Frequentist vs Bayesian and Probabilistic Programming

Filipe Rodrigues

Francisco Pereira

DTU Management Engineering
Department of Management Engineering

Outline

- Step back: The big picture so far
- Approximate inference overview
- Frequentist vs Bayesian perspective
- Probabilistic Programming in STAN
- Mixture models in STAN

Step back: The big picture so far

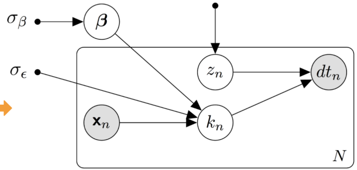
- Probability and statistics recap
  - Probability theory at the center of everything that we do
  - Allows to capture uncertainty
- Probabilistic graphical models (PGMs)
  - Intuitive and compact way of representing the structure of a prob. model
  - Relationships between variables and conditional independencies
  - How the joint distribution factorizes
- Generative processes
  - A "story" of how the observed data was generated
  - Explicit description of how the different variables in the model are related
  - Complementary to PGM representation: more detailed, but less intuitive
- Joint probability distribution and Bayesian inference
  - Joint probability of the model: central object for all computations
  - Bayesian inference: model + data $\rightarrow$ patterns
  - Important concepts: likelihood, prior, posterior, conjugate prior, etc.

Step back: The big picture so far

- Everything is related...

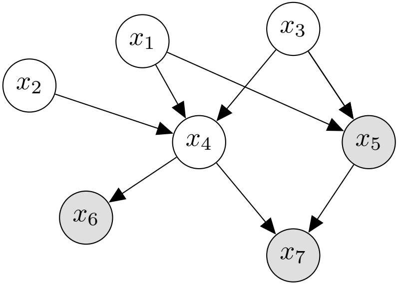
$$p(\boldsymbol{\beta}, \mathbf{z}, \mathbf{k}, \mathbf{dt}) = p(\boldsymbol{\beta}|\sigma_\beta) \prod_{n=1}^{N} p(k_n|\mathbf{x}_n, \boldsymbol{\beta}, \sigma_\epsilon)\, p(z_n|\pi)\, p(dt_n|z_n, k_n)$$

1. Draw a pair of parameters[1], $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, I\sigma_\beta)$
2. For $n = 1..N$
    1. Draw one value for z_n, such that $z_n \sim Bern(\pi)$.
        - If $z_n = 1$, the bus has stopped ($z_n = 0$ otherwise)
        - Distributed as Bernoulli, with parameter π
    2. Draw one value for k_n, such that $k_n \sim \mathcal{N}(\mathbf{x}_n^T\boldsymbol{\beta}, \sigma_\epsilon)$
    3. If $z_n = 1$, $dt_n = k_n$,
        - otherwise $dt_n = 0$



The problem of inference

- **Model + Data → Insights**
- Answer various types of questions about the data by computing the posterior distribution of the latent variables given the observed ones



- Example: $p(x_2|x_5, x_6, x_7) = ?$

The problem of inference

- Inference in general: given a set of latent variables $\mathbf{z} = \{z_m\}_{m=1}^{M}$ and observed variables $\mathbf{x} = \{x_n\}_{n=1}^{N}$, compute $p(\mathbf{z}|\mathbf{x})$
- Two classes of approaches:
  - Exact inference (Bayes' theorem)

$$\overbrace{p(\mathbf{z}|\mathbf{x})}^{\text{posterior}} = \frac{\overbrace{p(\mathbf{x},\mathbf{z})}^{\text{joint}}}{p(\mathbf{x})} = \frac{\overbrace{p(\mathbf{x}|\mathbf{z})}^{\text{likelihood}}\ \overbrace{p(\mathbf{z})}^{\text{prior}}}{\underbrace{p(\mathbf{x})}_{\text{evidence}}}$$

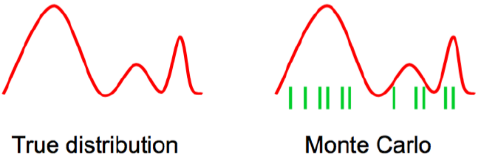
    - For most problems of interest, it is often infeasible to evaluate posterior exactly or to compute expectations with respect to it
  - Approximate Inference
    - STAN uses approximate inference!
    - Stochastic vs. variational methods

