Lab2

Andreas Stasinakis(andst745) & Mim Kemal Tekin(mimte666)

May 3, 2019

Contents

Question 1: Linear and polynomial regression

The dataset TempLinkoping.txt contains daily temperatures (in Celcius degrees) at Malmsl?tt, Link?ping over the course of the year 2016 (366 days since 2016 was a leap year). The response variable is temp and the covariate is

$$time = \frac{\text{the number of days since beginning of year}}{366}.$$

The task is to perform a Bayesian analysis of a quadratic regression

$$temp = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \epsilon \overset{iid}{\sim} N(0, \sigma^2).$$

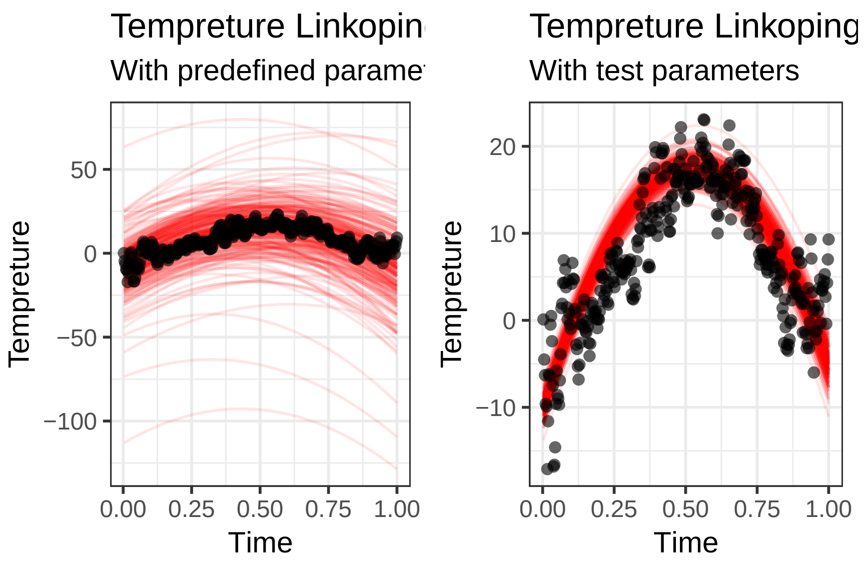
a) Joint conjugate prior for linear regression. Add lines in a plot

Determining the prior distribution of the model parameters. Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters μ_0, Ω_0, ν_0 and σ_0^2 to sensible values. Start with $\mu_0 = (-10, 100, -100)^T$, $\Omega_0 = 0.01 \cdot I_3$, $\nu_0 = 4$ and σ_0^2 =1. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves, one for each draw from the prior. Do the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves do agree with your prior beliefs about the regression curve. [Hint: the R package mvtnorm will be handy. And use your Inv-X^2 simulator from Lab 1.]

```
library(ggplot2)
library(gridExtra)
data_temp = read.table("datasets/TempLinkoping.txt", sep = "\t", header = T)

base_plot =
  ggplot(data_temp, aes(x=data_temp$time, y=data_temp$temp)) +
    geom_point() +
    labs(title = "Tempreture Linkoping",
         x = "Time", y="Tempreture") +
    theme_bw()
```

```
library(mvtnorm)

r_inv_chisq = function(n=1, df, tau_sq){
  return(df * tau_sq / rchisq(n, df))
}

prior_coef = function(mu, omega, var, n=1){
  return(rmvnorm(n, mu, var*solve(omega)))
}

prior_var = function(df, var, n=1){
  return( r_inv_chisq(n, df, var) )
}

nonlinear_model = function(x, coefs, var){
  err = rnorm(1,0, var)
  y = integer(length(x))
  for(i in 1:length(coefs)){
    y = y + x^(i-1)*coefs[i]
  }
  return(y+err)
}

# # initial prior parameters
mu_0 = c(-10, 100, -100)
omega_0 = 0.01 * diag(3)
v_0 = 4
var_0 = 1
nDraw = 250

# test prior parameters
test_mu_0 = c(-10, 105, -100)
test_omega_0 = 0.1 * diag(3)
test_v_0 = 20
test_var_0 = 0.1
test_nDraw = 200


# create dataframe to store different models
models = data.frame(x=data_temp$time,
                    y=data_temp$temp)
test_models = data.frame(x=data_temp$time,
                    y=data_temp$temp)

# CREATE nDraw model for predefined (by instructions) parameters
set.seed(12345)
for(draw in 1:nDraw){
  # draw one var with predefined parameters of priors
  drawn_var = prior_var(v_0, var_0)
  # draw the model parameters by using its prior
  coefs = prior_coef(mu_0, omega_0, drawn_var)
```

```r
  yy = nonlinear_model(models$x,
                       coefs,
                       drawn_var)
  models[[paste("y",draw,sep="_")]] = yy
}

# CREATE nDraw model for parameters which we set by hand we select them
# visually from the plot of the models with predefined parameters.
set.seed(12345)
for(draw in 1:nDraw){
  # draw one var with predefined parameters of priors
  drawn_var = prior_var(test_v_0, test_var_0)
  # draw the model parameters by using its prior
  coefs = prior_coef(test_mu_0, test_omega_0, drawn_var)
  yy = nonlinear_model(models$x,
                       coefs,
                       drawn_var)
  test_models[[paste("y",draw,sep="_")]] = yy
}

# PLOT for predefined parameters
plot_predefined = ggplot(models , aes(x=models$x)) +
  labs(title = "Tempreture Linkoping",
       subtitle = "With predefined parameters",
       x = "Time", y="Tempreture") +
  theme_bw()

for(name in names(models)[-c(1,2)]){
  plot_predefined = plot_predefined +
    geom_line(aes_string(y = name), color="red", alpha=0.1)
}

# PLOT for test parameters
plot_test = ggplot(test_models , aes(x=test_models$x)) +
  labs(title = "Tempreture Linkoping",
       subtitle = "With test parameters",
       x = "Time", y="Tempreture") +
  theme_bw()

for(name in names(models)[-c(1,2)]){
  plot_test = plot_test +
    geom_line(aes_string(y = name), color="red", alpha=0.1)
}

plot_predefined = plot_predefined +
  geom_point(aes(y = y), alpha=0.6)

plot_test = plot_test +
  geom_point(aes(y = y), alpha=0.6)

grid.arrange(grobs = list(plot_predefined, plot_test), ncol=2)
```



