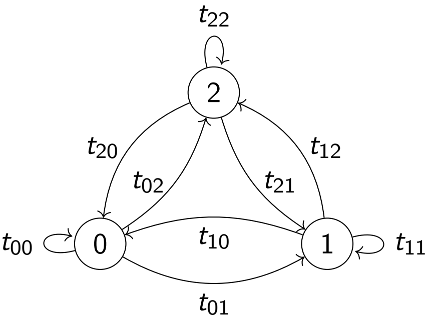
- $\mathcal{V}$ is observed: use observed value
- Otherwise: sampler-dependent, e.g. use proposal distribution or reuse from previous trace

Run results in:

- trace θ
- joint probability $f(\theta)$

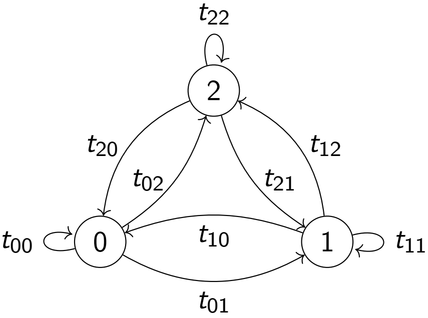
Markov chains

Next state only depends on current state



Markov chains

Next state only depends on current state



s: probability distribution over states

Definition (Steady state)

s is a steady state iff $sT = s$

Theorem

$s_i t_{ij} = s_j t_{ji}$ (aka reversibility) $\Rightarrow$ s is steady state

Monte-Carlo Markov Chains

Bayes Theorem: $p(\theta \mid x) = \frac{p(x|\theta)p(\theta)}{p(x)} = Cf(\theta)$

Problems:

- C unknown
- many dimensions possible, therefore hard to handle

$\rightarrow$ sample from $p(\theta \mid x)$
But: no direct way

Idea: create Markov chain with steady state $p(\theta \mid x)$

Metropolis Hastings

State space: parameter configurations θ
Transitions: use proposal distribution $q(\theta'|\theta)$
Make $p(\theta|x)$ reversible by rejecting proposals with probability $1 - A(\theta'|\theta)$
Reversible if

$$p(\theta|x)q(\theta'|\theta)A(\theta'|\theta) = p(\theta'|x)q(\theta|\theta')A(\theta|\theta')$$

Fulfilled with

$$A(\theta'|\theta) = \min\left(1, \frac{p(\theta'|x)q(\theta|\theta')}{p(\theta|x)q(\theta'|\theta)}\right)$$

Metropolis Hastings

State space: parameter configurations θ
Transitions: use proposal distribution $q(\theta'|\theta)$
Make $p(\theta|x)$ reversible by rejecting proposals
with probability $1 - A(\theta'|\theta)$
Reversible if

$$p(\theta|x)q(\theta'|\theta)A(\theta'|\theta) = p(\theta'|x)q(\theta|\theta')A(\theta|\theta')$$

Fulfilled with

$$A(\theta'|\theta) = \min\left(1, \frac{Cf(\theta')q(\theta|\theta')}{Cf(\theta)q(\theta'|\theta)}\right)$$

Metropolis Hastings

State space: parameter configurations θ
Transitions: use proposal distribution $q(\theta'|\theta)$
Make $p(\theta|x)$ reversible by rejecting proposals
with probability $1 - A(\theta'|\theta)$
Reversible if

$$p(\theta|x)q(\theta'|\theta)A(\theta'|\theta) = p(\theta'|x)q(\theta|\theta')A(\theta|\theta')$$

Fulfilled with

$$A(\theta'|\theta) = \min\left(1, \frac{f(\theta')q(\theta|\theta')}{f(\theta)q(\theta'|\theta)}\right)$$

Metropolis Hastings

State space: parameter configurations θ
Transitions: use proposal distribution $q(\theta'|\theta)$
Make $p(\theta|x)$ reversible by rejecting proposals with probability $1 - A(\theta'|\theta)$
Reversible if

$$p(\theta|x)q(\theta'|\theta)A(\theta'|\theta) = p(\theta'|x)q(\theta|\theta')A(\theta|\theta')$$

Fulfilled with

$$A(\theta'|\theta) = \min\left(1, \frac{f(\theta')q(\theta|\theta')}{f(\theta)q(\theta'|\theta)}\right)$$

Algorithm 4:

Data: Distributions f, q, initial parameters θ, #Iterations n
Result: n samples
$S \leftarrow \{\}$;
for $i \in \{1, \ldots, n\}$ **do**
  sample $\theta' \sim q(\cdot|\theta)$;
  sample $a \sim \mathrm{Uniform}(0, 1)$;
  if $a < \min\left(1, \frac{f(\theta')q(\theta|\theta')}{f(\theta)q(\theta'|\theta)}\right)$ **then**
    $\theta \leftarrow \theta'$
  end
  append θ to S;
end
return S

