INFSCI 2595 Homework: 05

Assigned March 23, 2020; Due: March 31, 2020

Ying Zhang

Submission time: March 31, 2020 at 9:00AM EST

Collaborators Wenbiao Li, Tom Yang, Brinden Elton

Overview

This homework assignment serves as a review to help prepare for Test 02. It covers important mathematical concepts behind linear and generalized linear models. You will get extra practice working with the equations behind the functions you have programmed over the last several assignments.

```
library(dplyr)
library(ggplot2)
```

Problem 01

Homework 04 was all about applying linear and generalized linear modeling techniques to identify the best performing models. You created many spline basis function models with many degrees of freedom to identify the simplest possible model that did not overfit to the training data. You also fit two logistic regression models to identify the simplest model that does not overfit the binary outcome. This problem focuses on understanding the matrix sum of squares since it is so critical for understanding the uncertainty in the coefficients and the relationship between them.

A data set is loaded for you in the code chunk below. The data set consists of 2 inputs, x_1 and x_2, and a continuous response, y. A glimpse of the data set is provided in the output of the code chunk below.

```
prob_01_df <- readr::read_csv("https://raw.githubusercontent.com/jyurko/INFSCI_2595_Spring_2020/master/
```

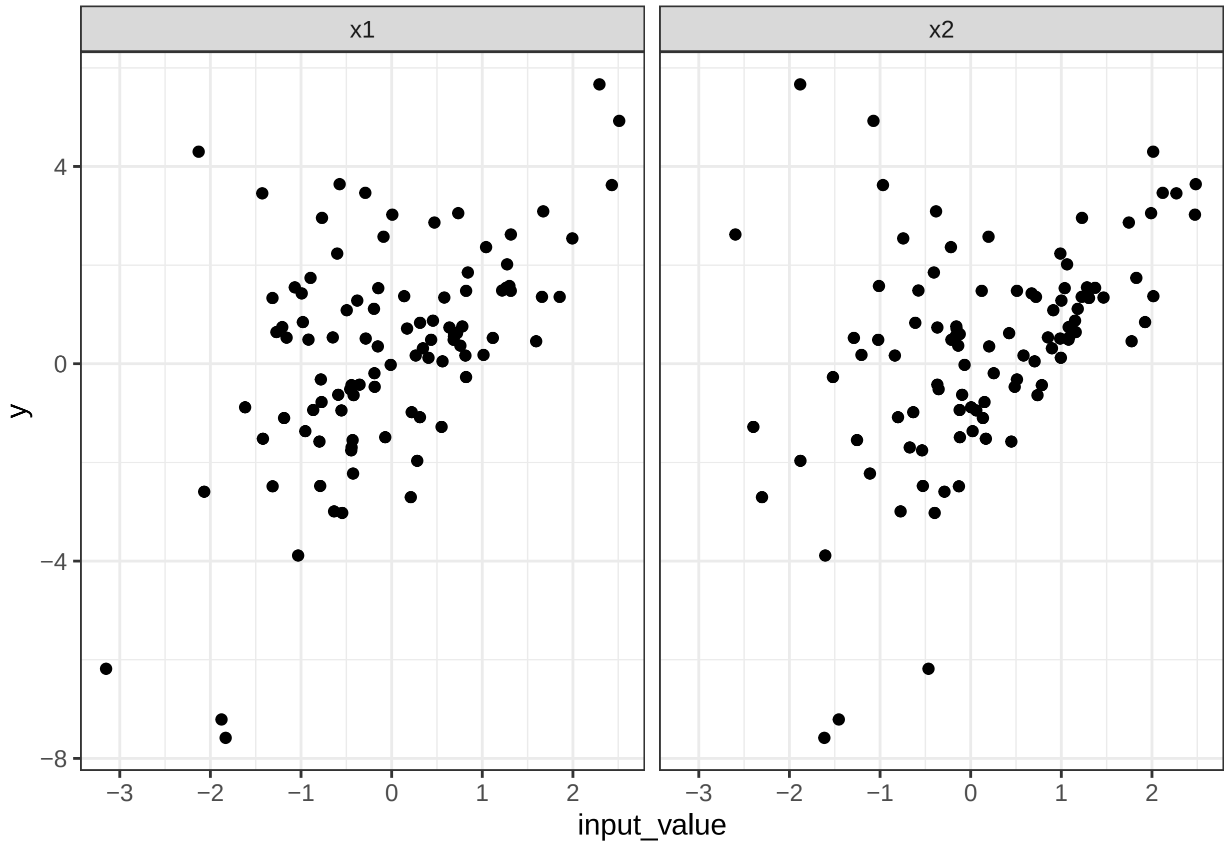
```
## Parsed with column specification:
## cols(
##   x1 = col_double(),
##   x2 = col_double(),
##   y = col_double()
## )
```

```
prob_01_df %>% glimpse()
```

```
## Observations: 100
## Variables: 3
## $ x1 <dbl> -0.14775576, 1.59426168, 0.84069129, 1.67248595, -0.64943858, 0....
## $ x2 <dbl> 1.0375823, 1.7757642, -0.4059740, -0.3821706, 0.8524313, -0.8020...
## $ y  <dbl> 1.5331526, 0.4566711, 1.8520017, 3.0913179, 0.5349807, -1.083487...
```