Using only the priors in order to draw samples from the response variable temp given the initial parameters $\mu_0 = (-10, 100, -100), \Omega_0 = 0.01, \nu_0 = 4, \sigma^2 = 1$, the output does not really make sense. More specific, many of the draws have always really low negatives temperatures for the whole year, which is not reasonable even if we are referring to a Swedish city. Changing some of the parameres, gives us more resonable results which can be seen in the second plot. Of course we should find a good fit without using the data proviced. Therefore, we used information from wikepedia in order to be helped for some reasonable temperatures. In order to choose the hyperparameters, we test a lot of different combinations. The parameter which did the bigger difference was the variance, so we focus there. As we can see from the first plot, the variance of the temperature values is really high, so we tried to reduce it. Therefore, the final hyperparameters are $\mu_0 = (-10, 105, -100), \Omega_0 = 0.1I, \nu_0 = 20, \sigma^2 = 0.1$

b) Simulate from joint posterior, plot the marginal, posterior credible and predictions intervals.

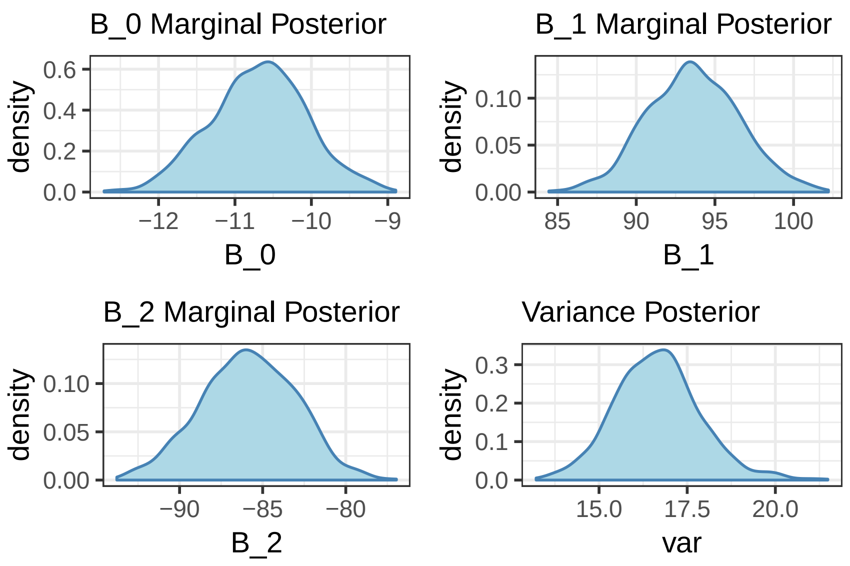
Write a program that simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2$ and σ^2. Plot the marginal posteriors for each parameter as a histogram. Also produce another figure with a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function f(time) $= \beta_0 + \beta_1 \cdot$ time $+ \beta_2 \cdot$ time^2, computed for every value of time. Also overlay curves for the lower 2.5% and upper 97.5% posterior credible interval for f(time). That is, compute the 95% equal tail posterior probability intervals for every value of time and then connect the lower and upper limits of the interval by curves. Does the interval bands contain most of the data points? Should they?

```
r_post_coef = function(n=1, mu_n, var, omega_n){
  return(rmvnorm(n, mu_n, var*solve(omega_n)))
}


r_post_var = function(n=1, v_n, var_n){
  return(as.vector(r_inv_chisq(n, v_n, var_n)))
}

######
# joint_posterior_coef  = function(x, y, mu_0, omega_0,
#                                  var_0, v_0, n=1){
#   # if mu_0 is not a matrix convert it a matrix. it have to have (3x1) dimensions
#   if(!is.matrix(mu_0)){
```

```
#     mu_0 = matrix(mu_0)
#   }
#   # calculate matrix multiplication of x data and inverse of it
#   # xx is 3x3 matrix
#   xx = t(x) %*% x
#   inv_xx = solve(xx)
#   # calculate b_hat which will be used at mu_n calculation
#   # this b_hat is estimation of b coefficients in the model
#   b_hat = inv_xx %*% t(x) %*% y
#   # calculate mu_n (3x1)
#   mu_n = solve(xx + omega_0) %*% (xx%*%b_hat + omega_0 %*% mu_0)
#   # calculate omega_n (3x3)
#   omega_n = xx + omega_0
#
#   # to draw from var posterior we need v_n, var_n
#   # calculate v_n first
#   v_n = v_0 + length(y)
#   # calculate var_n
#   # first we calculate v_n*var_n as temporary
#   temp = v_0*var_0 + (t(y) %*% y +
#                          t(mu_0)%*%omega_0%*%mu_0 -
#                          t(mu_n)%*%omega_n%*%mu_n)
#   var_n = temp / v_n                      # get var_n from the temp
#   var = c(r_post_var(1, v_n, var_n))       # sample var from posterior
#   print(var)
#
#   # draw sample from the posterior distribution and return it
#   return(r_post_coef(n, t(mu_n), var, omega_n))
# }
#
# joint_posterior_var = function(x, y, mu_0, omega_0, v_0, var_0, n=1){
#   # if mu_0 is not a matrix convert it a matrix. it have to have (3x1) dimensions
#   if(!is.matrix(mu_0)){
#     mu_0 = matrix(mu_0)
#   }
#   # calculate matrix multiplication of x data and inverse of it
#   # xx is 3x3 matrix
#   xx = t(x) %*% x
#   inv_xx = solve(xx)
#   # calculate b_hat which will be used at mu_n calculation
#   # this b_hat is estimation of b coefficients in the model
#   b_hat = inv_xx %*% t(x) %*% y
#   # calculate mu_n (3x1)
#   mu_n = solve(xx + omega_0) %*% (xx%*%b_hat + omega_0 %*% mu_0)
#   # calculate omega_n (3x3)
#   omega_n = xx + omega_0
#   # calculate v_n
#   v_n = v_0 + length(y)
#   # calculate v_n*var_n as temporary
#   temp = v_0*var_0 + (t(y) %*% y +
#                          t(mu_0)%*%omega_0%*%mu_0 -
#                          t(mu_n)%*%omega_n%*%mu_n)
#   var_n = temp / v_n      # get var_n from the temp
```

