SDS 383D: Modeling II **Spring 2017**

Section 3: Bayesian GLMs

3.1 Modeling non-Gaussian observations

So far, we've assumed real-valued observations. In this setting, our likelihood model is a univariate normal, parametrized by a mean $x_i^T\beta$ and some precision that does not directly depend on the value of x_i. In general, $x_i^T\beta$ will take values in $\mathbb{R}$

If we don't want to use a Gaussian likelihood, we typically won't be able to parametrize our data using a real-valued parameter. Instead, we must transform it via an appropriate link function. This is, in essence, the generalized linear model.

As a first step into other types of data, let's consider binary valued observations. Here, the natural likelihood model is a Bernoulli random variable; however we cannot directly parametrize this by $x_i^T\beta$. Instead, we must transform $x_i^T\beta$ to lie between 0 and 1 via some function $g^{-1} : \mathbb{R} \to (0, 1)$. We can then write a linear model as

$$\begin{aligned} y_i|p_i \sim& \text{Bernoulli}(p_i) \\ p_i =& g^{-1}(x_i^T\beta) \\ \beta|\theta \sim& \pi_\theta(\beta) \end{aligned}$$

where $\pi_\theta(\beta)$ is our choice of prior on β. Unfortunately, there is no choice of prior here that makes the model conjugate.

Let's start off with a normal prior on β. One appropriate function for g^{-1} is the inverse CDF of the normal distribution – known as the probit function. This is equivalent to assuming our data are generated according to

$$\begin{aligned} y_i =& \begin{cases} 1 & if z > 0 \\ 0 & \text{otherwise} \end{cases} \\ z_i \sim& \text{N}(x_i^T\beta, \tau^2) \end{aligned}$$

If we put a normal-inverse gamma prior on β and τ, then we have a *latent* regression model on the (x_i, z_i) pairs, that is idential to what we had before! Conditioned on the z_i, we can easily sample values for β and τ.

Exercise 3.1 *To complete our Gibbs sampler, we must specify the conditional distribution $p(z_i|x_i, y_i, \beta, \tau)$. Write down the form of this conditional distribution, and write a Gibbs sampler to sample from the posterior distribution. Test it on the dataset* `pima.csv`*, which contains diabetes information for women of Pima indian heritage. The dataset is from National Institute of Diabetes and Digestive and Kidney Diseases, full information and explanation of variables is available at* `http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes`*.*

Solution

$$p(z_i|x_i, y_i, \beta, \tau) \propto p(y_i|z_i, \beta, \tau)p(z_i|x_i, \beta, \tau) \propto \begin{cases} p(z_i|x_i, \beta, \tau) \,\text{with} z_i > 0 & if y_i = 1 \\ p(z_i|x_i, \beta, \tau) \,\text{with} z_i < 0 & if y_i = 0 \end{cases}$$

Which is a truncated normal distribution at 0 and sampling from the positive values and the negative values respectively.

By comparing the labels and the prediction, the learned model does not have any mis prediction when using the whole data for both training and testing. Better analysis of accuracy can be obtained by running cross-validation.
(*The code can be found on Github under python directory with name Ch3_1.py*)
Solution End

Another choice for $g^{-1}(\theta)$ might be the logit function, $\frac{1}{1+e^{-x^T\beta}}$. In this case, it's less obvious to see how we can construct an auxilliary variable representation (it's not impossible! See **?**. But for now, we'll assume we haven't come up with something). So, we're stuck with working with the posterior distribution over β.

Exercise 3.2 *Sadly, the posterior isn't in a "known" form. As a starting point, let's find the maximum a posteriori estimator (MAP). The dataset "titantic.csv" contains survival data from the Titanic; we're going to look at probability of survival as a function of age. For now, we're going to assume the intercept of our regression is zero – i.e. that β is a scalar. Write a function (that can use a black-box optimizer! No need to reinvent the wheel. It shouldn't be a long function) to estimate the MAP of β. Note that the MAP corresponds to the frequentist estimator using a ridge regularization penalty.*