The code chunk below creates a scatter plot for you between the response and the two inputs. The data is first reshaped into a "long" format to allow making the separate scatter plots side-by-side with `facet_grid()`.

```r
prob_01_df %>%
  tibble::rowid_to_column() %>%
  tidyr::gather(key = "input_name",
                value = "input_value",
                -rowid, -y) %>%
  ggplot(mapping = aes(x = input_value, y = y)) +
  geom_point() +
  facet_grid(~input_name) +
  theme_bw()
```



1a)

In this problem, you will study the behavior of a linear relationship with each input plus an interaction term between the two. The mean trend expression is written below:

$$\mu_n = \beta_0 + \beta_1 x_{n,1} + \beta_2 x_{n,2} + \beta_3 x_{n,1} x_{n,2}$$

You must create the design matrix for this particular interaction model.

PROBLEM Create the design matrix associated with the model which includes the interaction term between x_1 and x_2. Assign the design matrix to the variable `Xmat`. How many columns are in the design matrix? Which column corresponds to the interaction term?

HINT: The column names from a `matrix` in `R` can be accessed with the `colnames()` function.

```r
Xmat <- model.matrix(y ~x1 + x2 + x1*x2, prob_01_df)
ncol(Xmat)
```

SOLUTION

```
## [1] 4
```

```r
colnames(Xmat)
```

```
## [1] "(Intercept)" "x1"          "x2"          "x1:x2"
```

4 columns are in the design matrix. "x1:x2" column corresponds to the interaction term. ### 1b)

You will now calculate the matrix sum of squares.

PROBLEM Calculate the matrix sum of squares and assign the result to the `SSmat` variable. What are the dimensions of `SSmat`?

```r
SSmat <- t(Xmat)%*%Xmat
SSmat
```

SOLUTION

```
##             (Intercept)         x1         x2      x1:x2
## (Intercept)  100.000000  -3.076267  16.261047 -25.811514
## x1            -3.076267 112.044042 -25.811514 -10.508878
## x2            16.261047 -25.811514 128.670387   1.603736
## x1:x2        -25.811514 -10.508878   1.603736 154.846461
```

```r
dim(SSmat)
```

```
## [1] 4 4
```

1c)

Let's take a closer look at the matrix sum of squares to get a better picture of its structure.

PROBLEM What is the value of the first row, first column $[1, 1]$ element within `SSmat`? Why does it equal that particular value?

```
SSmat[1,1]
```

SOLUTION

```
## [1] 100
```

1d)

Let's look at another element in the matrix sum of squares.

PROBLEM What is the value of the 2nd row, 3rd column in the matrix sum of squares? Demonstrate how that value is calculated

```
SSmat[2,3]
```

SOLUTION

```
## [1] -25.81151
```

It equals to sum of one hundred x1*x2 element, which is also the same as x1x2 element.

$$\begin{bmatrix} 1 & x_{1,1} & x_{2,1} & x_{1,1}x_{2,1} \\ x_{1,1} & x_{1,1}^2 & x_{1,1}x_{2,1} & x_{1,1}^2x_{2,1} \\ x_{2,1} & x_{2,1}x_{1,1} & x_{2,1}^2 & x_{2,1}^2x_{1,1} \\ x_{1,1}x_{2,1} & x_{1,1}^2x_{2,1} & x_{1,1}x_{2,1}^2 & x_{1,1}^2x_{2,1}^2 \end{bmatrix} + \begin{bmatrix} 1 & x_{1,2} & x_{2,2} & x_{1,2}x_{2,2} \\ x_{1,2} & x_{1,2}^2 & x_{1,2}x_{2,2} & x_{1,1}^2x_{2,2} \\ x_{2,2} & x_{2,2}x_{1,2} & x_{2,2}^2 & x_{2,2}^2x_{1,2} \\ x_{1,2}x_{2,2} & x_{1,2}^2x_{2,2} & x_{1,2}x_{2,2}^2 & x_{1,2}^2x_{2,2}^2 \end{bmatrix} + \dots$$

$$\sum_{n=1}^{100} x_{1,n}x_{2,n}$$

1e)

Let's now consider the entire matrix sum of squares.

PROBLEM What other elements in the matrix sum of squares are equal to the value in the 2nd row and 3rd column? Why is that the case?

```
which(SSmat == SSmat[2,3], arr.ind = TRUE)
```

SOLUTION

```
##             row col
## x1:x2         4   1
## x2            3   2
## x1            2   3
## (Intercept)   1   4
```

The elements on antidiagonal are the same, because they all represent multiplication of x1 and x2.