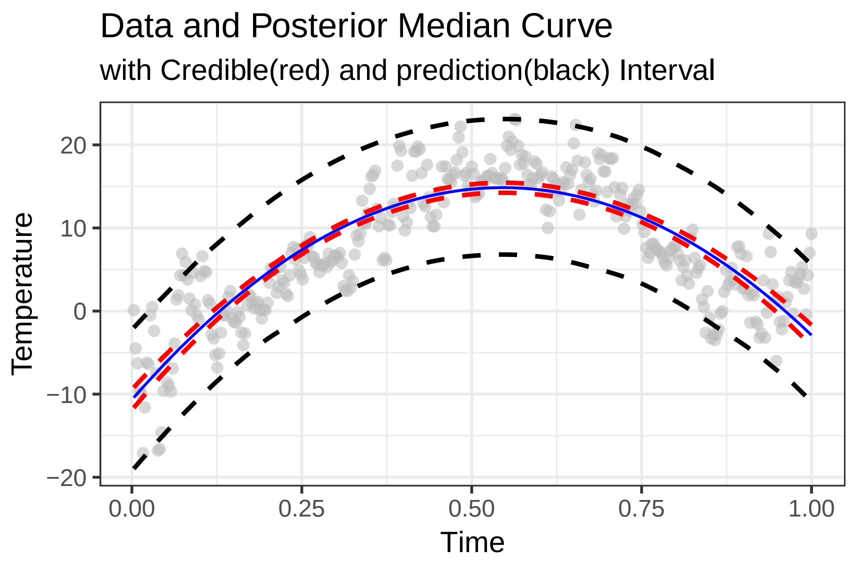
```r
#
#   return(r_post_var(n, v_n, var_n))
# }
######



# prepare the x data for multivariate prediction (normal)
# x should be in format as following:
# x^0; x^1; x^2
x = matrix(c(rep(1, dim(data_temp)[1]),
                  data_temp$time,
                  data_temp$time^2), ncol=3)
y = data_temp$temp

# # initial prior parameters
mu_0 = c(-10, 100, -100)
omega_0 = 0.01 * diag(3)
v_0 = 4
var_0 = 1
nDraw = 250

# # calculate <n> parameters
# calculate matrix multiplication of x data and inverse of it
# xx is 3x3 matrix
xx = t(x) %*% x
inv_xx = solve(xx)
# calculate b_hat which will be used at mu_n calculation
# this b_hat is estimation of b coefficients in the model
b_hat = inv_xx %*% t(x) %*% y
# calculate mu_n (3x1)
mu_n = solve(xx + omega_0) %*% (xx%*%b_hat + omega_0 %*% mu_0)

# calculate omega_n (3x3)
omega_n = xx + omega_0

# to draw from var posterior we need v_n, var_n
# calculate v_n first (integer)
v_n = v_0 + length(y)

# calculate var_n (integer)
# first we calculate v_n*var_n as temporary
temp = v_0*var_0 + (t(y) %*% y +
                     t(mu_0)%*%omega_0%*%mu_0 -
                     t(mu_n)%*%omega_n%*%mu_n)
var_n = temp / v_n                    # get var_n from the temp

# # Draw&Store coefficients and variances
set.seed(12345)
nDraw = 1000
var_drawn = numeric(nDraw)
coefs_drawn = matrix(numeric(3*nDraw), ncol=3)
```

```r
for(i in 1:nDraw){
  # draw from variance posterior
  var_drawn[i] = r_post_var(1, v_n, var_n)
  # draw from coefficient posterior
  coefs_drawn[i,] = r_post_coef(1, t(mu_n), var_drawn[i], omega_n)
}

# marginal posteriors.
# We dont draw from T-Student because we have conjugate prior here
# NOT non-informative prior. If it was the case we had to draw marginals from
# Beta | y \sim T_n-k(Beta_estimated, s^2*(X'X)^-1)
marg_b0 = coefs_drawn[,1]
marg_b1 = coefs_drawn[,2]
marg_b2 = coefs_drawn[,3]

hist1 = ggplot() +
  geom_density(aes(x = marg_b0, y=..density..),
               fill="lightblue", color="steelblue") +
  labs(subtitle = "B_0 Marginal Posterior",
       x = "B_0") +
  theme_bw()

hist2 = ggplot() +
  geom_density(aes(x = marg_b1, y=..density..),
               fill="lightblue", color="steelblue") +
  labs(subtitle = "B_1 Marginal Posterior",
       x = "B_1") +
  theme_bw()

hist3 = ggplot() +
  geom_density(aes(x = marg_b2, y=..density..),
               fill="lightblue", color="steelblue") +
  labs(subtitle = "B_2 Marginal Posterior",
       x = "B_2") +
  theme_bw()

hist4 = ggplot() +
  geom_density(aes(x = var_drawn, y = ..density..),
               fill="lightblue", color="steelblue") +
  labs(subtitle = "Variance Posterior",
       x = "var") +
  theme_bw()