Approximate Inference

- Stochastic
- Variational

Approximate Inference

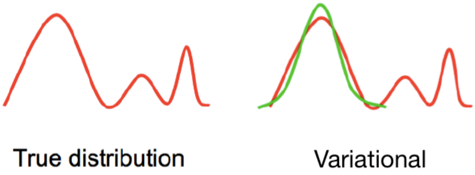
- Stochastic
  - We try to sample from the posterior distribution
  - Samples provide approximate representation of the true posterior
  - We can use samples to compute expectations w.r.t. the posterior
  - Example: Markov Chain Monte Carlo (MCMC) methods



- Variational

Approximate Inference

- Stochastic
- Variational
  - Approximate intractable distribution with a simpler, tractable one
  - Goal: find the parameters of the simpler distribution that make it as similar as possible to the true distribution
  - Similar in what sense?
    - E.g. using Kullback-Leibler (KL) divergence
  - Becomes an optimization problem (of minimizing the difference between *true* and *approximate* distribution)



Approximate Inference

- Stochastic
- Variational

- STAN can use:
  - MCMC (Hamiltonian Monte Carlo or NUTS)
  - Automatic Differentiation Variational Inference (ADVI) - a variational approach with a stochastic component...

Frequentist vs Bayesian perspective

- What is probability?

"The probability that a coin will land heads is 0.5"

But what does this mean?

- Two different interpretations of probability:
- **Frequentist** interpretation
  - Probabilities represent long run frequencies of events - e.g. if we flip the coin many times, we expect it to land heads about half the time
- **Bayesian** interpretation
  - Probability is used to quantify our **uncertainty** about something
  - It is fundamentally related to information rather than repeated trials - e.g. we believe the coin is equally likely to land heads or tails on the next toss
  - Can be used to model our uncertainty about events that do not have long term frequencies! E.g. what is the probability that the polar ice cap will melt by 2025?

Frequentist vs Bayesian perspective

- Frequentist or Bayesian: which one are you? :-)
- Consider the following ML problems:
  - You received a new email. What is the probability that it is spam?
  - Your self-driving car receives data from its cameras. What is the probability that the pedestrian in the sidewalk will cross the road?
  - You keep track of the public transport demand. What is the probability that the demand tomorrow will exceed X given that Metallica is playing nearby?
- In all these cases, the idea of repeated trials does not make sense
- Also, we want to be able to quantify the uncertainty in the predictions!

Note

We are not strictly advocating in favour of the Bayesian perspective. In many cases, a frequentist approach works perfectly fine! And it is often much easier to implement and computationally efficient...

Frequentist vs Bayesian in practice

- Consider that you have a probabilistic model with parameters $\boldsymbol{\theta}$. Given that you observe some data $\mathbf{X}$, you want to estimate $\boldsymbol{\theta}$
- A frequentist approach would be to use **maximum likelihood estimation** (MLE)

$$\boldsymbol{\theta}_{\mathsf{MLE}} = \arg\max_{\boldsymbol{\theta}} \Big(\log p(\mathbf{X}|\boldsymbol{\theta}) \Big)$$

- We can take a step towards a Bayesian approach by considering a prior $p(\boldsymbol{\theta})$, and using **maximum-a-posteriori** (MAP) estimation

$$\boldsymbol{\theta}_{\mathsf{MAP}} = \arg\max_{\boldsymbol{\theta}} \Big(\log p(\mathbf{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \Big)$$

- Both MLE and MAP provide point-estimates of $\boldsymbol{\theta}$! In a fully Bayesian approach, we perform **Bayesian inference** of the posterior distribution over $\boldsymbol{\theta}$

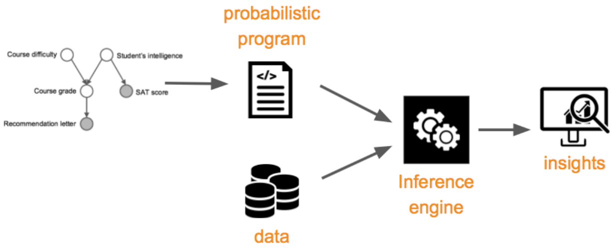
$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{p(\mathbf{X}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{X})}$$

Playtime!

- Frequentist vs Bayesian: a practical example
  - See “4 - Frequentist vs Bayesian.ipynb" notebook
  - Expected duration: 20 minutes

Probabilistic programming

- Allows you to specify a probabilistic model



- Don't need to worry about inference*: it does inference for you!
- Many probabilistic programming languages available:
  - Stan (we will use Stan in this tutorial)
  - Edward
  - Pyro
  - Infer.NET
  - BUGS
  - And so many more...