1f)

The matrix sum of squares controls the posterior covariance matrix of the coefficients. Let's examine the behavior of the posterior covariance matrix under two different assumptions for the noise, σ.

PROBLEM Calculate the posterior covariance matrix assuming $\sigma = 1$. Assign the result to `bcov_1`. Then calculate the posterior covariance matrix assuming $\sigma = 5$ and assign the result to `bcov_5`.

SOLUTION The posterior covariance matrix is the squared noise, σ^2, multiplied by the inverse of the matrix sum of squares. The two different cases are calculated below.

```
bcov_1 <- solve(SSmat)*1^1

bcov_5 <- solve(SSmat)*5^2
bcov_1
```

```
##               (Intercept)           x1            x2         x1:x2
## (Intercept)  0.0106892313 0.0001535942 -0.0013425789  0.0018061277
## x1           0.0001535942 0.0094187648  0.0018619646  0.0006455365
## x2          -0.0013425789 0.0018619646  0.0083172693 -0.0001835724
## x1:x2        0.0018061277 0.0006455365 -0.0001835724  0.0068047868
```

```
bcov_5
```

```
##              (Intercept)          x1           x2        x1:x2
## (Intercept)  0.267230781 0.003839854 -0.033564473  0.045153192
## x1           0.003839854 0.235469121  0.046549114  0.016138413
## x2          -0.033564473 0.046549114  0.207931734 -0.004589311
## x1:x2        0.045153192 0.016138413 -0.004589311  0.170119670
```

1g)

Problem 1f) considered two possible noise values. Does the posterior *correlation* between the coefficients change based on our assumed noise?

PROBLEM Convert the posterior covariance matrices to correlation matrices. Are the correlation matrices different?

```


cov2cor(bcov_1)
```

SOLUTION

```
##             (Intercept)         x1          x2       x1:x2
## (Intercept)  1.00000000 0.01530751 -0.14238892  0.21177172
## x1           0.01530751 1.00000000  0.21037012  0.08063370
## x2          -0.14238892 0.21037012  1.00000000 -0.02440112
## x1:x2        0.21177172 0.08063370 -0.02440112  1.00000000
```

```r
cov2cor(bcov_5)
```

```
##             (Intercept)         x1          x2       x1:x2
## (Intercept)  1.00000000 0.01530751 -0.14238892  0.21177172
## x1           0.01530751 1.00000000  0.21037012  0.08063370
## x2          -0.14238892 0.21037012  1.00000000 -0.02440112
## x1:x2        0.21177172 0.08063370 -0.02440112  1.00000000
```

No, they're the same.

Problem 02

In lecture, most of the detailed derivations we covered were for the case a linear model with known noise, σ, and an infinitely diffuse prior, $p(\boldsymbol{\beta}) \propto 1$, on the mean trend coefficients. We discussed graphically the behavior of the posterior with an informative prior and discussed the closed form analytic expression for the posterior when we use Gaussian (or MVN) distributions on the coefficients. We also discussed the relationship between the Gaussian prior with the Ridge penalty term in non-Bayesian settings.

In this problem, you will get practice working through the structure of the equations when the noise, σ, is considered unknown and must be learned along with the $\boldsymbol{\beta}$ parameters. This is the complete structure you worked with in the last assignment when you fit the various spline basis models. So you will now step into the math to see what was going on behind the scenes in the functions you programmed in the `lm_logpost()` function from Homework 04.

The complete probability model is written for you below. It uses independent Gaussian priors on the coefficients with shared (or common) prior mean μ_β and shared prior standard deviation τ_β. It also uses an Exponential distribution as the prior on the unknown noise, σ. The rate hyperparameter is now enoded as ν to avoid confusion with the regularization or penalty factor λ. So please be aware of the notation change relative to Homework 04. You will still work with the mean trend with the interaction between the two inputs.

$$y_n \mid \mu_n, \sigma \sim \text{normal}\left(y_n \mid \mu_n, \sigma\right)$$

$$\mu_n = \beta_0 + \beta_1 x_{n,1} + \beta_2 x_{n,2} + \beta_3 x_{n,1} x_{n,2}$$

$$\boldsymbol{\beta} \mid \mu_\beta, \tau_\beta \sim \prod_{d=0}^{D} \left(\text{normal}\left(\beta_d \mid \mu_\beta, \tau_\beta\right)\right)$$

$$\sigma \mid \nu \sim \text{Exp}\left(\sigma \mid \nu\right)$$

I like to use this format to describe the models because it high lights each of the key piecs, the likelihood, the deterministic relationship, and the priors. However, if we focus on the log-posterior directly, it is more convenient to write things in more compact notation. The un-normalized log-posterior written as the summation of the log-likelihood and the log-prior is given to you below.

$$\log\left[p\left(\boldsymbol{\beta}, \sigma \mid \mathbf{X}, \mathbf{y}\right)\right] \propto \log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma\right)\right] + \log\left[p\left(\boldsymbol{\beta} \mid \mu_\beta, \tau_\beta\right)\right] + \log\left[p\left(\sigma \mid \nu\right)\right]$$

In this problem, you work through the manipulating each of the components of the un-normalized log-posterior to get a feel for the math behind the functions when the noise σ is unknown.