# plot all marginal distibution histograms
grid.arrange(grobs = list(hist1, hist2, hist3, hist4), ncol=2)
```



```r
estimate_y = function(x, coefs){
  n = length(x)
  X = matrix(c(rep(1, n), x, x^2), ncol=3)
  head(X)
  y = X %*% coefs
  return(c(y))
}

# a matrix for all predicted y values. all rows will be an estimation.
# so the matrix will have nDraw (in this case1000) rows and obs count (366) cols.
y_estimated = matrix(0, nrow = nDraw, ncol = dim(data_temp)[1])
for(i in 1:nDraw){
  coef_set = coefs_drawn[i,]
  y_est = estimate_y(data_temp$time, coef_set)
  y_estimated[i, ] = y_est
}

#for the predictions bounadry we have to add the error in the model
#so because we have a quadratic model, we can add the error in the intercept
#all the other procedure is the same
y_estimated_error = matrix(0, nrow = nDraw, ncol = dim(data_temp)[1])
for(i in 1:nDraw){
  coef_set = coefs_drawn[i,]

  #add the error in the intercept
  coef_set[1] = coef_set[1] + rnorm(n = 1,mean = 0,sd = sqrt(var_drawn[i]))
  y_est = estimate_y(data_temp$time, coef_set)
  y_estimated_error[i, ] = y_est
}



# we have all estimations for each coefficient set
# we will caclulate the median of these estimations and we will obtain
# one "final" curve for our model.
```

```r
median_curve = apply(y_estimated, 2, median)

# Now we are asked for credible interval for this curve.
# We will calculate lower 2.5% and upper 97.5% posterior credible interval
# by using all our curves that we calculated from estimations of y
credible_boundaries = apply(y_estimated, 2, quantile, c(0.025, 0.975))

predictions_boundaries = apply(y_estimated_error, 2, quantile, c(0.025, 0.975))


# create dataframe to plot data, model, and credible intervals
df_plot = data.frame(x=data_temp$time,
                     y=data_temp$temp,
                     curve = median_curve,
                     lower_boundary = credible_boundaries[1, ],
                     upper_boundary = credible_boundaries[2, ],
                     lower_prediction = predictions_boundaries[1, ],
                     upper_prediction = predictions_boundaries[2, ])


plot = ggplot(df_plot) +
  geom_point(aes(x=x, y=y), alpha=0.6, color = "gray") +
  geom_line(aes(x=x, y=curve), color="blue") +
  geom_line(aes(x=x, y=lower_boundary), lty=2, color="red", size=0.8) +
  geom_line(aes(x=x, y=upper_boundary), lty=2, color="red", size=0.8) +
  geom_line(aes(x=x, y=lower_prediction), lty=2, color="black", size=0.8) +
  geom_line(aes(x=x, y=upper_prediction), lty=2, color="black", size=0.8) +
  labs(title = "Data and Posterior Median Curve",
       subtitle = "with Credible(red) and prediction(black) Interval",
       x="Time", y = "Temperature") +
  theme_bw()

plot
```



