TDDE07 Bayesian Learning - Lab 2

Erik Linder-Norén - erino397

2017-04-26

1. Linear and polynomial regression

(a)

When choosing the parameters of the prior I reasoned that the intercept (temperature in the beginning of the year) would be around -10 degrees. The curve of the prior predictive line reaches its maximum slightly after 6 months and declines towards the end of the year.

$\mu_0 = [-10, 90, -80]$
$\Omega_0 = diag([.5, .5, .5])$
$\nu_0 = n - n_{params} = 366 - 3 = 363$
$\sigma_0^2 = 1$

(b)

I drew 10 beta samples and plotted the resulting curves. They can be seen plotted against the data in figure 1.

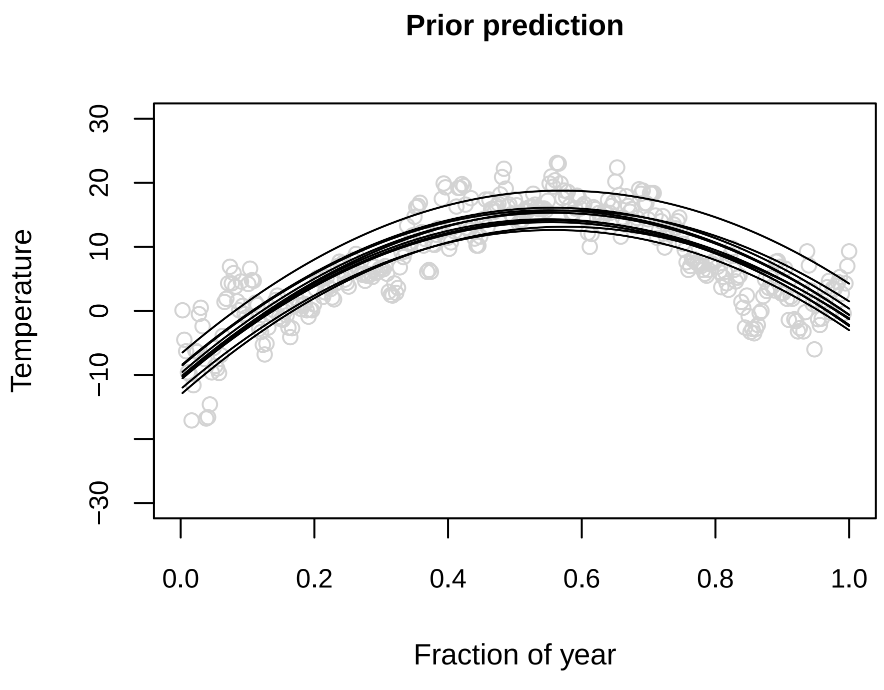

Figure 1: Predictions beta priors

(c):

I drew 1000 samples from the joint posterior distribution and calculated the 95% equal tail intervals for each collection of drawn beta parameters. I then plotted the line corresponding to the mean values of the beta distributions against the two lines corresponding to the lower and upper limit of the 95% credibility interval. The results can be seen in figure 2.