NOTE: Throughout this problem you are allowed to use separate equation blocks for each equation you type. It is easier than placing all equations in a single equation block when going through the PDF rendering process.

2a)

You will start with the log-likelihood.

PROBLEM Write out the log-likelihood as a function of the unknown noise, σ, and the unknown coefficients, $\boldsymbol{\beta}$. Make use of matrix notation to write the log-likelihood in terms of the design matrix X. You can drop all constants that do not directly involve σ or $\boldsymbol{\beta}$, thus you can write out the log-likelihood up to a normalizing constant.

SOLUTION Your derivation here.

$$\log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma\right)\right] = \sum_{n=1}^{N}(-\frac{1}{2}log(\sigma^2) - \frac{1}{2}log[2\pi] - \frac{1}{2\sigma^2}(y_n - \mu_n)^2)$$

$$\log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma\right)\right] \propto -Nlog\sigma - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mu_n)^2$$

$$\log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma\right)\right] \propto -Nlog\sigma - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mathbf{x}_{n,:}\boldsymbol{\beta})^2$$

$$\log\left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma\right)\right] \propto -Nlog\sigma - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

2b)

Next, let's consider the log-prior on the mean trend coefficients.

PROBLEM Write out the log-prior on the $\boldsymbol{\beta}$ parameters as a function of $\boldsymbol{\beta}$ and σ. You can write the log-prior up to a normalizing constant. Does the log-prior on $\boldsymbol{\beta}$ depend on σ?

SOLUTION Your derivation here.

$$\log\left[p\left(\boldsymbol{\beta} \mid \mu_\beta, \tau_\beta\right)\right] = \sum_{d=0}^{D}(-\frac{1}{2}\log[\tau_\beta^2] - \frac{1}{2}\log[2\pi] - \frac{1}{2\tau_\beta^2}(\beta_d - \mu_\beta)^2)$$

$$\log\left[p\left(\boldsymbol{\beta} \mid \mu_\beta, \tau_\beta\right)\right] \propto -\frac{1}{2\tau_\beta^2}\sum_{d=0}^{D}(\beta_d - \mu_\beta)^2$$

The log-prior on β doesn't depend on σ. ### 2c)

And now let's consider the log-prior on the unknown noise σ.

PROBLEM Write out the log-prior on the noise σ as a function of $\boldsymbol{\beta}$ and σ. You can write the log-prior up to a normalizing constant, and thus drop constant terms. Does the log-prior on σ depend on $\boldsymbol{\beta}$?

HINT: You worked with the pdf to the Exponential distribution in Homework 02.

SOLUTION Your derivation here.

$$\log\left[p\left(\sigma \mid \nu\right)\right] = \log\nu - \nu\sigma$$

$$\log\left[p\left(\sigma \mid \nu\right)\right] \propto -\nu\sigma$$

The log-prior on σ doesn't depend on β. ### 2d)

You have written out each of the components of the log-posterior. It's now time to combine them together.

PROBLEM Combine the log-likelihood and the log-priors together to write out the log-posterior up to a normalizing constant.

SOLUTION

$$\log\left[p\left(\boldsymbol{\beta},\sigma \mid \mathbf{X},\mathbf{y}\right)\right] \propto \log\left[p\left(\mathbf{y} \mid \mathbf{X},\boldsymbol{\beta},\sigma\right)\right] + \log\left[p\left(\boldsymbol{\beta} \mid \mu_\beta,\tau_\beta\right)\right] + \log\left[p\left(\sigma \mid \nu\right)\right]$$

$$\log\left[p\left(\boldsymbol{\beta},\sigma \mid \mathbf{X},\mathbf{y}\right)\right] \propto -N log\sigma - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) - \frac{1}{2\tau_\beta^2}\sum_{d=0}^{D}(\beta_d-\mu_\beta)^2 - \nu\sigma$$

2e)

Throughout the semester we have assumed that the prior mean on the $\boldsymbol{\beta}$ parameters is zero, $\mu_\beta = 0$. Write out the log-posterior function assuming that is the case.

PROBLEM Write out the log-posterior function assuming $\mu_\beta = 0$. Does the prior still have effect when we make that assumption? What controls the "strength" of the prior?

