UofT ISSC Bayesian Workshop

Eric Zhu

All pictures taken by me!

About me

- Just finished 3rd year CS/Stats
- Currently PEY @ AMD
  - Working on HIP Compute
- Interests in ML/Bayesian
- STA365 was super interesting
  - A lot of this is inspired by/what I learned in STA365, taught by Prof. Dan Simpson

Overview

- Workshop is an intro to Bayesian Stats
    - Is not a workshop on creating GANs/theory/derivations
- **Roadmap:**
    - Intro (25 minutes)
        - Probability/notation Recap
        - Bayesian Motivation
        - Priors
    - A bit on Stan (~20 minutes)
    - Modelling/hands on (rest of the time)

[ThisPersonDoesNotExist.com uses AI to generate endless fake faces - The Verge]

Introduction

Notation Recap

- $\tilde{y}$ means predicted values
- $\propto$ means proportional to
- Vectors and matrices may at times be bolded
  - Differentiate from scalars
  - $\boldsymbol{X}$ would denote the design matrix
- Integration (summation) over a parameter space:

$$\int_{\theta} p(\theta)\partial\theta = 1$$

- Matrices denoted by number of rows and columns
- All vectors are the usual column vectors, i.e., an n by 1 matrix
- $y^{(i)}$, $x^{(i)}$denotes the i-th y value and i-th observation (row) respectively

Probability Recap

A few key concepts will be important for this workshop:

- Bayes rule:

$$p(\,A \mid B\,) \;=\; \frac{p(B \mid A) \cdot p(A)}{p(B)}$$

- Marginalization:

$$p(y) = \int_\theta p(y, \theta) \partial\theta = \int_\theta p(y \mid \theta) p(\theta) \partial\theta$$

- Likelihood Function:

$$L(\theta) \;=\; \prod_{i=1}^{N} p(y_i \mid \theta)$$

What is Bayes?

- Bayesian Statistics applies Bayes' rule to get distributions for the parameters!
- Since we either have parameters or data, we instead posit that parameters vary, while the data is fixed
    - "Data comes on a spreadsheet"
    - Need priors!
- Allows us to create **generative models**
    - Your predictions can be pulled from a reliable/sensible distribution

What is Bayes - interpretations

- Frequentists measure the long-run frequency of an event
  - A fair coin has an expected value of 0.5
    - We expect half of the number tosses to be heads
    - Asymptotic
  - A parameter is a fixed value, so we estimate these fixed values
    - Maximum Likelihood Estimation
- Bayesians incorporate *a priori* beliefs (information)
  - A Bayesian may conclude infer the same as a frequentist
  - Or something different...
    - Perhaps one guy believes the coin isn't totally fair
  - We can also flip the question on its head
    - Examine the "fairness" of the coin given some flips
    - Compute a posterior distribution for p given y
      - p being the probability of success

Applying Bayes

- Our goal is to obtain a posterior distribution for our parameters
    - Computed by hand or numerically (MCMC, INLA)
    - Drop the normalizing constant

$$p(\theta \mid y) \propto p(y \mid \theta) \cdot p(\theta)$$

- From there, we can obtain a marginal distribution
    - Predictive distributions - come back to this later
        - Done for both prior and posterior distributions
- Rather slow in practice due to large number of samples/complexity

Maximum Likelihood Estimation

- Why not just use MLE then?
  - It's simple - just take the partial derivatives
  - Clearly works to some degree
    - Neural Networks often minimize some negative log likelihood
  - Unbiased estimator at times
    - Not really applicable under the Bayesian Framework
- Suppose we have a dataset with *n* observations:
  - $X^T = [x_1, x_2, x_3, \ldots, x_{n-1}, x_n]$ - *n* fair coin flips
    - Assume 80% of flips are heads
    - MLE is the proportion of heads
  - Risk overfitting
    - Data sparsity
  - **Bet on sparsity principle**
    - 79-91.pdf (ssc.ca)
    - Variable selection in high-dimensional genetic data. Sahir Bhatnagar, McGill University. - YouTube
    - Statistical Analysis on Sparse data? - Cross Validated (stackexchange.com)