In the scaterplot above, we plot the data points, a curve for the posterior median and a credible interval. As it can easily observed, most of the points are outside of this interval. We should expect that, because we compute the credible interval for the median of the posterior, not for the data points. To be more specific, we want this interval not to be wide because in this way we are more confident about our estimators. We also plot the predictions bands,black dashed line, considering also the noise to our model. As we can see, most of the points are inside the predictions interval which is what we expected.

c) Find the maximal x of a quadratic function using simulation of the posterior.

It is of interest to locate the time with the highest expected temperature (that is, the time where f(time) is maximal). Let's call this value $\tilde{x}$. Use the simulations in b) to simulate from the posterior distribution of $\tilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\tilde{x}$ given β_0, β_1 and β_2.]

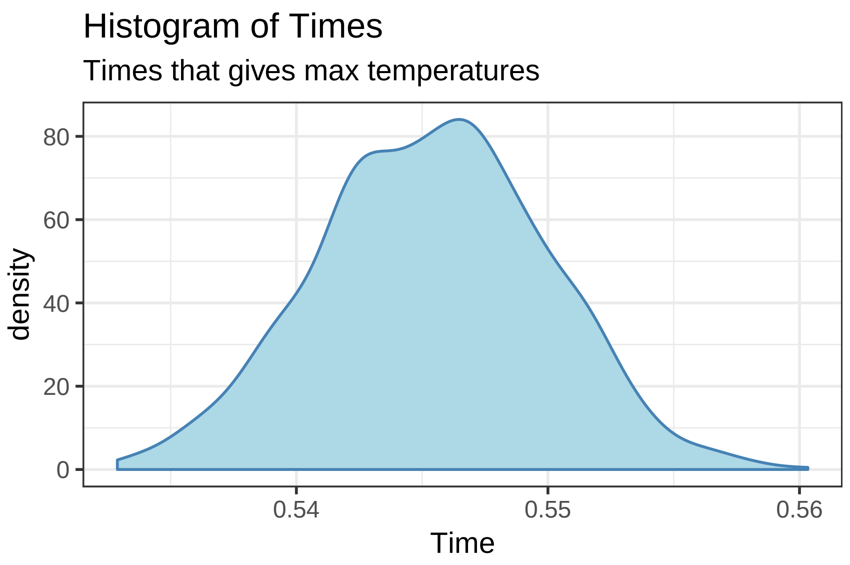
$$f(time) = \beta_0 + \beta_1 x + \beta_2 x^2$$

We want to find the time (x value) that gives maximum temperature. We know this is a quadratic function and since it is a curve, we can find optimal value for this function by making first derivation of this function equal to zero and solving equation for x. After applying this method we get a formula for this specific time value as below:

$$x = \frac{-\beta_1}{2\beta_2}$$

Now we can use this formula to find times that gives the maximum temperature by given β coefficients and we can see the density plot of these times.

```r
# we will calculate time values that gives the maximum temperature
# by using coefficient posterior that we sampled in the last task.
times_hightemp = -coefs_drawn[,2]/(2*coefs_drawn[,3])

plot_times_density = ggplot() +
  geom_density(aes(x = times_hightemp, y=..density..), fill="lightblue", color="steelblue") +
  labs(title = "Histogram of Times",
       subtitle = "Times that gives max temperatures",
       x = "Time") +
  theme_bw()

plot_times_density
```



As we can seen from the plot, the majority of the points are between 0.54-0.55. If we transope this number into a date, we can conclude that the hotest day will be at the end of July, which of course make sense.

d) Higher degree polynomial may not be needed(Lasso and Ridge).

Say now that you want to estimate a polynomial model of order 7, but you suspect that higher order terms may not be needed, and you worry about overfitting. Suggest a suitable prior that mitigates this potential problem. You do not need to compute the posterior, just write down your prior. [Hint: the task is to specify μ_0 and Ω_0 in a smart way.]

We will use Ridge regression as a model in order to avoid overfitting. In order to apply Ridge, We will take as prior for the coefficients, the multivariate normal distribution with $\mu = \vec{0}$ and covariance matrix $\sigma^2\Omega_0^{-1}$, where in this case $\Omega_0 = \lambda I$. So our goal now is to find the optimal penalize factor λ. We will use a bayessian approach, so we consider λ as an unknown parameter and we now need to specify only the starting parameters. We will take as λ prior the Gamma$(1, \frac{1}{2})$.