SOLUTION

$$\log\left[p\left(\boldsymbol{\beta},\sigma \mid \mathbf{X},\mathbf{y}\right)\right] \propto -N log\sigma - \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) - \frac{1}{2\tau_\beta^2}\sum_{d=0}^{D}(\beta_d)^2 - \nu\sigma$$

Yes, it does. σ controls the "strength" of the prior.

Problem 03

You will now focus more closely on the unknown σ parameter within the log-posterior. In this problem, you will build off of your solutions in Problem 02 and make use of the probability change-of-variables. In this way you will see what goes on behind the scenes when the Laplace Approximation approximates the posterior in the unbounded space with the φ parameter.

3a)

Before performing the change-of-variables, let's rewrite the log-posterior to depend on the sum of squared errors (SSE).

PROBLEM Rewrite the un-normalized log-posterior in terms of the SSE.

SOLUTION

$$\log[p(\beta|\mathbf{X},\boldsymbol{y},\sigma)] \propto -\frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}(SSE + \lambda\sum_{d=0}^{D}(\beta_d^2))$$

3b)

Up to this point in the semester, you have fit many linear models with unknown noise terms. The Laplace Approximation function `my_laplace()` took care of the optimization and Hessian matrix evaluations for you. You had to provide the log-posterior function. Because the Laplace Approximation approximates the posterior as a MVN, you have been using a change-of-variables transformation to transform the lower bounded σ parameter to an unbounded parameter, φ. The Laplace Approximation was then applied to the joint posterior in the unbounded space between $\boldsymbol{\beta}$ and φ.

The transformation is applied via a *link* function, $g(\cdot)$. The unbounded transformed noise φ is then:

$$\varphi = g(\sigma)$$

The original noise parameter is then equal to the *inverse link* function applied to the transformed variable:

$$\sigma = g^{-1}(\varphi)$$

You will work through how the transformation is applied and modifies the log-posterior function. Write out the log-posterior function for the unbounded variables $\boldsymbol{\beta}$ and φ. You do not need to perform all of the substitutions just yet. What must be added to the log-posterior based on σ in order to write out the log-posterior for φ?

PROBLEM Complete the expression below. How should you write out the log-posterior relative to σ in terms of φ and what must be added to that log-posterior? Work in general terms for now with a general link function $g(\cdot)$ and general inverse link function $g^{-1}(\cdot)$

SOLUTION Complete the expression below.

$$\log\left[p\left(\boldsymbol{\beta}, \varphi \mid \mathbf{X}, \mathbf{y}\right)\right] = \log\left[p\left(\boldsymbol{\beta}, g^{-1}(\varphi) \mid \mathbf{X}, \mathbf{y}\right)\right] + \log[\frac{d}{d\varphi}g^{-1}(\varphi)]$$

3c)

Let's now make use of the transformation that you used in the previous homework assignments. The unbounded variable φ will be the log-transformation of the noise σ:

$$\varphi = \log[\sigma]$$

PROBLEM Derive the log-posterior function relative to $\boldsymbol{\beta}$ and φ completely in terms of $\boldsymbol{\beta}$ and φ.

SOLUTION

$$\log\left[p\left(\boldsymbol{\beta}, \varphi \mid \mathbf{X}, \mathbf{y}\right)\right] = -\frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}(SSE + \lambda\sum_{d=0}^{D}(\beta_d^2)) + \varphi$$

3d)

The first step in the Laplace Approximation is to identify the posterior mode. The gradient vector is calculated and an optimization scheme iterates until the mode is found. You do NOT need to calculate the complete gradient vector in this problem. You will focus on the partial first derivative of the un-normalized log-posterior with respect to φ.

PROBLEM Derive the partial first derivative of the un-normalized log-posterior with respect to φ.

HINT: You can leave the result in terms of the SSE.

SOLUTION

$$\frac{df}{d\varphi} = -N + SSE \exp^{-2}(\varphi) + 1$$

3e)

Once the posterior mode has been identified, the second step of the Laplace Approximation is calculate the Hessian matrix - the matrix of second derivatives. You do NOT need to calculate the complete Hessian matrix. In this problem you will focus on the partial second derivative of the un-normalized log-posterior with respect to φ.

PROBLEM Derive the partial second derivative of the un-normalized log-posterior with respect to φ.

SOLUTION

$$\frac{df}{d\varphi^2} = -2SSE \exp^{-2}(\varphi)$$

3f)

Let's now assume that we are using a very diffuse prior on the noise, and thus $\nu \rightarrow 0$, as well as a very diffuse prior on the $\boldsymbol{\beta}$ parameters, $\tau_\beta \rightarrow \infty$. Under these conditions the sum of squared errors associated with the OLS estimates, $\boldsymbol{\beta}_{OLS}$, will be denoted as SSE_{OLS}.

PROBLEM Based on your derivation in Problem 3d), derive the maximum likelihood estimate (MLE) on the φ parameter given the OLS estimate to the SSE.