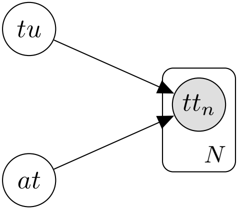
Case study: Analyzing a cyclist's daily travel times

- Suppose that you commute to work everyday by bicycle
- As a methodic cycler, you keep track of your daily travel times (tt)

$$\mathcal{D} = \{tt_1, \ldots, tt_N\}$$

- Based on your collected data, you start building a (simple) PGM to understand your cycling behaviour

tt_n - travel time in the n^{th} day
at - average travel-time
tu - traffic uncertainty



- Making it a bit more formal...

$$tt_n \sim \mathcal{N}(tt_n | at, tu)$$

Case study: Analyzing a cyclist's daily travel times

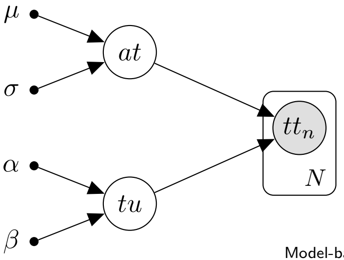
- We have our likelihood

$$tt_n \sim \mathcal{N}(tt_n | at, tu)$$

- Time to specify the priors

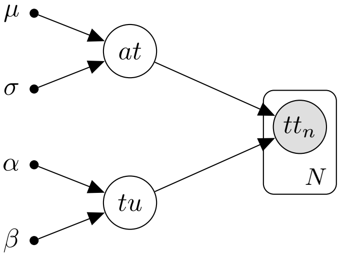
$$at \sim \mathcal{N}(at | \mu, \sigma^2)$$
$$tu \sim \mathcal{IG}(tu | \alpha, \beta)$$

- We chose conjugate priors (for all the advantages explained before)
  However, STAN does not care about conjugacy!
- More complete representation of the model



Case study: Analyzing a cyclist's daily travel times

- Complete representation of the model

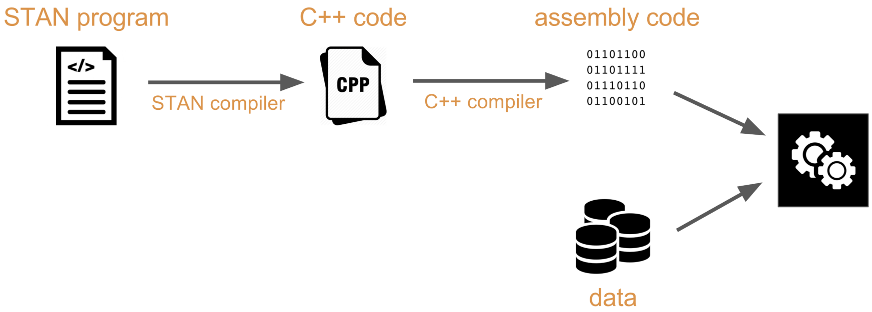


- Corresponding generative process

  1. Draw average travel time $at \sim \mathcal{N}(at|\mu, \sigma^2)$
  2. Draw traffic uncertainty $tu \sim \mathcal{IG}(tu|\alpha, \beta)$
  3. For each day $n \in \{1, \ldots, N\}$
     (a) Draw travel time $tt_n \sim \mathcal{N}(tt_n|at, tu)$

- This generative process description is what STAN relies on!

STAN Workflow



- The top part is completely **seamless** to the user!
- The user needs only to:
    - Specify STAN program (based on the generative process)
    - Assemble data in a Python dictionary
    - Call one of STAN's inference methods
    - Extract and interpret the results

Building blocks of a STAN program

```
functions {
      // Define functions (optional)
}
data {
      // Declare the input data to the model (observed variables)
}
transformed data {
       // Apply transformations to the data (optional)
}
parameters {
       // Declare latent variables in the model (to be inferred)
}
transformed parameters {
       // Apply transformations to the latent variables (optional)
}
model {
       // Specify the model (generative process)
}
generated quantities {
       // Generate data from the model (e.g. predictions for testset)
}
```

Building blocks of a STAN program

- Going back to our case study of cyclist travel times...
- **Data block**: where we declare the input data to the model (observed variables)

```
data {
   int<lower=1> N;  // number of samples
   vector[N] tt;    // observed travel times
}
```

  - We can specify constraints on the inputs (sanity checks)
    E.g. N must be positive

- **Parameters block**: where we declare the latent variables in the model

```
parameters {
   real at;             // average travel time
   real<lower=0> tu;  // traffic uncertainty
}
```

  - We can also specify constraints on the latent variables
    E.g. the traffic uncertainty tu (variance of a Gaussian) must be positive