Question 2 Posterior approximation for classification with logistic regression

The dataset WomenWork.dat contains n = 200 observations (i.e. women) on the following nine variables:

a) Fit a logistic regression using MLE

Consider the logistic regression

$$Pr(y=1|x) = \frac{\exp(x^T\beta)}{1+\exp(x^T\beta)}$$

,

where y is the binary variable with y = 1 if the woman works and y = 0 if she does not. x is a 8-dimensional vector containing the eight features (including a one for the constant term that models the intercept). Fit the logistic regression using maximum likelihood estimation by the command: glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial). Note how I added a zero in the model formula so that R doesn't add an extra intercept (we already have an intercept term from the Constant feature). Note also that a dot (.) in the model formula means to add all other variables in the dataset as features. family = binomial tells R that we want to fit a logistic regression.

```r
data = read.table("datasets/WomenWork.dat", stringsAsFactors = F, header = T)

#fit a logistic regression model
glmModel = glm(Work ~ 0 + ., data = data, family = binomial)
```

b) Approximation of the posterior, credible intervals using marginal posterior and generated sample

Now the fun begins. Our goal is to approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution

$$\beta|y,X \sim N\left(\tilde{\beta}, J_y^{-1}(\tilde{\beta})\right)$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 \ln p(\beta|y)}{\partial\beta\partial\beta^T}\Big|_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 \ln p(\beta|y)}{\partial\beta\partial\beta^T}$ is an 8x8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 \ln p(\beta|y)}{\partial\beta_i\partial\beta_j^T}$ on the offdiagonal. It is actually not hard to compute this derivative by hand, but don't worry, we will let the computer do it numerically for you. Now, both $\tilde{\beta}$ and $J(\tilde{\beta})$ are computed by the optim function in R. See my code https://github.com/ mattiasvillani/BayesLearnCourse/raw/master/Code/MainOptimizeSpam.zip where I have coded everything up for the spam prediction example (it also does probit regression, but that is not needed here). I want you to implement you own version of this. You can use my code as a template, but I want you to write your own file so that you understand every line of your code. Don't just copy my code. Use the prior $\beta \sim N(0, \tau^2 I)$, with $\tau = 10$. Your report should include your code as well as numerical values for $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$ for the WomenWork data. Compute an approximate 95% credible interval for the variable NSmallChild. Would you say that this feature is an important determinant of the probability that a women works?

```r
library(mvtnorm)
set.seed(12345)

#When we have a distribution which its pdf can not be identified
```

```r
#i can use the approximation of the posterior
#we just need define the pdf and use optim function
#The approximation posterior dis is multivariate normal

y = as.vector(data$Work) # response variable
X = as.matrix(data[,-1]) #all the features
nPara = ncol(X) #the number of parameteres we want to estimate
mu = rep(0,nPara) # mean of the prior
Sigma = (100*diag(nPara)) #cov matrix of the prior

#pdf of the log-posterior
# we found it analytically, we have the prior and the likelihood
# we take the log of the likelihood and the prior so we have a sum

log_posterior = function(betas,mu,X,y,Sigma){

  a = X%*%betas
  Npara = ncol(X) #number of parameters

  # for the likelihood
  logLike = sum(y*a - log(1 + exp(a)))
  #if (abs(logLik) == Inf) logLik = -20000

  # for the prior we use the package for the pdf
  logPrior = dmvnorm(betas, mu, Sigma, log=TRUE)

  #we need the log post so the product of prior and likelihood is now sum
  return(logLike + logPrior)
}

#We have to define the initial values.
init <- rep(0,nPara)

OptimResults = optim(init,log_posterior,gr=NULL,mu,X,y,Sigma,
                    method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)

optim_coef = as.vector(OptimResults$par) #theta hat = optimal coef
info_matrix = -solve(OptimResults$hessian)
names(optim_coef) = names(glmModel$coefficients)
colnames(info_matrix) = names(glmModel$coefficients)
rownames(info_matrix) = names(glmModel$coefficients)

#function to generate from the posterior
# we know that the aproximate posterior dist is a multivariate normal
# with parameters the optimal coef and the hessian matrix
#we also have from optimResults

betas_samples = as.matrix(rmvnorm(n = 1000,
                                  mean = optim_coef,sigma = info_matrix))



#we have two ways to find the CI
```

```r
#FIRST way, using the generated sample and taking the quantiles
#transpose the matrix in order for each row to have one coef
#so we can add two columns in the end withe the quantiles

#betas_samples = as.data.frame(t(betas_samples))
#rownames(betas_samples ) = names(glmModel$coefficients)

#CI for all thetas
q_0.025 = apply(betas_samples, 2, FUN = quantile,0.025)
q_0.975 = apply(betas_samples, 2, FUN = quantile,0.975)

#betas_samples$low_0.025 = q_0.025
#betas_samples$up_0.975 = q_0.975
#eq_tail_NsmallChild = betas_samples[7,1001:1002]
eq_tail_NsmallChild = c(q_0.025[7],q_0.975[7])

#SECOND way using the marginal
#confidence interval using the marginal posterior
#The marginal posterior of the multinomial is normal
#with par:mean = mode of the posterior for the specific i(marginal i want)
#variance : The variance of the ith element calculated from the cov matrix
#Here we need the marginal for the coef of feature NsmallChild

#This is the mean parameter for the normal marginal
coef_NsmallChild = optim_coef[7] # take the mode for the coef we want the marg.

#This is the standard deviation for the normal marginal
sd_NsmallChild = sqrt(diag(info_matrix))[7] # take the sd we need

#We do not need to generate random samples from the marginal normal
#For every normal distribution we can take the mean +- 1.96*sd
#1.96 = qnorm(0.975) approximately

CI_NsmallChild = c(coef_NsmallChild - 1.96*sd_NsmallChild,coef_NsmallChild +
                     1.96*sd_NsmallChild)
```