Solution:
The likelihood is given by:

$$\text{likelihood} = \prod_{i=1}^{n} P(y_i|x_i, \beta)$$

The log likelihood is:

$$\text{log-likelihood} = -\sum_{i=1}^{n} \log(1 + e^{-y_i x^T \beta})$$

the MAP objective:

$$\min \sum_{i=1}^{n} \log(1 + e^{-y_i x^T \beta}) + \frac{1}{2\sigma^2}(\beta - \mu)^2$$

$$\hat{\beta} \approx -0.011014$$

(*The code can be found on Github under python directory with name Ch3_2-4.py: this code computes the MAP estimator, evaluate the posterior distribution, and compares it to the laplace approximation.*)

Solution End

Exercise 3.3 *OK, we don't know how to sample from the posterior, but we can at least look at it. Write a function to calculate the posterior pdf $p(\beta|\mathbf{x}, \mathbf{y}, \mu, \sigma^2)$, for some reasonable hyperparameter values μ and θ (up to a normalizing constant is fine!). Plot over a reasonable range of β (your MAP from the last question should give you a hint of a reasonable range).*

Solution:

$$p(\beta|\mathbf{x},\mathbf{y},\mu,\sigma^2) \propto \left\{\prod_{i=1}^{n}\frac{1}{1+e^{-y_i x_i^T\beta}}\right\} e^{\frac{1}{2\sigma^2}(\beta-\mu)^2}$$

Plot shown in next exercise

(*The code can be found on Github under python directory with name Ch3_2-4.py: this code computes the MAP estimator, evaluate the posterior distribution, and compares it to the laplace approximation.*)
Solution End

The Laplace approximation is a method for approximating a distribution with a Gaussian, by matching the mean and variance at the mode.[1] Let P^* be the (unnormalized) PDF of a distribution we wish to approximate. We start by taking a Taylor expansion of the log (unnormalized) PDF at the global maximizing value x^*

$$\log P^*(x) \approx \log P^*(x^*) - \frac{c}{2}(x-x^*)^2$$

where $c = -\frac{\delta^2}{\delta x^2}\log P^*(x)\Big|_{x=x^*}$.

We approximate P^* with an unnormalized Gaussian, with the same mean and variance as P^*:

$$Q^*(x) = P^*(x^*)\exp\left\{-\frac{c}{2}(x-x^*)^2\right\}$$

Exercise 3.4 *Find the mean and precision of a Gaussian that can be used in a Laplace approximation to the posterior distribution over β.*

Solution:
DATA is Normalized

mean $= \hat{\beta} \approx -0.011014$.

$$\begin{aligned} c = -\frac{\partial^2}{\partial\beta^2}\log P^*(x)\Big|_{\beta=\beta^*} &= \frac{\partial^2}{\partial\beta^2}\sum_{i=1}^{n}\log(1+e^{-y_i x^T\beta}) + \frac{1}{2\sigma^2}(\beta-\mu)^2\Big|_{\beta=\beta^*} \\ &= \sum_{i=1}^{n}\frac{y_i^2 x_i^2 e^{y_i x_i\beta}}{(1+e^{y_i x_i\beta})^2} + \frac{1}{\sigma^2} \approx 200894 \end{aligned}$$

precision ≈ 200894.

[1] More generally, the Laplace approximation is used to approximate integrands of the form $\int_A e^{Nf(x)}dx...$ but for our purposes we will always be working with PDFs.