SOLUTION

$$\frac{df}{d\varphi} = -N + SSE \exp^{-2}(\varphi) + 1 = 0$$

$$\varphi = 2 \log[\frac{N-1}{SSE_{OLS}}]$$

Problem 04

Now that you have explored the math behind the log-posterior in great detail, it's time to fit a Bayesian linear model. You will continue to use independent Gaussian priors on the $\boldsymbol{\beta}$ parameters with common prior mean $\mu_\beta = 0$ and common prior standard deviation τ_β. Specify the prior on the unknown noise, σ, as an Exponential distribution with rate parameter $\nu = 1.5$. You will also continue to use the log-transformation on the noise parameter. Thus the log-posterior function will be defined in terms of the unknown $\boldsymbol{\beta}$ parameters and the unknown φ parameter.

You will continue to work with the linear relationship which includes the interaction term:

$$\mu_n = \beta_0 + \beta_1 x_{n,1} + \beta_2 x_{n,2} + \beta_3 x_{n,1} x_{n,2}$$

4a)

You will define the `lm_logpost()` function, just as you did in Homework 04. Thus, the first step is to create the list of required information which includes the responses, design matrix, and the prior hyperparameters. Specify the prior standard deviation to be 5.

However, compared to Homework 04, you must standardize the response variable y before fitting the model.

PROBLEM Define the list of required information `info_04` in the code chunk below. Calculate the standardized response, `y_stan` and set that equal to the `yobs` variable in the `info_04` list.

SOLUTION Complete the code chunk below by filling in the required information.

```r
### calculate the standardized response
y_stan <- scale(prob_01_df$y)

info_04 <- list(
  yobs = y_stan,
  design_matrix = Xmat,
  mu_beta = 0,
  tau_beta = 5,
  sigma_rate = 1.5
)
```

4b)

Define the log-posterior function, `lm_logpost()`, in the code chunk below.

PROBLEM Complete the code chunk below. The comments specify what needs to be completed.

Once completed test out the function by evaluating the log-posterior at two different sets of parameter values. Try out values of -2 and 2 for all parameters.

HINT: This should look familiar...

HINT: If your function is completed successfully, you should get a value of -50214.42 for the -2 guess, and a value of -325.3106 for the guess of all 2's.

```r
lm_logpost <- function(unknowns, my_info)
{
  # specify the number of unknown beta parameters
  length_beta <- ncol(my_info$design_matrix)

  # extract the beta parameters from the `unknowns` vector
  beta_v <- unknowns[1:length_beta]

  # extract the unbounded noise parameter, varphi
  lik_varphi <- unknowns[length_beta + 1]

  # back-transform from varphi to sigma
  lik_sigma <- exp(lik_varphi)

  # extract design matrix
  X <- my_info$design_matrix

  # calculate the linear predictor
  mu <- as.vector(X%*%as.matrix(beta_v))

  # evaluate the log-likelihood
  log_lik <- sum(dnorm(x = my_info$yobs,
                       mean = mu,
                       sd = lik_sigma,
                       log = TRUE))

  # evaluate the log-prior
  log_prior_beta <- sum(dnorm(x = beta_v,
                              mean = my_info$mu_beta,
                              sd = my_info$tau_beta,
                              log = TRUE))

  log_prior_sigma <- dexp(x = lik_sigma,
                          rate = my_info$sigma_rate,
                          log = TRUE)

  # add the mean trend prior and noise prior together
  log_prior <- log_prior_beta + log_prior_sigma

  # account for the transformation
  log_derive_adjust <- lik_varphi

  # sum together
  log_lik + log_prior + log_derive_adjust

}
```

Test out the function with a guess of -2 for all parameters below.

```r
lm_logpost(rep(-2, ncol(Xmat)+1), info_04)
```

```
## [1] -50214.42
```

Test out the function with a guess of 2 for all parameters below.

```r
lm_logpost(rep(2, ncol(Xmat)+1), info_04)
```

```
## [1] -325.3106
```

4c)

The `my_laplace()` and the `generate_lm_post_samples()` functions are provided to you in the code chunk below. You do not need to modify these code chunks at all in this assignment.

```r
### the my_laplace() function is the same you have used in the
### last several assignments
my_laplace <- function(start_guess, logpost_func, ...)
{
  # code adapted from the `LearnBayes`` function `laplace()`
  fit <- optim(start_guess,
               logpost_func,
               gr = NULL,
               ...,
               method = "BFGS",
               hessian = TRUE,
               control = list(fnscale = -1, maxit = 1001))

  mode <- fit$par
  h <- -solve(fit$hessian)
  p <- length(mode)
  int <- p/2 * log(2 * pi) + 0.5 * log(det(h)) + logpost_func(mode, ...)

  list(mode = mode,
       var_matrix = h,
       log_evidence = int,
       converge = ifelse(fit$convergence == 0,
                         "YES",
                         "NO"),
       iter_counts = fit$counts[1])
}

### pay attention to the arguments to this function, they are the same
### as those used in the previous assignment
generate_lm_post_samples <- function(mvn_result, length_beta, num_samples)
{
  MASS::mvrnorm(n = num_samples,
                mu = mvn_result$mode,
                Sigma = mvn_result$var_matrix) %>%
    as.data.frame() %>% tbl_df() %>%
    purrr::set_names(c(sprintf("beta_%02d", (1:length_beta) - 1), "varphi")) %>%
    mutate(sigma = exp(varphi))
}
```

You will now perform the Laplace Approximation to fit the Bayesian linear model and generate posterior samples of the parameters.

