Predictive checking / Intro to probabilistic programming with PyMC3

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

February 15, 2021

Announcements

- Modifications to HW1: deadline extended a week, drop your least favorite problem
- Link on Slack to Google Forms survey
- Later this week: road map for future topics, with alternate sources where possible

Outline

Last week:

- One- and multi-parameter models
- Using random sampling to make inferences from models

This week:

- Fixing the basketball model
- Assessing models with predictive checks
- Probabilistic programming: specifying and sampling from a model in PyMC3
- Multilevel models (?)

Posterior predictive checks

Goals

We want to answer the following questions:

- Do the inferences from the model make sense?
- Can the model reproduce features of interest in the original data?

Tools for today:

- Posterior predictive checks

Predictive checking

Basic idea:

- Bayesian models are generative: they give a framework for generating data given parameter values
- So, we can generate data and check it for reasonableness

Predictive checking

Basic idea:

- Bayesian models are generative: they give a framework for generating data given parameter values
- So, we can generate data and check it for reasonableness
- Goals of predictive checking:
  - Prior: confirm that the prior model makes possible predictions
  - Posterior: confirm that the posterior (fitted) model makes predictions that resemble existing data

Example: speed of light measurements

Example

Simon Newcomb's speed-of-light experiment (1882):

- Place a mirror at the base of the Washington Monument
- Flash a light from the US Naval Observatory (where Newcomb worked) about 7442 m away
- Measure travel time

Example

Example (from BDA section 3.3 and 6.3): Newcomb's speed of light measurements.

Model:

$$y_i \sim \mathrm{Normal}(\mu, \sigma)$$
$$p(\mu, \sigma) \propto \sigma^{-1}$$

(Same as the basketball score model from before.)

Assessing the model

How can we assess the accuracy of this model?

- External validation: Compare model predictions to new observations
    - Model estimates the speed of light inaccurately based on current estimates
    - But, this is more because of the experimental design and limitations of data collection

Assessing the model

How can we assess the accuracy of this model?

- External validation: Compare model predictions to new observations
  - Model estimates the speed of light inaccurately based on current estimates
  - But, this is more because of the experimental design and limitations of data collection
- Internal validation: Assess model accuracy / plausibility with the data we already have
  - Do the model predictions look right *relative to the data we have*?

Posterior predictive check for the speed-of-light

Simple posterior predictive check:

- Generate 66 observations from the posterior predictive distribution
- Repeat many times
- View histograms of these sets of 66 observations

Posterior predictive check for the speed-of-light

Simple posterior predictive check:

- Generate 66 observations from the posterior predictive distribution
- Repeat many times
- View histograms of these sets of 66 observations
- If we notice anything odd, drill down on that

Revising the model

What should we do?

- We see that the model reproduces some properties of the data, but the two outliers are not consistent with the model
- Problem: normal distribution has short tails, won't predict extreme outliers
- A normal with larger σ can produce the extreme observations, but doesn't fit the bulk

Updating the model:

- Replace the model likelihood with something that has heavier tails:

$$y_i \sim \mathrm{Normal}(\mu, \sigma)$$
$$y_i \sim \mathrm{StudentT}(\nu, \mu, \sigma)$$
$$y_i \sim \mathrm{Cauchy}(\alpha, \beta)$$

Revising the model

What should we do?

- We see that the model reproduces some properties of the data, but the two outliers are not consistent with the model
- Problem: normal distribution has short tails, won't predict extreme outliers
- A normal with larger σ can produce the extreme observations, but doesn't fit the bulk

Updating the model:

- Replace the model likelihood with something that has heavier tails:

$$y_i \sim \mathrm{Normal}(\mu, \sigma)$$

$$y_i \sim \mathrm{Cauchy}(\alpha, \beta)$$

Revising the model

What should we do?

- We see that the model reproduces some properties of the data, but the two outliers are not consistent with the model
- Problem: normal distribution has short tails, won't predict extreme outliers
- A normal with larger σ can produce the extreme observations, but doesn't fit the bulk

Updating the model:

- Replace the model likelihood with something that has heavier tails:

$$y_i \sim \mathrm{Normal}(\mu, \sigma)$$
$$y_i \sim \mathrm{StudentT}(\nu, \mu, \sigma)$$
$$y_i \sim \mathrm{Cauchy}(\alpha, \beta)$$

Probabilistic programming and PyMC3

A flexible way to sample from models

We have the tools to set up and sample from the heavier-tailed models, but:

- writing ad-hoc sampling code can be error-prone
- this approach does not scale well to models with many parameters
    - 1-3 parameters is not so bad – but even the simplest multilevel model can have 10+, and complex ones can have hundreds or thousands

Probabilistic programming

The basic idea behind probabilistic programming:

- Programmatically specify the probability distributions and relationships between them
- Automatically describe the posterior distribution
- Draw samples from the posterior that can be used for inference

Two especially popular frameworks:

- PyMC3 (what we'll use)
- Stan (popular with R users)

Both of these names reference their close ties to Markov chain methods – more on this soon!

Diagramming two of our models

The same model in mathematical notation

Known variance:

$$y \sim \mathrm{Normal}(\mu, 24)$$
$$\mu \sim \mathrm{Normal}(150, 100)$$

Unknown variance:

$$y \sim \mathrm{Normal}(\mu, \sigma)$$
$$p(\mu) \propto 1$$
$$p(\sigma) \propto \sigma^{-1}$$

The same model in PyMC3

```python
with pm.Model() as normal_model:
    # Prior
    mean = pm.Normal('mu', mu=150, sigma=100)
    # Likelihood
    y = pm.Normal('y', mu=mean, sigma = 24, \
                  observed = modern['Combined'])
```

The same model in PyMC3

```
with pm.Model() as normal_model:
    # Prior
    mean = pm.Normal('mu', mu=150, sigma=100)
    # Likelihood
    y = pm.Normal('y', mu=mean, sigma = 24, \
                  observed = modern['Combined'])
```

Notes:

- Everything happens inside the `with` block
- Variables are abstract/symbolic (which is why they are given string names)

What can we do with the model?

The main use case is to sample from the posterior distribution:

- Automatically apply MCMC samplers (more on the details later)
- For simple models, default sampling is good

Let's try it out on some familiar models:

- normal model for basketball scores
- normal and non-normal model for speed of light