Building blocks of a STAN program

- **Model block**: where we specify the model (generative process)

```
model {
   at ~ normal(12, 10); // prior on the avg travel times
   tu ~ cauchy(0, 10);  // prior on the traffic uncertainty
   for (n in 1:N) {
      tt[n] ~ normal(at, tu);  // likelihood
   }
}
```

  - We placed an informative prior on at (you should do this whenever you can!)
  - In STAN, the second parameter of "`normal(12, 10)`" is a standard deviation and not a variance! So, the variance is actually $10^2 = 100$
  - *Cauchy* distribution is often recommended as a prior for variances[1]
    - It has a bell-shape like the Gaussian, but fatter tails
    - Due to the positive constraint on tu, this is effectively a *half-Cauchy*
  - We can make the "for" loop more efficient (*vectorization*)

```
tt ~ normal(at, tu);  // likelihood
```

[1] See STAN best practices online

Case study: Analyzing a cyclist's daily travel times

- Putting everything together...

```
data {
   int<lower=1> N;  // number of samples
   vector[N] tt;    // observed travel times
}
parameters {
   real at;             // average travel time
   real<lower=0> tu;  // traffic uncertainty
}
model {
   at ~ normal(12, 10);  // prior on the avg travel times
   tu ~ cauchy(0, 10);   // prior on the traffic uncertainty
   tt ~ normal(at, tu);  // likelihood
}
```

- The model is specified! Let's now look at the data...

Note

"model" block encodes the (log) joint distribution (used by STAN for inference)!

Input data to STAN

- Recall our data block:

```
data {
   int<lower=1> N;  // number of samples
   vector[N] tt;    // observed travel times
}
```

- In Python, we wrap input data in a **dictionary** object

```
cyclist_dat = {'N': 14,
               'tt': [13,17,16,32,12,13,28,12,14,18,36,16,16,31]}
```

- Dictionary keys must match **exactly** the names in the data block declaration!

Inference with STAN

- Recall that STAN provides two types of inference methods
  - Markov chain Monte Carlo (MCMC) - the No U-Turn Sampler (NUTS)
  - Automatic Differentiation Variational Inference (ADVI) - a combination of variational and stochastic...
- We begin by compiling the model (regardless of the inference method)

```
sm = pystan.StanModel(model_code=model_definition)
```

- Run MCMC (NUTS) to compute the posterior distribution of the latent variables (inference)

```
fit = sm.sampling(data=data, iter=1000, chains=4)
```

- Or use ADVI (typically much faster, but still experimental)

```
fit = sm.vb(data=data, iter=10000)
```

Coming soon to STAN...

Riemannian manifold Hamiltonian Monte Carlo (RHMC), expectation propagation, and streaming (stochastic) variational inference, etc.

