Bayesian Statistics with Python

An Example Method:
No Resampling Necessary

Charles Lindsey
Principal Data Scientist, Aptos

Agenda

- Bayesian Statistics Exploration
  - Prior
  - Confidence
  - Observed Data
  - Posterior
  - PyStan
  - Optimization
  - Simulation
  - Variances

- We will start in this powerpoint and move into Jupyter as our exploration becomes more involved.

How Modeling Works - Prior

- We use a probabilistic model to study real world phenomena.
- Parameters dictate how model the works.
- In the Bayesian framework, these parameters: cannibalization, elasticity, latent ability, etc. are random variables.
- Before we see the real world data we are modeling, we already have assumptions about these parameters.
- These assumptions, this **prior** information tells us about the **distribution** of these parameters.
  - Where is elasticity centered?
  - What is the variance of latent ability?
  - How **confident** are we in salestype lift?

STAN

- STAN is a probabilistic programming language for Bayesian Statistical inference.
- It is written in C++.
- There are several packages that allow interface with STAN in other languages.
- PyStan is one which allows interface with Python.
- Parameters are specified in a dedicated block in STAN:

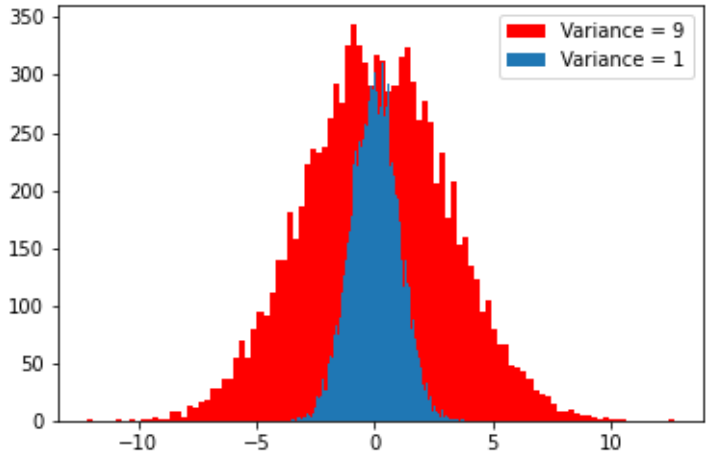
```
parameters {
    real<lower=0> alpha;
    real beta;
}
```

How Modeling Works - Confidence

- What did we mean when we used "confident"?
- Every question should be answered by:
  - An answer **x**
  - A measure of confidence **c** in that answer **x**.
- How certain are we in the answer that we provide?
- What is our degree of belief?
- There are a variety of ways that confidence can be measured and that these measures can be conveyed.

How Modeling Works - Confidence

- Here is an example.
- Here are two histograms for Normal random variables x1 and x2 with mean 0 and variance 1 and 9. We say that x1 ~ N(0,1), x2 ~ N(0,9).



- We have more confidence that x1 is close to 0 than x2 is close to 0.

Confidence Interval

- Using variance to summarize confidence is useful.
- But the fact that variances can be any positive number can lead to subjective interpretation.
- Probabilities are bounded on [0,1] so they can be interpreted more objectively.
- A 1-α % confidence interval (formally a credible interval) (l,u) for parameter y satisfies $Pr(l < y < u) = 1 - \alpha$.
- We see that the confidence interval gets larger as the variance gets larger.

How Modeling Works: Observed Data

- The parameters that we have prior information on **parameterize** the population that we sample the observed data from.
- For example, the parameter μ has a N(0,1) distribution. Conditional on μ, we observe a sample of size N from a N(μ,1) distribution.
- The observed data gives us more information on the parameters than our initial (prior) assumptions.
- We will be able to evaluate the confidence in the parameters based on the new information.
- To contrast with the prior information, this new information that uses the observed data and prior is called **posterior**.

Observed Data and STAN

- Observed data is specified in a dedicated block in STAN:

```
data {
    int<lower=0> N;
    vector[N] x;
    vector[N] y;
}
```

Posterior Density

- The parameters and their confidence are evaluated using the parameter's probability distributions, characterized by their density.
- $\pi(\boldsymbol{\theta})$ is the prior density for the parameters $\boldsymbol{\theta}$.
- $\pi(\boldsymbol{\theta}|\mathbf{x})$ is the posterior density for the parameters $\boldsymbol{\theta}$.
- This is the density of $\boldsymbol{\theta}$ conditional on the observed data $\mathbf{x}$.
- From the posterior density we would like to learn the values that the parameters are centered at and how confident we are that they are close to those values.
- This can be done in many situations by using simulation-based methods to sample from the posterior distribution.
- But when you have enough data, you can use numeric optimization to estimate the central values of the parameters and their confidence.