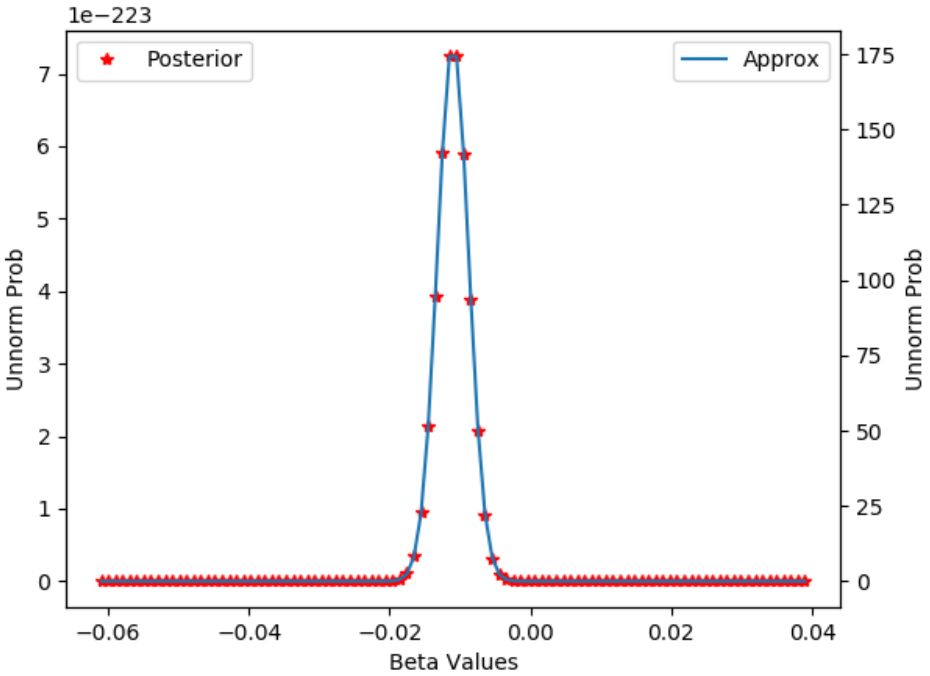

Figure 3.1: Comparison posterior and laplace approximation.

(*The code can be found on Github under python directory with name Ch3_2-4.py: this code computes the MAP estimator, evaluate the posterior distribution, and compares it to the laplace approximation.*)
Solution End

Exercise 3.5 *That's all well and good... but we probably have a non-zero intercept. We can extend the Laplace approximation to multivariate PDFs. This amounts to estimating the precision matrix of the approximating Gaussian using the negative of the Hessian – the matrix of second derivatives*

$$H_{ij} = \frac{\delta^2}{\delta x_i \delta x_j} \log P^*(x)\Big|_{x=x^*}$$

Use this to approximate the posterior distribution over β. Give the form of the approximating distribution, plus 95% marginal credible intervals for its elements.

Solution:
DATA is normalized and y in the derivative will not make a difference 0,1 values.

$$H = \begin{bmatrix} \sum_{i=1}^n \frac{y_i^2 x_{i,1}^2 e^{y_i x_i \beta}}{(1+e^{y_i x_i \beta})^2} + \frac{1}{\sigma^2} & \sum_{i=1}^n \frac{y_i^2 x_{i,1} x_{i,2} e^{y_i x_i \beta}}{(1+e^{y_i x_i \beta})^2} \\ \sum_{i=1}^n \frac{y_i^2 x_{i,1} x_{i,2} e^{y_i x_i \beta}}{(1+e^{y_i x_i \beta})^2} & \sum_{i=1}^n \frac{y_i^2 x_i^2 e^{y_i x_i \beta}}{(1+e^{y_i x_{i,2} \beta})^2} + \frac{1}{\sigma^2} \end{bmatrix} = \begin{bmatrix} 138142 & 4199214 \\ 4199214 & 155697463 \end{bmatrix}$$

Precision $= H$
mean$= [-0.079, -0.0088]$
$P(\hat{\beta}) = 8.26152586e - 223$

$$Q^*(\beta) = P(\hat{\beta}) \exp\left\{-\frac{1}{2}(\beta - \hat{\beta})^T H (\beta - \hat{\beta})\right\}$$

$\sigma_1 = 2.69e-3$
$\sigma_2 = 8.04e-5$
95% confidence intervals:
$\beta_1 = -0.079 +/- 1.91e-4$
$\beta_2 = -0.0088 +/- 5.7e-6$
(*The code can be found on Github under python directory with name Ch3_5.py*)
Solution End

Let's try the same thing with a Poisson likelihood. Here, the obvious transformation is to let $g^{-1}(\theta) = e^{\theta}$, i.e.

$$\begin{aligned} y_i|p_i \sim & \text{Poisson}(\lambda_i) \\ \lambda_i = & e^{x_i^T \beta} \end{aligned}$$

We're going to work with the dataset `tea_discipline_oss.csv`, a dataset gathered by Texas Appleseed, looking at the number of out of school suspensions (ACTIONS) accross schools in Texas. The data is censored for privacy reasons – data points with fewer than 5 actions are given the code "-99". For now, we're going to exclude these data points.