Interpreting the output of STAN

- We can print a summary of the results using

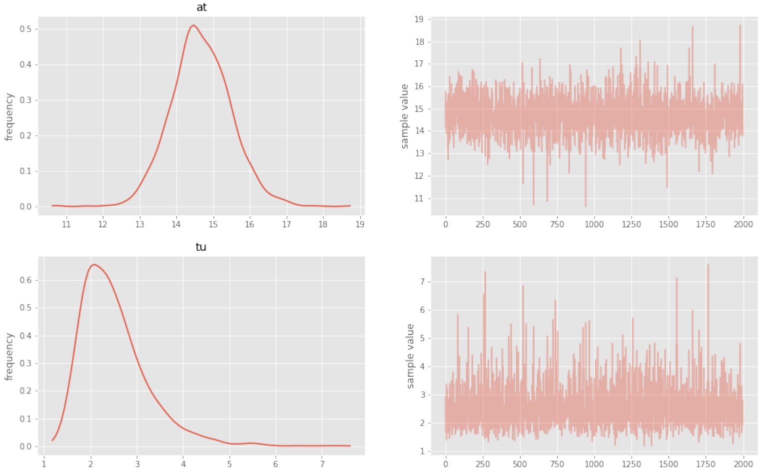
```
print(fit)
```

```
Inference for Stan model: anon_model_257f22c9ec6a2127b7174a37ecde293c.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

       mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff  Rhat
at    14.67    0.03 0.83  13.14  14.15  14.64  15.21  16.33   874   1.0
tu     2.53    0.03 0.75   1.54   1.99   2.38   2.86   4.39   735  1.01
lp__ -12.64    0.04 1.13 -15.63 -13.05 -12.31 -11.86 -11.53   635  1.01

Samples were drawn using NUTS at Mon Jan 29 15:18:04 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

- Make sure to check the diagnostics provided!
  - The value of *Rhat* should be close to 1 (or slightly higher)
  - The number of effective samples (*n_eff*) should not be small

Interpreting the output of STAN

- We can plot the posterior distributions using `fit.plot()`



- Extract the samples from the posterior distribution:

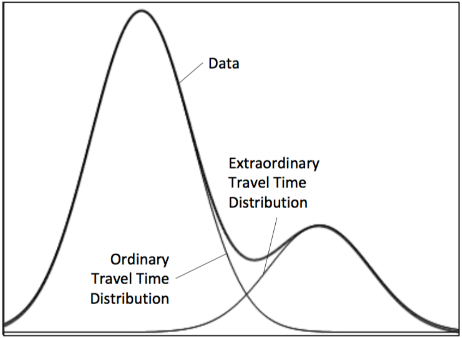
```
samples = fit.extract(permuted=True)
```

Playtime!

- The basics of STAN and PyStan (Section 1 of the notebook)
- First STAN model: Cyclist's daily travel times (Sections 2.1 and 2.2)
  - See "4 - Probabilistic Programming with STAN.ipynb" notebook
  - Only until Section 2.2 (inclusive)!
  - Expected duration: 45 minutes

Case study: Analyzing a cyclist's daily travel times

- A single Gaussian distribution might not be the best choice...
  - Ocasional extraordinary circumstances (e.g. flat tire or a road closed by construction) often add a substantial amount to the usual travel time



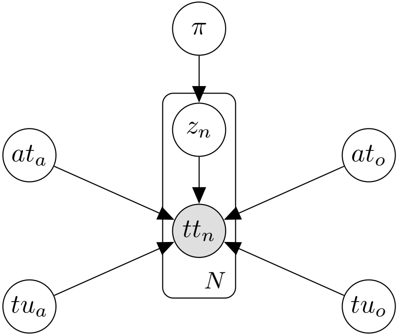
Case study: Analyzing a cyclist's daily travel times

- Mixture model with two Gaussians
  - First Gaussian models the travel time of ordinary trips
$$\mathcal{N}(at_o, tu_o)$$
  - Second Gaussian models abnormal travel times
$$\mathcal{N}(at_a, tu_a)$$
  - Latent Bernoulli variable z_n indicates which mixture component was responsible for the each outcome tt_n
$$z_n \sim \text{Bernoulli}(z_n|\pi)$$
  - Variable π controls mixing proportions, where $\pi \sim \text{Beta}(\pi|\alpha, \beta)$
- The likelihood becomes

$$\begin{aligned} p(tt_n|at_o, tu_o, at_a, tu_a) &= p(z_n = 1)\,\mathcal{N}(at_o, tu_o) + p(z_n = 0)\,\mathcal{N}(at_a, tu_a) \\ &= \pi\,\mathcal{N}(at_o, tu_o) + (1 - \pi)\,\mathcal{N}(at_a, tu_a) \end{aligned}$$

Case study: Analyzing a cyclist's daily travel times

- The graphical model becomes



Case study: Analyzing a cyclist's daily travel times

- Corresponding generative process

  1. Draw average travel time for ordinary days $at_o \sim \mathcal{N}(at_o|\mu_o, \sigma_o^2)$
  2. Draw traffic uncertainty for ordinary days $tu_o \sim \mathcal{IG}(tu_o|\alpha_o, \beta_o)$
  3. Draw average travel time for abnormal days $at_a \sim \mathcal{N}(at_a|\mu_a, \sigma_a^2)$
  4. Draw traffic uncertainty for abnormal days $tu_a \sim \mathcal{IG}(tu_a|\alpha_a, \beta_a)$
  5. Draw mixing proportions $\pi \sim \text{Beta}(\pi|\alpha, \beta)$
  6. For each day $n \in \{1, \ldots, N\}$
     (a) Decide type of day $z_n \sim \text{Bernoulli}(z_n|\pi)$
     (b) If $z_n = 1$
         Draw ordinary travel time $tt_n \sim \mathcal{N}(tt_n|at_o, tu_o)$
     (c) If $z_n = 0$
         Draw abnormal travel time $tt_n \sim \mathcal{N}(tt_n|at_a, tu_a)$

Playtime!

- Mixture model of cyclist's daily travel times (Sections 2.3 and 2.4)
- K-means clustering (Part 2 of the notebook)
  - See "4 - Probabilistic Programming with STAN.ipynb" notebook
  - Expected duration: 45 minutes