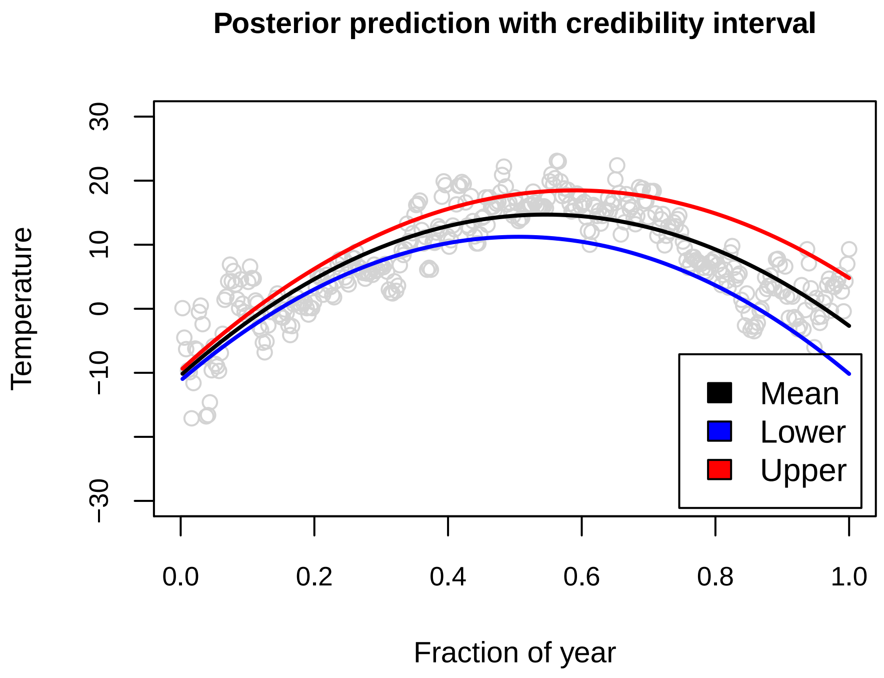

Figure 2: Prediction of the mean betas drawn from the posterior and the upper and lower limit of the 95% credibility interval

(d)

To compute the time that corresponded to the highest predicted temperature of the posterior I ran the following code.

```
i = which.max(preds_mean)
max_time = t[i]
```

Where i first calculate the index of the maximum temperature and then use that index to find the time corresponding to that index. In this piece of code pred_mean is a list containing the mean values of the beta draws from the posterior and t is the list containing the time. Multiplying the variable max_time by 366 gave the day that had the highest predicted temperature. That day was the 200th day.

(e)

To set the new prior parameters in a way that would combat overfitting by the new model with a higher complexity I defined the prior mean as the posterior mean calculated in (c) with the additional beta parameter means set to zero, and the prior covariance matrix defined as the previous posterior covariance matrix and the rest set to zero. The reason behind this is so that the new prior has the complexity of the model that gave the prediction in (c) and the prior for the additional complexity introduced set to zero.

	1	2	3	4	5	6	7
1	-10.39	92.00	-84.27	0.00	0.00	0.00	0.00

Table 1: Prior mean

	1	2	3	4	5	6	7	8
1	366.50	183.50	122.50	0.00	0.00	0.00	0.00	0.00
2	183.50	123.00	92.00	0.00	0.00	0.00	0.00	0.00
3	122.50	92.00	74.20	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: Prior covariance

2. Posterior approximation for classification with logistic regression

(b)

Computing the posterior mode $\tilde{\beta}$ and the observed Hessian evaluated at the posterior mode $J^{-1}(\tilde{\beta})$ I received values seen in table 3 and table 4 respectively.

	1	2	3	4	5	6	7	8
1	0.63	-0.02	0.18	0.17	-0.14	-0.08	-1.36	-0.02

Table 3: Posterior mode

Plotting the distribution of the beta parameter NSmallChild and its 95% credibility interval it could be seen that it ranged between $[-2.121, -0.597]$. Comparing that to the rest of the beta distributions I could see that those values were relatively large to the others. This feature should thereby be a good determinant of whether a woman works or not. The plot of the credibility interval for this parameter can be seen in figure 3.

(c)

I used the posterior mode and the posterior covariance calculated in (b) to draw beta samples from the posterior distribution of beta. I then used each beta sample together with the sample to be determined as working or not to calculate the probability using the inverse logit. This probability was then used as a parameter to draw a $\hat{y}$ from the predictive distribution. I ran the simulation for 1000 iterations and plotted the density of this distribution. The plot can be seen in figure 5. By this distribution it seems likely that the woman in question is not working. Calculating the number of draws where the prediction is 1 divided by the total number of draws gives an approximate probability of 23.7% of the woman working.