Maximum Likelihood Estimation

- Regularization would help!
  - Minimize the penalized objective (cost) function
    - Use L_2 or L_1

$$J^{\beta}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}\mathcal{L}\left(y^{(i)}, t^{(i)}\right) + \lambda\mathcal{R}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}\left(\mathbf{w}^{\top}\mathbf{x}^{(i)} - t^{(\mathrm{i})}\right)^2 + \lambda\mathcal{R}(\mathbf{w})$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(\mathbf{w}^{\top}\mathbf{x}^{(i)} - t^{(\mathrm{i})}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{D} w_j^2$$

$$= \underbrace{\frac{1}{2N}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|_2^2}_{\text{MSE } (-\text{log-likelihood})} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{L_2 \text{ norm}}$$

- Turns out that regularization is analogous to a prior!
  - Interpreting Regularization as a Bayesian Prior – Rohan Varma – Software Engineer @ Facebook
  - Bayesian Interpretations of Regularization (mit.edu)
  - A Probabilistic Interpretation of Regularization | Bounded Rationality (bjlkeng.github.io)

Creating Priors

- Analogous to regularization:
  - Do not want too constrained of priors
    - Possibly too informative
  - Do not want too loose of priors
    - Possibly too uninformative
  - Meet in the middle
    - **Weakly informative priors!**
- Incorporate some of your information into your priors
  - Use diagnostics to check for behaviour
  - Prior/Posterior Predictive Plots
  - PSIS
  - Comparison of prior/posterior

Predictions

Prior predictive distribution:

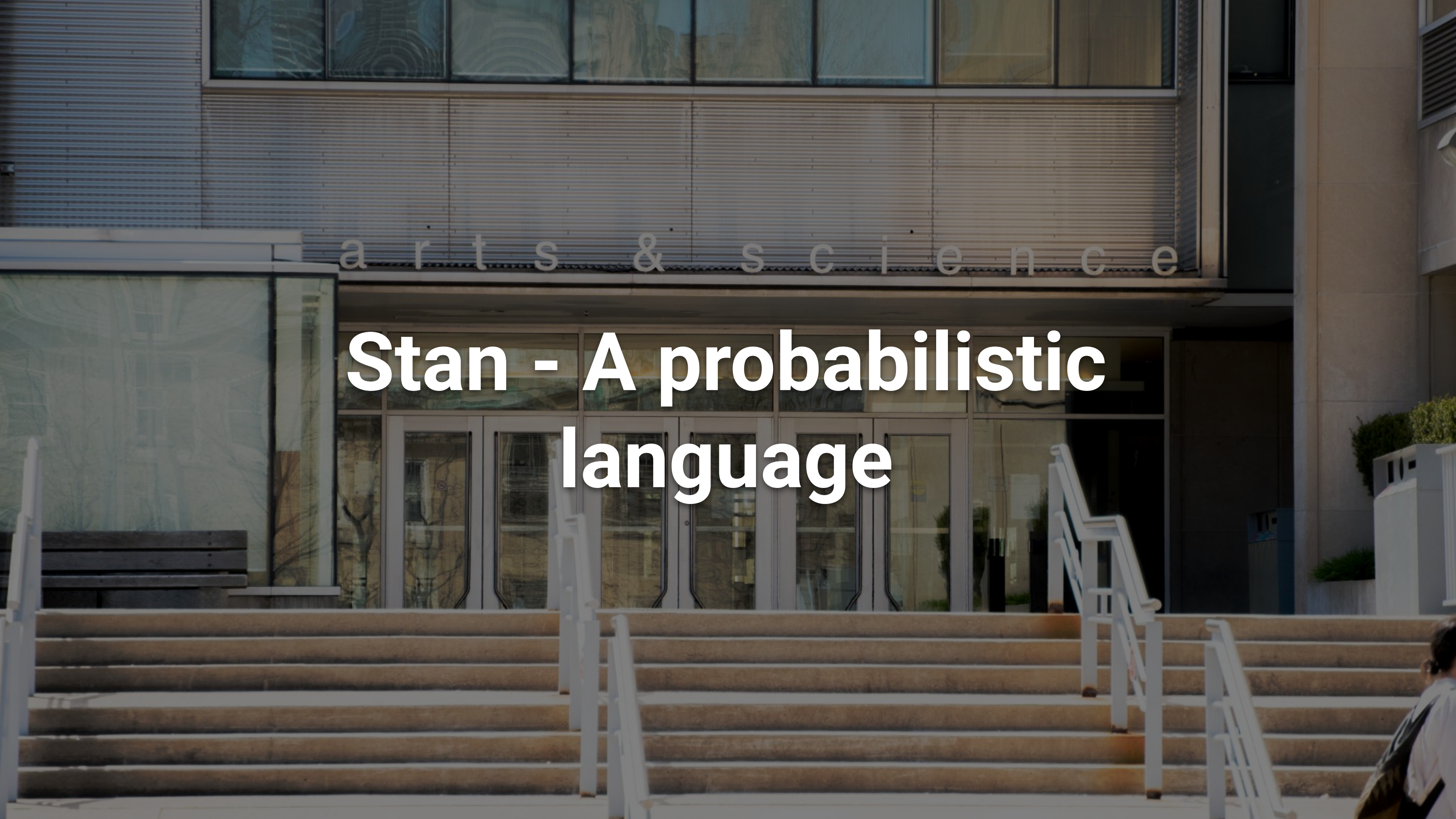
$$p(y) \; = \; \int_\theta p(y,\,\theta)\partial\theta = \int_\theta p(y \,|\, \theta) \underbrace{p(\theta)}_{\text{prior distribution}} \partial\theta$$