PROBLEM Fit the Bayesian linear model with the Laplace Approximation, using a starting guess of all zeros. Generate 2500 posterior samples. What is the posterior median on σ? What is the posterior 25th and 75th quantiles on σ? Also calculate the 25th, median, and 75th posterior quantiles on the interaction parameter, β_3.

SOLUTION Fit the model and generate the posterior samples in the code chunk below.

```
laplace_result_a <- my_laplace(rep(0,ncol(Xmat)+1),lm_logpost,info_04)


post_samples_a <- generate_lm_post_samples(laplace_result_a, ncol(Xmat), 2500 )

quantile(post_samples_a$sigma, prob = c(0.25, 0.5, 0.75))
```

```
##       25%       50%       75% 
## 0.3211181 0.3366926 0.3532695
```

```
quantile(post_samples_a$beta_03, prob = c(0.25, 0.5, 0.75))
```

```
##        25%        50%        75% 
## -0.4615369 -0.4431503 -0.4243823
```

4d)

Let's now see what happens if we try out several different prior standard deviations on the coefficients. Refit the model but this time with $\tau_\beta = 25$, and then explore the results.

PROBLEM You must create a new list of required information, fit the model with the Laplace Approximation, and generate 2500 posterior samples. Calculate the 25th, median, and 75th quantiles on the interaction parameter, β_3, and the noise, σ. Are the quantiles on σ different than the case with τ_β?

SOLUTION Make the required list of information, fit the model, and generate the samples.

```
info_04_weak <- list(
  yobs = y_stan,
  design_matrix = Xmat,
  mu_beta = 0,
  tau_beta = 25,
  sigma_rate = 1.5
)

laplace_result_weak <- my_laplace(rep(0,ncol(Xmat)+1),lm_logpost,info_04_weak)

post_samples_weak <- generate_lm_post_samples(laplace_result_weak, ncol(Xmat), 2500 )

quantile(post_samples_weak$sigma, prob = c(0.25, 0.5, 0.75))
```

```
##       25%       50%       75% 
## 0.3206175 0.3356503 0.3526428
```

```r
quantile(post_samples_weak$beta_03, prob = c(0.25, 0.5, 0.75))
```

```
##        25%        50%        75% 
## -0.4632954 -0.4439787 -0.4247524
```

Yes, they're different.

4e)

Now try a strong prior with a prior standard deviation of 1/25. Refit the model and regenerate the posterior samples. Do the results change now?

PROBLEM **You must create a new list of required information, fit the model with the Laplace Approximation, and generate 2500 posterior samples. Calculate the 25th, median, and 75th quantiles on the interaction parameter, β_3, and the noise, σ.**

How do the results compare with the previous two prior specifications?

SOLUTION Respecify the model and fit the model in the code chunk below.

```r
info_04_strong <- list(
  yobs = y_stan,
  design_matrix = Xmat,
  mu_beta = 0,
  tau_beta = 1/25,
  sigma_rate = 1.5
)

laplace_result_strong <- my_laplace(rep(0,ncol(Xmat)+1),lm_logpost,info_04_strong)

post_samples_strong <- generate_lm_post_samples(laplace_result_strong, ncol(Xmat), 2500 )

quantile(post_samples_strong$sigma, prob = c(0.25, 0.5, 0.75))
```

```
##       25%       50%       75% 
## 0.7615360 0.8088336 0.8544920
```

```r
quantile(post_samples_strong$beta_03, prob = c(0.25, 0.5, 0.75))
```

```
##         25%         50%         75% 
## -0.14617875 -0.12012599 -0.09469276
```

The 25th, median, and 75th quantiles on the interaction parameter, β_3, and the noise, σ shift to right(become larger).

Problem 05

In addition to stepping through the derivation of the linear model, we also went through the key mathematical concepts behind logistic regression. In lecture, we derived the gradient and Hessian for a logistic regression model assuming an infinitely diffuse prior. We discussed the classic approach to fitting the logistic regression model with the Iteratively Reweighted Least Squares (IRLS) algorithm. You will go through one iteration of IRLS to see how the concepts from the linear model extend to a binary classification setting.