Posterior in STAN

- The likelihood of the observed data, $L(\boldsymbol{\theta}|\mathbf{x})$ is the probability density of the observed data conditional on the parameters. You could view this as $\pi(\mathbf{x}|\boldsymbol{\theta})$.
- The posterior density is proportional to the product of the likelihood and the prior density.
- $\pi(\boldsymbol{\theta}|\mathbf{x}) \propto L(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})$
- The prior and likelihood are specified in the model block in STAN:

```
model {
    alpha ~ chi_square(4);
    beta ~ normal(1,1);
    y ~ normal(alpha + beta * x,1);
}
```

Posterior in STAN

- The likelihood of the observed data, $L(\boldsymbol{\theta}|\mathbf{x})$ is the probability density of the observed data conditional on the parameters. You could view this as $\pi(\mathbf{x}|\boldsymbol{\theta})$.
- The posterior density is proportional to the product of the likelihood and the prior density.
- $\pi(\boldsymbol{\theta}|\mathbf{x}) \propto L(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})$
- The prior and likelihood are specified in the model block in STAN:

```
model {
    alpha ~ chi_square(4);
    beta ~ normal(1,1);
    y ~ normal(alpha + beta * x,1);
}
```

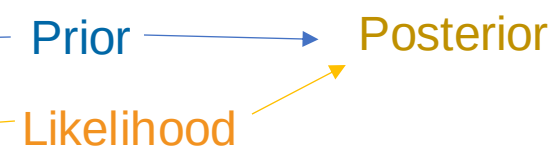
Prior

Likelihood

Posterior in STAN

- The likelihood of the observed data, $L(\boldsymbol{\theta}|\mathbf{x})$ is the probability density of the observed data conditional on the parameters. You could view this as $\pi(\mathbf{x}|\boldsymbol{\theta})$.
- The posterior density is proportional to the product of the likelihood and the prior density.
- $\pi(\boldsymbol{\theta}|\mathbf{x}) \propto L(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})$
- The prior and likelihood are specified in the model block in STAN:

```
model {
    alpha ~ chi_square(4);
    beta ~ normal(1,1);
    y ~ normal(alpha + beta * x,1);
}
```



PyStan

- Here is how the model and data are specified in PyStan:

[39]:
```
model = """
data {
    int<lower=0> N;
    vector[N] x;
    vector[N] y;
}

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

model {
    alpha ~ chi_square(4);
    beta ~ normal(1,1);
    y ~ normal(alpha + beta * x,1);
}
"""
with suppress_stdout_stderr():
    stan_model = pystan.StanModel(model_code=model)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_a0d1a455dfb70045e204be5bae7f04b9 NOW.
```

How modeling works

- If we want to maximize the posterior density with respect to $\boldsymbol{\theta}$, we only need to maximize this product. For numerical stability we can look at its natural logarithm as well.
- The maximum corresponds to the posterior mode, $\hat{\theta}$ the point in the parameter space that has the highest posterior density.
- This is a good point estimate for the parameter, especially when we have a distribution like we saw in the earlier example.
- We use a Newton or quasi-Newton numerical maximization technique to get an estimate of the posterior mode,

```
sm = stan_model
optimizing_fit = sm.optimizing(stan_data,algorithm="Newton",init='0')
```

How modeling works

- At the posterior mode estimate, we can approximate the posterior with:

$$\log \pi(\boldsymbol{\theta}|\mathbf{x}) \approx \log \pi(\hat{\boldsymbol{\theta}}|\mathbf{x}) + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T S(\boldsymbol{\theta})|_{\theta=\hat{\theta}} + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T H(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$$

- $S(\boldsymbol{\theta})$, the score function is the derivative of the log posterior with respect to the parameters $\boldsymbol{\theta}$.
- $H(\boldsymbol{\theta})$, the Hessian function is the second-derivative of the log posterior with respect to the parameters $\boldsymbol{\theta}$.
- Our approximation gets more accurate as the amount of observed data increases.
- Using Calculus, $S(\boldsymbol{\theta})$ becomes 0 at the mode. So by exponentiating we have:

$$\pi(\boldsymbol{\theta}|\mathbf{x}) \approx \pi(\hat{\boldsymbol{\theta}}|\mathbf{x}) \exp\left(\frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T H(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})\right)$$

Confidence

- So the posterior is approximately proportional to a multivariate normal density. So we can say

$$\boldsymbol{\theta}|x \approx_D N(\widehat{\boldsymbol{\theta}}, -H(\widehat{\boldsymbol{\theta}})^{-1})$$

- This key result tells us the values that the parameters are centered at and it gives us rules that we can use to learn how confident we are that they are close to those values.
- We can get posterior variances and standard deviations from $-H(\widehat{\boldsymbol{\theta}})^{-1}$
- It is easy to get confidence intervals using the quantile values of the normal distribution and components of the posterior mean and variance. We'll show how a little later.

PyStan

- Unfortunately, PyStan will not return the Hessian for us.
- However, we can use numerical differentation.
- Let's go into Jupyter for more details.
- and to see how perform simulation as well!

Simulation

- We can corroborate our theoretical results by simulation.
- We will sample **draws** = 1000 draws of our parameters **α** and **β** from their prior distribution.
- At each prior draw we will draw a sample of **n** = 600 of the observed data variables **x** and **y**.
- To check whether the posterior distributions are normal with means at the posterior modes and variances given by the inverse Hessian, we can compare to the resampling generated posterior distribution from **.sampling()** in PyStan.
- We use MCMC for resampling to get **L** = 500 posterior samples.

Thank You!