Metropolis Hastings

State space: parameter configurations θ
Transitions: use proposal distribution $q(\theta'|\theta)$
Make $p(\theta|x)$ reversible by rejecting proposals with probability $1 - A(\theta'|\theta)$
Reversible if

$$p(\theta|x)q(\theta'|\theta)A(\theta'|\theta) = p(\theta'|x)q(\theta|\theta')A(\theta|\theta')$$

Fulfilled with

$$A(\theta'|\theta) = \min\left(1, \frac{f(\theta')q(\theta|\theta')}{f(\theta)q(\theta'|\theta)}\right)$$

$\rightarrow$ Jupyter-Notebook

Algorithm 5:

Data: Distributions f, q, initial parameters θ, #Iterations n
Result: n samples

$S \leftarrow \{\}$;
for $i \in \{1, \ldots, n\}$ **do**
  sample $\theta' \sim q(\cdot|\theta)$;
  sample $a \sim \mathrm{Uniform}(0, 1)$;
  if $a < \min\left(1, \frac{f(\theta')q(\theta|\theta')}{f(\theta)q(\theta'|\theta)}\right)$ **then**
    $\theta \leftarrow \theta'$
  end
  append θ to S;
end
return S

Other inference algorithms

- Hamiltonian Monte Carlo (HMC) and No U-Turn Sampler (NUTS)
- Variational Inference (VI)
- Gibbs sampling

Visualization: `https://chi-feng.github.io/mcmc-demo/app.html`

Predicting User Availabilities

→ Jupyter-Notebook

Captchas

procedure CAPTCHA
  $\nu \sim p(\nu)$ ▷ sample number of letters
  $\kappa \sim p(\kappa)$ ▷ sample kerning value
Generate letters:
  $\Lambda \leftarrow \{\}$
  for $i = 1, \ldots, \nu$ **do**
    $\lambda \sim p(\lambda)$ ▷ sample letter identity
    $\Lambda \leftarrow \text{append}(\Lambda, \lambda)$
Render:
  $\gamma \leftarrow \text{render}(\Lambda, \kappa)$
  $\pi \sim p(\pi)$ ▷ sample noise parameters
  $\gamma \leftarrow \text{noise}(\gamma, \pi)$
  return γ

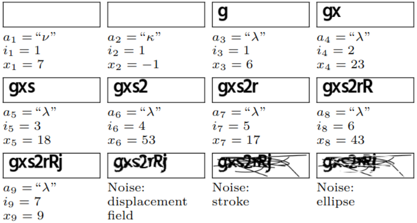
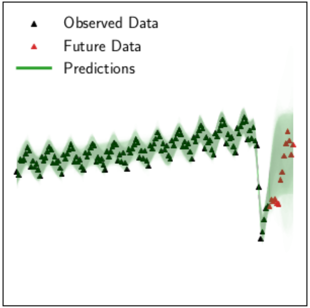


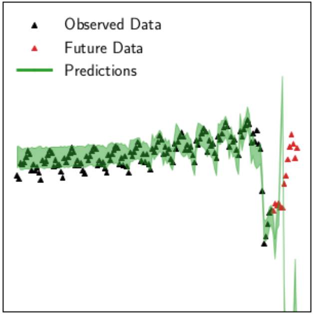
Table 1: Captcha recognition rates.

	Baidu 2011	Baidu 2013	eBay	Yahoo	reCaptcha	Wikipedia	Facebook
Our method	99.8%	99.9%	99.2%	98.4%	96.4%	93.6%	91.0%
Bursztein et al. (2014)	38.68%	55.22%	51.39%	5.33%	22.67%	28.29%	
Starostenko et al. (2015)				91.5%	54.6%		
Gao et al. (2014)	34%			55%	34%		
Gao et al. (2013)		51%		36%			
Goodfellow et al. (2014)					99.8%		
Stark et al. (2015)					90%		

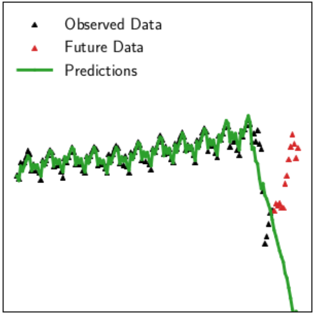
Airline passenger forecast



(a) Gaussian Process DSL
100 Synthesized Structures



(c) Facebook Prophet
With ChangePoint (Custom)



(e) Neural Prophet
With ChangePoint (Custom)

References I

T. A. Le, A. G. Baydin, and F. Wood.
Inference compilation and universal probabilistic programming.
In *Artificial Intelligence and Statistics*, pages 1338–1348. PMLR, 2017.

F. A. K. Saad.
Scalable Structure Learning, Inference, and Analysis with Probabilistic Programs.
PhD thesis, Massachusetts Institute of Technology, 2022.