In this task we want to approximate the posterior distribution of the 8-dim parameter vector β. Using the central limit theorem we know that every random variable, given a large sample, will follow a normal distribution. More specific, in this case, we can approximate the posterior distribution of β as a multivariate normal, using the parameters below.

$$\beta|y, X \sim N\left(\tilde{\beta}, J_y^{-1}(\tilde{\beta})\right)$$

The first step was to, analytically, find the log posterior of the parameters. The log-posterior is the sum of the prior and the likelihood(sum because we use the properties of logarithm). It is given that the prior is a multivariate normal distribution with known parameteres, so we can directly generate using the function dmvnorm. For the likelihood we have that ,

$$P(y|X,\beta) = \prod_{i=1}^{n} \frac{[\exp(x_i^T\beta)]^{y_i}}{1+\exp(x_i^T\beta)} \Rightarrow$$

$$LogP(y|X,\beta) = Log\prod_{i=1}^{n} \frac{[\exp(x_i^T\beta)]^{y_i}}{1+\exp(x_i^T\beta)} = \sum_{i=1}^{n} log\frac{[\exp(x_i^T\beta)]^{y_i}}{1+\exp(x_i^T\beta)} = \sum_{i=1}^{n} log\frac{\exp(x_i^T\beta y_i)}{1+\exp(x_i^T\beta)} \overset{\text{logarithm properties}}{\Rightarrow}$$

$$LogP(y|X,\beta) = \sum_{i=1}^{n} x_i^T \beta y_i - \log(1+\exp(x_i\beta))$$

Now, we need to found the parameters in order to generate from the multinormal posterior. We use the Taulor expansion of log-posterior around the posterior mode, but we do not need to do any derivations on our own. We use the function optim for that. Therefore the output of the optim function is the coefficients we want. The posterior mode vector is printed below. We also print the coefficients obtained from the task 2a using the logistic model and the glm function. The results are really close to each other.

```r
print(list("The coefficients of the model using glm are "= glmModel$coefficients))
```

```
## $`The coefficients of the model using glm are `
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.64430363 -0.01977457  0.17988062  0.16751274 -0.14435946 -0.08234033
## NSmallChild   NBigChild
## -1.36250239 -0.02542986
```

```r
print(list("The posterior mode vector is "= optim_coef))
```

```
## $`The posterior mode vector is `
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## NSmallChild   NBigChild
## -1.35913317 -0.02468351
```

```r
print(list("The information matrix is "= info_matrix))
```

```
## $`The information matrix is `
##                 Constant    HusbandInc     EducYears      ExpYears
## Constant     2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02
## HusbandInc   0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05
## EducYears   -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04
## ExpYears    -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03
## ExpYears2    0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02
## Age         -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04
## NSmallChild -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03
## NBigChild   -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04
##                 ExpYears2           Age   NSmallChild     NBigChild
## Constant     0.0457807243 -3.029345e-02 -0.1887483542 -0.0980239285
## HusbandInc   0.0001414915 -3.588562e-05  0.0005066847 -0.0001444223
## EducYears    0.0018962893 -3.240448e-06 -0.0061345645  0.0017527317
## ExpYears    -0.0142490853 -1.340888e-04 -0.0014689508  0.0005437105
## ExpYears2    0.0555786706 -3.299398e-04  0.0032082535  0.0005120144
## Age         -0.0003299398  7.184611e-04  0.0051841611  0.0010952903
## NSmallChild  0.0032082535  5.184161e-03  0.1512621814  0.0067688739
## NBigChild    0.0005120144  1.095290e-03  0.0067688739  0.0199722657
```

We also compute an approxiate 95% credible interval for the variable NSmallChild. We use two differents ways. For the first one, we use the generated sample for NSmallChild coefficient. For the second, we use the marginal posterior distribution. We know that the marginal posterior of a multivariate normal distribution is also a normal distribution with parameters $\mu = \tilde{\theta}$(NsmallChild) and standard deviation the square root of the diagonal for the NsmallChild parameter. As we can see from the two intervals, which are printed below, the feature NSmallChild is an important determinant of the probability that a woman works. The reason why, is because 0 is not inside its interval, so this parameter always plays role in the predictions.

```r
print(list("Credible interval using the generated sample "= eq_tail_NsmallChild))
```

```
## $`Credible interval using the generated sample `
## NSmallChild NSmallChild
##  -2.1296887  -0.5759195
```