Exercise 3.6 *We're going to use a Poisson model on the counts. Ignoring the fact that the data is censored, why is this not quite the right model? Hint: there are several answers to this – the most fundamental involve considering the support of the Poisson.*

Solution:
Among the issues:

- There are no zero values in the data
- The data is over spread
- The data can be multi-model

Solution End

Exercise 3.7 *Let's assume our only covariate of interest is GRADE*[2] *and put a normal prior on β. Using a Laplace approximation and an appropriately vague prior, find 95% marginal credible intervals for the entries of β. You'll probably want to use an intercept.*

Solution:
First step is to solve the MAP problem to get the estimate for β. The likelihood is given by:

$$\text{likelihood} = \prod_{i=1}^{n} P(y_i|x_i, \beta)$$

The log likelihood is:

$$\text{log-likelihood} = \sum_{i=1}^{n} y_i(x^T\beta) - e^{x_i^T \beta}$$

[2] I have manually replaced Kindergarten and Pre-K with Grades 0 and -1, respectively.

the MAP objective:

$$\min \sum_{i=1}^{n} (-y_i(x_i^T\beta) + e^{x_i^T\beta}) + \frac{1}{2\sigma^2}(\beta - \mu)^2$$

$\hat{\beta} \approx [2.389, 0.0502]$.
The posterior distribution is:

$$p(\beta|\mathbf{x}, \mathbf{y}, \mu, \sigma^2) \propto \left\{ \prod_{i=1}^{n} \frac{(e^{x_i^T\beta})^{y_i} e^{-e^{x_i^T\beta}}}{y_i!} \right\} e^{\frac{1}{2\sigma^2}(\beta-\mu)^2}$$

The Hessian Matrix:

$$H = \begin{bmatrix} \sum_{i=1}^{n} x_{i,1}^2 e^{x_i^T\beta} + \frac{1}{\sigma^2} & \sum_{i=1}^{n} x_{i,1}x_{i,2} e^{x_i^T\beta} \\ \sum_{i=1}^{n} x_{i,1}x_{i,2} e^{x_i^T\beta} & \sum_{i=1}^{n} x_{i,2}^2 e^{x_i^T\beta} + \frac{1}{\sigma^2} \end{bmatrix} = \begin{bmatrix} 6.97e9 & 5.1e10 \\ 5.1e10 & 4.334e11 \end{bmatrix}$$

$\sigma_1 = 1.19e-5$
$\sigma_2 = 1.52e-6$
95% confidence intervals:
$\beta_1 = 2.389 +/- 1.62e-7$
$\beta_2 = 0.05 +/- 2.05e-8$

(*The code can be found on Github under python directory with name Ch3_7.py*)

Solution End

Exercise 3.8 (Optional) *Repeat the analysis using a set of variables that interest you.*

Even though we don't have conjugacy, we can still use MCMC methods – we just can't use our old friend the Gibbs sampler. Since this isn't an MCMC course, let's use STAN, a probabilistic programming language available for R, python and Matlab. I'm going to assume herein that we're using RStan, and give appropriate scripts; it should be fairly straightforward to use if you're an R novice, or if you want to use a different language, there are hints on translating to PyStan at `http://pystan.readthedocs.io/en/latest/differences_pystan_rstan.htm` and info on MatlabStan (which seems much less popular) at `http://mc-stan.org/users/interfaces/matlab-stan`.

Exercise 3.9 *Download the sample STAN script* `poisson.stan` *and corresponding R script* `run_poisson_stan.R`. *The R script should run the regression vs GRADE from earlier (feel free to change the prior parameters). Run it and see how the results differ from the Laplace approximation. Modify the script to include more variables, and present your results.*

Solution:

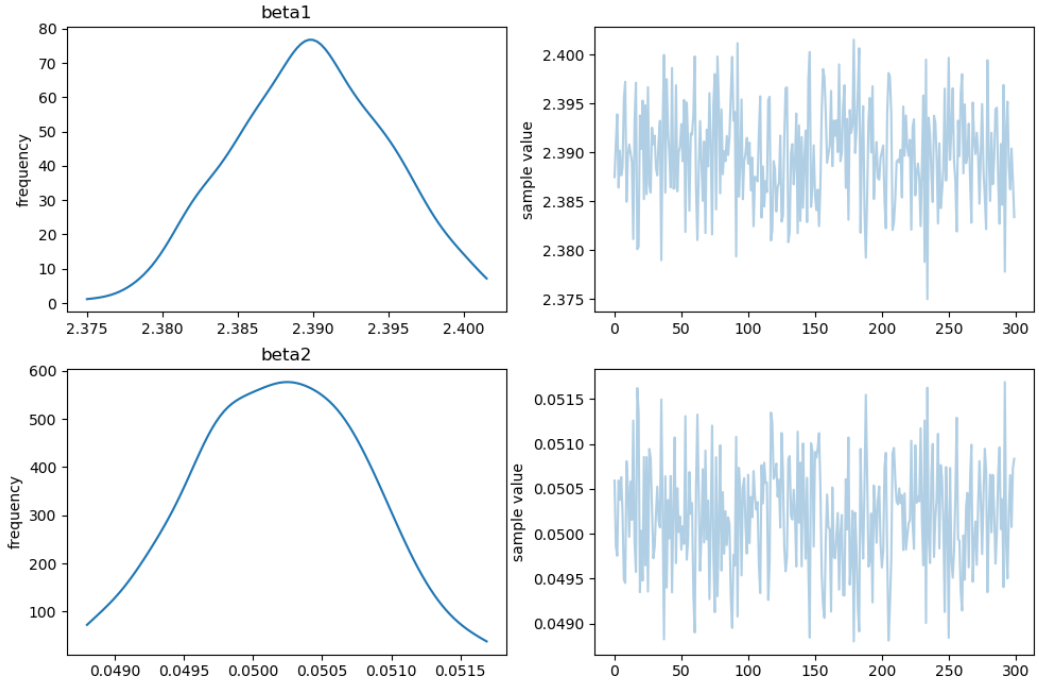

Figure 3.2: Stan run results.

(*The code can be found on Github under python directory with name Ch3_9.py, the used stan file can be found under the same directory with the name 'poisson.stan'*)
Solution End

Exercise 3.10 *Consider ways you might improve your regression (still, using the censored data) - while staying in the GLM framework. Ideas might include hierarchical error modeling (as we looked at in the last set of exercises), interaction terms... or something else! Looking at the data may give you inspiration. Implement this in STAN.*

Solution:
We can include more covariates to the model. After examining the data, one can notice that 'SEXX' can help in the model prediction. Hence, it was added as a covariate. Also, an interaction term 'SEXX*GRADE' was added. (*The code can be found on Github under python directory with name Ch3_10.py, the used stan file can be found under the same directory with the name 'poisson_s.stan'*)