Posterior predictive distribution

$$p(\tilde{y} \,|\, y) \; = \; \int_\theta p(\tilde{y},\,\theta \,|\, y)\partial\theta \; = \int_\theta p(\tilde{y} \,|\, \theta, y) \underbrace{p(\theta \,|\, y)}_{\text{posterior distribution}} \partial\theta$$

$$= \int_\theta p(\tilde{y} \,|\, \theta) p(\theta \,|\, y)\partial\theta$$

- Note that predictions (*y-tilde)* and *y* are conditionally independent given theta
- We always want to integrate out the unknown parameter space
    - Take lots of samples



Stan - A probabilistic language

What is Stan?

- A *strongly-typed* language that allows you to obtain accurate posterior distributions!
  - MCMC sampler for arbitrary distributions
- Many interfaces (not exhaustive)
  - Cmdstan
  - RStan
  - PyStan
  - CmdstanR
  - Rstanarm
  - brms
- Used in many fields, especially more in inferential applications

Data Types

- For discrete variables - `int`
    - Number of samples in dataset
- For continuous/floating point variables - `real`
    - Prior choices, parameters
- Array-like
    - Integer arrays - `int variable_name [N]`
    - Real number arrays - `real variable_name [N]`
    - A column vector - `vector[N] variable_name`
    - An *N* by *M* matrix - `matrix[N, M] variable_name`
    - Generic syntax - `array[N, M] data_type variable_name`
- Bound decorators
    - Lower bound - `<lower = a>`
    - Upper bound - `<upper = b>`
    - Above and below - `<lower = a, upper = b>`
- For more info - Stan docs
    - [5.1 Overview of data types | Stan Reference Manual (mc-stan.org)]

Blocks

Stan uses “blocks”, which do specific actions, e.g., compute posterior distributions

`functions {...}` – User defined functions

`data {...}` – Data input, must match with input from interface

`parameters {...}` – The parameters we calculate posterior distributions for

`transformed parameters {...}` – Parameters that are more complex, e.g., $\boldsymbol{\mu} = \mathbf{X}\mathbf{w}$

`model {...}` – Model code, i.e., priors and likelihood

`generated quantities {...}` – Put prediction code or other code here, e.g., code using fit parameter values

A Basic Model by the Blocks

```
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] t;
  int<lower=0, upper = 1> only_prior;
}
```

```
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
```

```
model {
  // priors
  alpha ~ normal(mu_alpha, tau_alpha);
  beta  ~ normal(mu_beta, tau_beta);
  sigma ~ normal(mu_sigma, tau_sigma);
  // likelihood
  if (only_prior == 0){
    y ~ normal(alpha + beta*t, sigma);
  }
}
```

```
transformed parameters{
  vector[N] mu = alpha + beta*t
}
```

```
generated quantities {
  vector[N] log_lik;
  vector[N] y_pred;
  for (i in 1:N) {
    log_lik[i] = normal_lpdf(y[i] | mu[i], sigma);
    y_pred[i] = normal_rng(mu[i], sigma);
  }
}
```