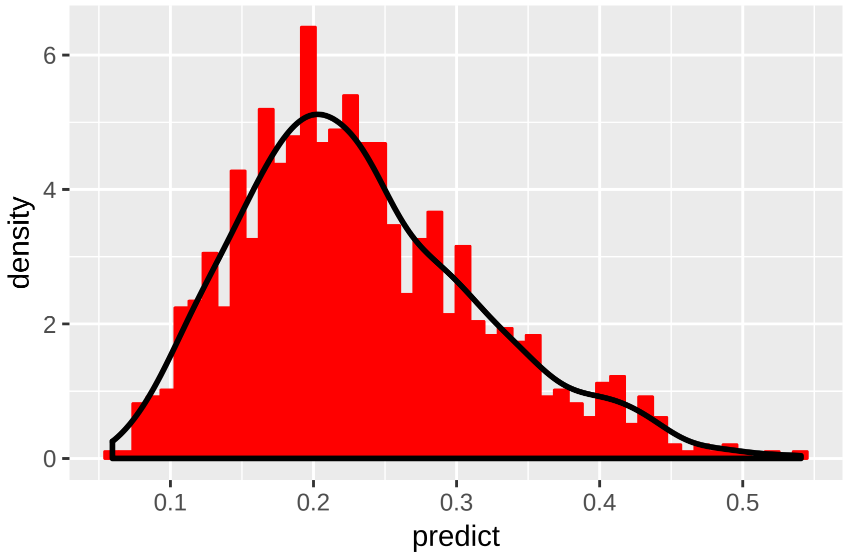
```r
print(list("Credible interval using the  marginal posterior distribution "=
             CI_NsmallChild))
```

```
## $`Credible interval using the  marginal posterior distribution `
## NSmallChild NSmallChild
##  -2.1214250  -0.5968414
```

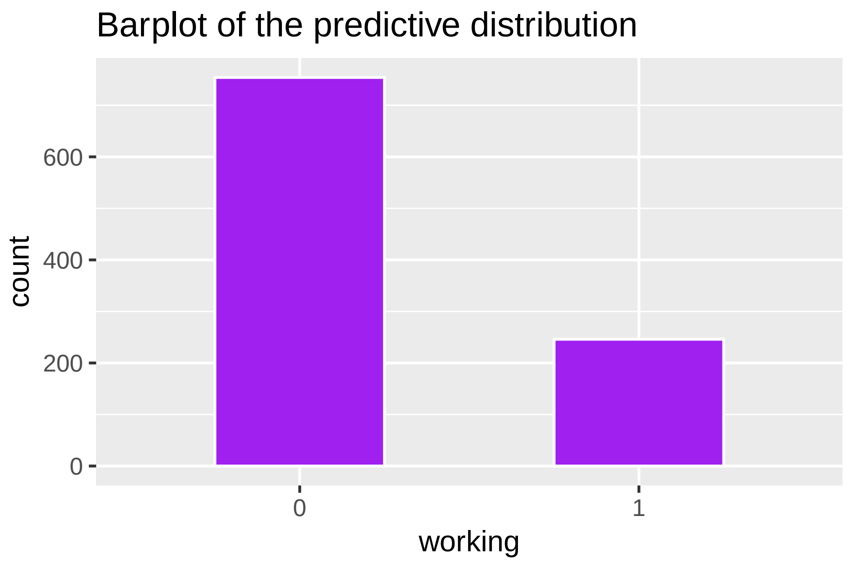
c) simulate from the predictive distribution using a normal approximation.

Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(b). Use that function to simulate and plot the predictive distribution for the Work variable for a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10. [Hint: the R package mvtnorm will again be handy. And remember my discussion on how Bayesian prediction can be done by simulation.]

```r
set.seed(12345)
library(ggplot2)
#We will use the coefficients above to simulate from the predictive distribution
#in order to simulate we need two steps:
# First simulate a sample from the posterior of the parameter
# For each vector of  parameters calculate the response variable

new_entry = c(1,10,8,10,1,40,1,1) #The features for the vector i want to predict

#function for the logistic distribution
logistic = function(linPred){

  res = exp(linPred)/(1+exp(linPred))

  return(res)
}

#vector to store the predictions
predict = rep(0,nrow(betas_samples))

#a for loop for simulating the predictive distribution
for (i in 1:nrow(betas_samples)) {

  linp = new_entry%*%as.vector(betas_samples[i,])
  predict[i] = logistic(as.numeric(linp))

}

ggplot(as.data.frame(predict)) +
  geom_histogram(mapping = aes(predict, y = ..density..),
                 color = "red", fill = "red", bins = 50)+
  geom_density(mapping = aes(predict), color = "black", size =1)
```



```r
#Now we have the distribution from the probabilities for the woman to work.
#We will convert them to zero and one(One is working, zero she is not)

predict_dist = rep(0,length(predict))
for (i in 1:length(predict)) {

  predict_dist[i] = rbinom(n = 1,size = 1,prob = predict[i])
}

#barplot of predictive distribution
ggplot()+
  geom_bar(mapping = aes(as.factor(predict_dist)), fill = "purple",color = "white", width = 0.5)+
  labs(title = "Barplot of the predictive distribution", x = "working")
```



```r
#probability of a woman working
prob_working = length(which(predict_dist==1))/length(predict_dist)
```

In this task we use the predictive distribution of the response variable in a logistic regression, using the normal approximation from the previous task. This procedure has 3 different steps. First, we simulate from the posterior distribution. In this task, we approximate the posterior distrubution and we generate a sample of 1000 observations. For each observation, we use the pdf of the logistic regression in order to compute the probabilities of working or not for the input. Finally, because we need the predictive distribution, we have to transpose those probabilities in to 0(not working) and 1(working). We plot the density of the probabilities of the predictive distribution and the histogramm of the predictive distribution. From the plots and the simulation of the predictive distribution we can say that the probability of a woman with the given features is approximately 0.246.