The code chunk below reads in a new data set, consisting of four continuous inputs, x_1 through x_4, and a binary response, y. The binary response is encoded as 0 for the non-event and 1 for event. A glimpse of the data is provided for you below.

```
prob_05_df <- readr::read_csv("https://raw.githubusercontent.com/jyurko/INFSCI_2595_Spring_2020/master/
```

```
## Parsed with column specification:
## cols(
##   x1 = col_double(),
##   x2 = col_double(),
##   x3 = col_double(),
##   x4 = col_double(),
##   y = col_double()
## )
```

```
prob_05_df %>% glimpse()
```

```
## Observations: 200
## Variables: 5
## $ x1 <dbl> -0.1477558, 0.8406913, -0.6494386, 0.5806242, 0.2112185, 0.55058...
## $ x2 <dbl> 1.0375823, -0.4059740, 0.8524313, 1.4656521, -2.3025271, -2.3980...
## $ x3 <dbl> 1.59426168, 1.67248595, 0.31214381, -0.78855150, -1.83168391, -0...
## $ x4 <dbl> 1.77576422, -0.38217064, -0.80204467, -0.52755470, -1.61478088, ...
## $ y  <dbl> 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1...
```

5a)

You will work with a linear additive relationship between the linear predictor, η, and the inputs. The linear predictor relationship is written out for you below.

$$\eta_n = \beta_0 + \beta_1 x_{n,1} + \beta_2 x_{n,2} + \beta_3 x_{n,3} + \beta_4 x_{n,4}$$

Create the design matrix associated with this model.

PROBLEM Create the design matrix for the linear additive relationship. Assign the result to the `X05` variable. How many columns are in the design matrix?

```
X05 <- model.matrix(y ~x1 + x2 + x3+ x4 ,prob_05_df)
ncol(X05)
```

SOLUTION

```
## [1] 5
```

5 columns are in the design matrix.

5b)

The IRLS algorithm is iterative and requires us to specify an initial guess for the coefficients. You will define an initial guess of 1.5 for all coefficients. Calculate the linear predictor based on the assumed coefficient values and the design matrix created in Problem 5a).

PROBLEM Define the initial guess parameter vector `binit` below to a vector 1.5 for all parameters. Calculate the linear predictor and assign the result to `eta_init`. You must use matrix math to calculate the linear predictor.

```r
binit <- rep(1.5, ncol(X05))

eta_init <- X05 %*% as.matrix(binit)
```

SOLUTION

5c)

The next step is to calculate the event probability based on the current linear predictor guess.

PROBLEM Calculate the event probability and assign the result to the variable `mu_init`.

```r
mu_init <- boot::inv.logit(eta_init)
```

SOLUTION

5d)

With the even probability calculated with for observation you can now calculate the weighting matrix $\mathbf{S}$.

PROBLEM Calculate the weighting matrix `Sweight` based on the current guess for the event probability. What is the dimensionality of the `Sweight` matrix?

```r
### feel free to use as many steps as you feel are appropriate
### to calculate the Sweight matrix
Sweight <- diag(mu_init[,])%*%diag((1-mu_init)[,])
dim(Sweight)
```

SOLUTION

```
## [1] 200 200
```

5e)

In lecture, we discussed how the IRLS algorithm updates the guess for the coefficients by calculating the "working response", **z**. Write out the expression for the working response and then calculate the working response based on the current guess for the coefficients.

PROBLEM Write out the expression for the working response and calculate it based on the current guess for the parameters. Assing the result to the variable `z_init`

SOLUTION Use an equation block to write the expression for the working response.

```r
### calculate the working response
z_init <- eta_init + solve(Sweight)%*%(prob_05_df$y-mu_init)
```

5f)

Before calculating the updated coefficient values, let's first consider the weighted sum of squares matrix.

PROBLEM Write out the expression for the weighted sum of squares matrix. Calculate it based on the current coefficient guess and assign the result to the variable `wSSmat`. How many rows and columns does `wSSmat` have?

SOLUTION Write the expression for the weighted sum of squares matrix in an equation block.

```r
wSSmat <- t(X05) %*% Sweight %*% X05
dim(wSSmat)
```

```
## [1] 5 5
```

5g)

It's now time to calculate the updated coefficient values, $\boldsymbol{\beta}_{k+1}$.

PROBLEM Write out the formula for the new coefficients based on the current coefficient guess. Calculate the new coefficients and assign the result to `b_new`. Display the updated coefficients to the screen.

SOLUTION Write out the formula for the updated coefficients in an equation block.

```
### calculate the udpated coefficients
b_new <- binit-solve(-wSSmat) %*% (t(X05)) %*% (prob_05_df$y-mu_init)
```

Print out the updated coefficients.

```
b_new
```

```
##                  [,1]
## (Intercept) -6.480098
## x1          -5.231033
## x2          -9.157568
## x3          -6.468299
## x4          -6.894211
```