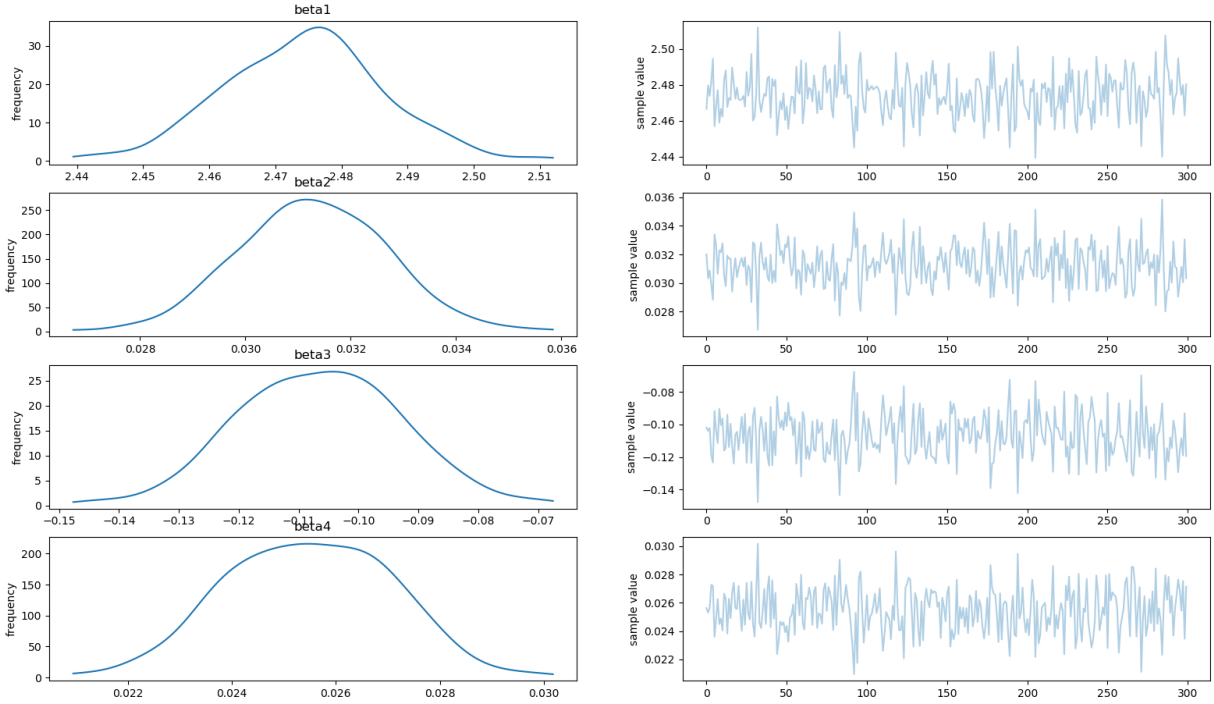

Figure 3.3: Stan run results.

Solution End

Exercise 3.11 *We are throwing away a lot of information by not using the censored data. Come up with a strategy, and write down how you would alter your model/sampler. Bonus points for actually implementing it in STAN (hint: look up the section on censored data in the STAN manual).*

Solution:
We can add to the likelihood function the likelihood that the censored data get a label between 1 and 4.

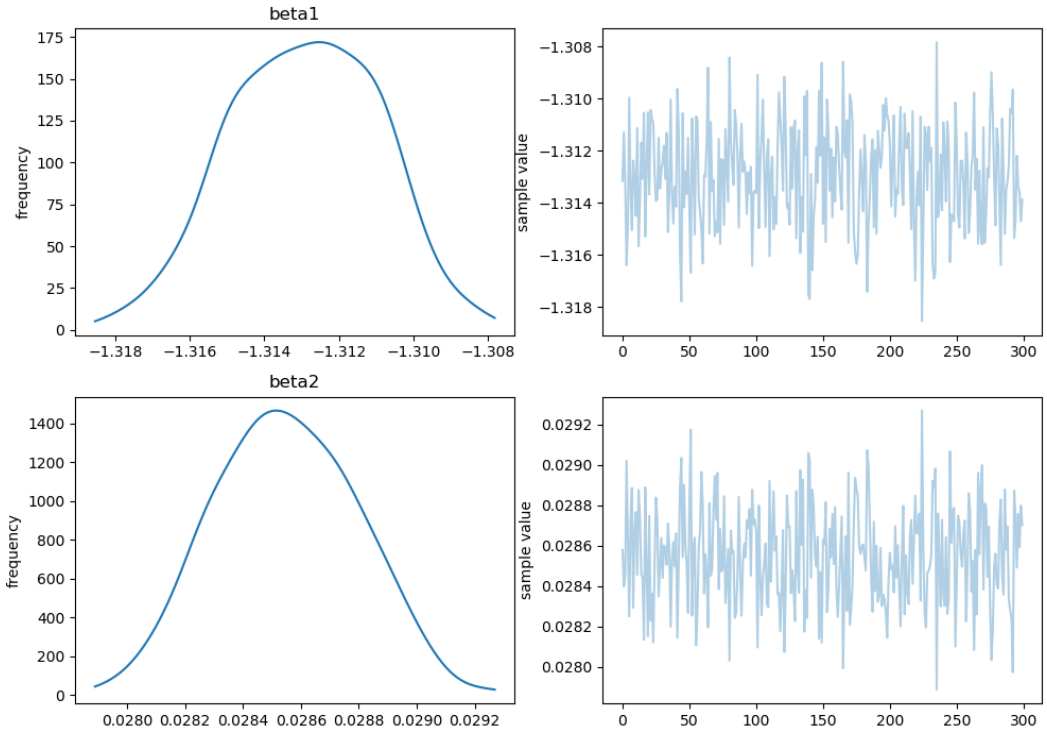

Figure 3.4: Stan run results.

The model is completely different from the one obtained with no censored data. (*The code can be found on Github under python directory with name Ch3_11.py, the used stan file can be found under the same directory with the name 'poisson_cen.stan'*)

Solution End