Predictions

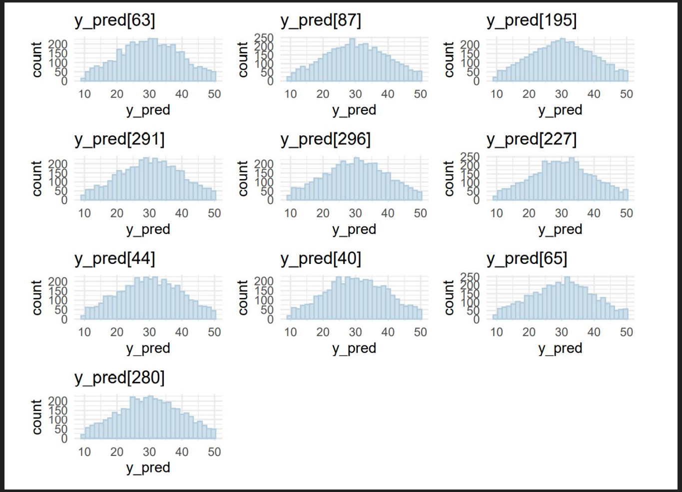
Recall the posterior predictive distribution:

$$p(\tilde{y} \mid y) = \int_\theta p(\tilde{y}, \theta \mid y) \partial\theta = \int_\theta p(\tilde{y} \mid \theta) p(\theta \mid y) \partial\theta$$

- We must integrate over all possible theta values
  - Consider all of the parameter space
- Equivalent to looking at the joint distribution
  - Ignore the dimensions we don't care about
  - [Sampling from marginal distribution using conditional distribution? - Cross Validated (stackexchange.com)]()
- Algorithm is:
  - Sample from the posterior distribution
  - Use that sampled parameter in drawing samples from the appropriate distribution
    - Essentially getting a joint sample - these samples correspond to each other
    - This is your prediction

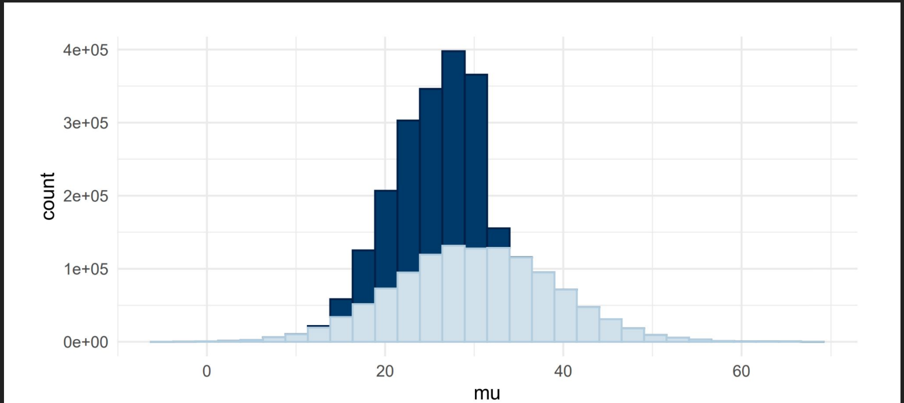
Interpreting Results

- First plot your prior and posterior predictive distributions
- Do they look right?
    - Check the scales
    - Check for bounds
    - Check the shape of the distributions
    - Are they super different from your beliefs?
- **Misspecification**!



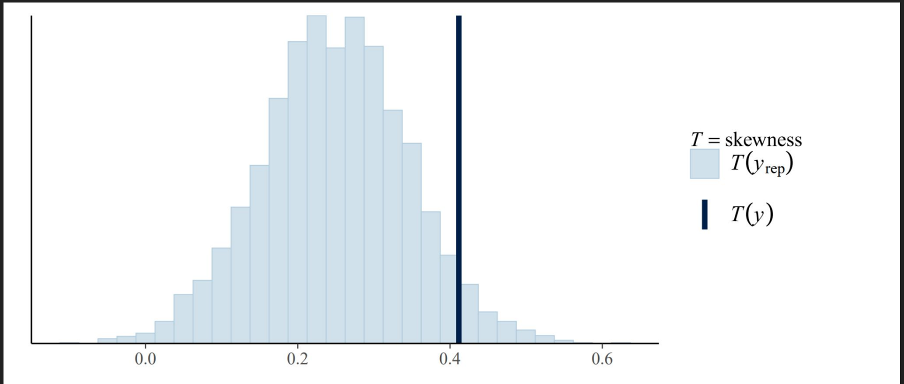
Interpreting Results

- Second check how the posterior compares to the prior
  - The posterior should overlap!
    - No overlapping mass is evidence for *prior data conflict*
  - Contract within the prior
    - The prior regularizes the posterior



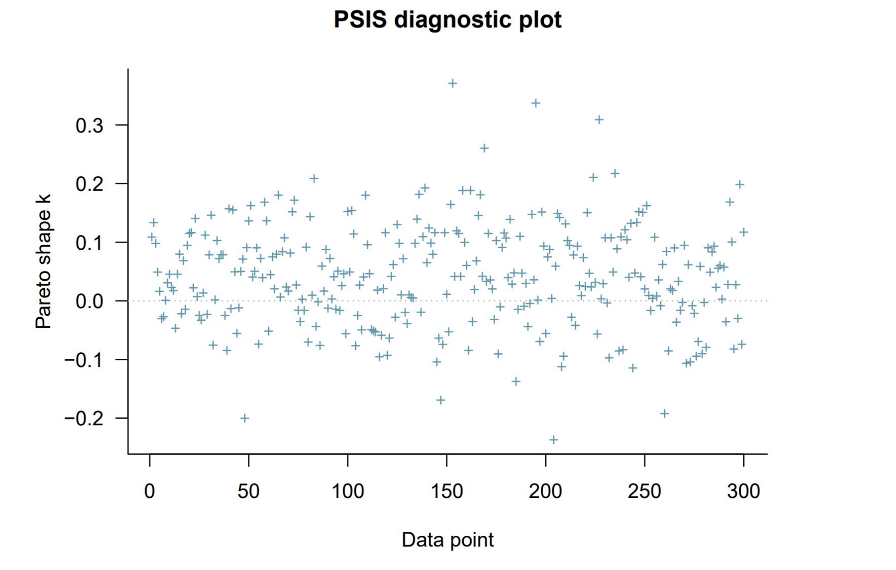
Interpreting Results

- Third, check the ancillary test statistics
  - Ancillary test statistics are those that don't change much with different samples from a distribution
  - Normal distribution ancillary test statistics:
    - Skewness
    - Min, Max



Interpreting Results

- Fourth, check the predictive performance
  - LOO-CV to evaluate predictive performance
  - Look at the PSIS plot
    - K-hat values function similarly to leverage
  - K-hat should be under 0.5 (negatives are ok!)
    - High k-hat are suspect - possibly a bad fit or problems with data



Interpreting Results

- Many more ways to evaluate your model/fit
  - Graphical
  - Summaries
    - Check relevant statistics (r-hat, ess-bulk, etc...)
- Graphical
  - Bayesian workflow paper - Prof. Andrew Gelman (also Prof. Dan Simpson)
    - Bayesian Workflow (arxiv.org)
  - PPC density overlays
    - Plots predictions overlaid with density estimate of sample
  - Time course plots
    - Relevant in certain scenarios
  - Arviz Package:
    - API Reference — ArviZ dev documentation (arviz-devs.github.io)
    - Comparison plots
    - Joint plots

Conclusions

Key Takeaways

Pros

- **Generative modelling!**
  - A focus on the distributions
- Incorporate more uncertainty into inference
- Control over your information
- Update your model with beliefs

Cons

- **Slow!**
- Need reliable information
- Possibly more work/probabilistic considerations
- Less of a community/current WIP

Next steps

- If you found this cool:
  - **STA365**, CSC412/STA414, STA465
  - **STA303**
- Practise coding in R (or Python)!
  - UofT uses R in Bayesian Courses
- Check out generative models!
  - Bayesian spatial models
  - MRP with poststratification
  - VAE (Variational Autoencoder)
  - GANS (Generative Adversarial Network)
- MCMC Lecture - YouTube

- If you didn't find this cool:
  - **STA314**, STA414/CSC412
  - **STA303**

Contact Info

- Website: https://ezhu.build
- LinkedIn: https://www.linkedin.com/in/eric-z-zhu/
- Github: https://github.com/GreatArcStudios
- UofT mail: e.zhu@mail.utoronto.ca