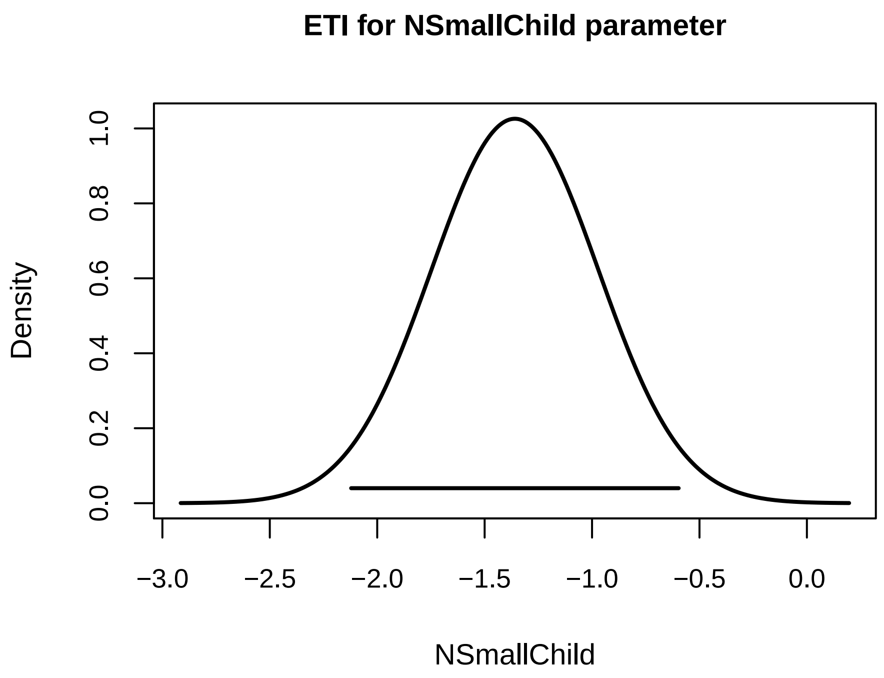

Figure 3: 95% credibility interval of NSmallChild probability distribution

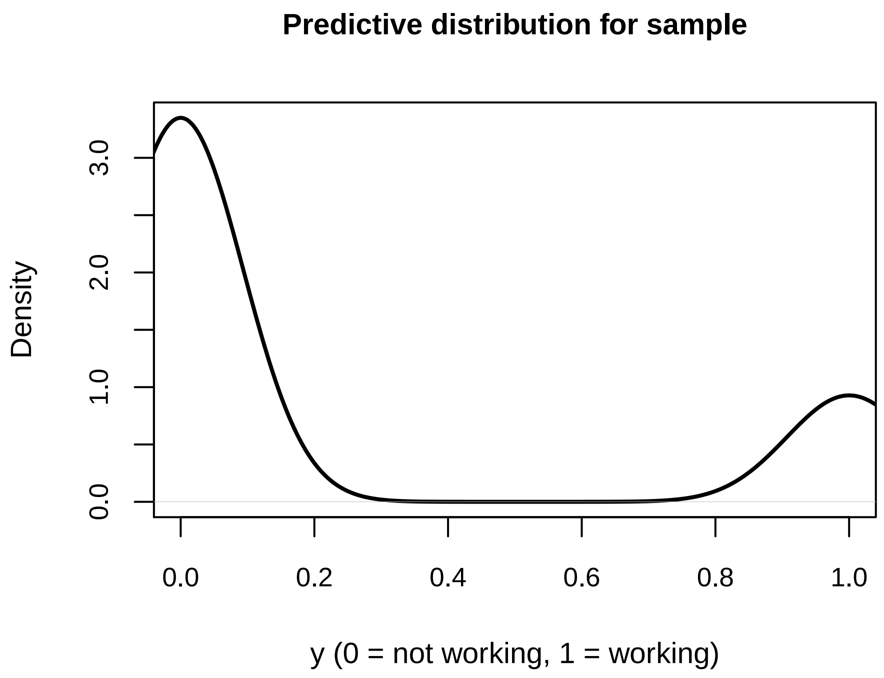

Figure 4: Predictive distribution

	1	2	3	4	5	6	7	8
1	2.27	0.00	-0.07	-0.01	0.05	-0.03	-0.19	-0.10
2	0.00	0.00	-0.00	-0.00	0.00	-0.00	0.00	-0.00
3	-0.07	-0.00	0.01	-0.00	0.00	-0.00	-0.01	0.00
4	-0.01	-0.00	-0.00	0.00	-0.01	-0.00	-0.00	0.00
5	0.05	0.00	0.00	-0.01	0.06	-0.00	0.00	0.00
6	-0.03	-0.00	-0.00	-0.00	-0.00	0.00	0.01	0.00
7	-0.19	0.00	-0.01	-0.00	0.00	0.01	0.15	0.01
8	-0.10	-0.00	0.00	0.00	0.00	0.00	0.01	0.02

Table 4: Observed hessian evaluated at the posterior mode

Assignment 1

```r
require(MASS)
require(geoR)


grid_w = 5
grid_h = 4


# Lab 2 - Assignment 1

df = read.table("data/TempLinkoping2016.txt", header=TRUE)

n = 366
n_params = 3

# (a)

# Linear model (test)
fit = lm(temp ~ time + I(time^2), data=df)
summary(fit)


mu0 = c(-10, 90, -80)
covar0 = diag(c(.5, .5, .5))
v0 = n - n_params
sigma_sq0 = 1

# (b)

t = df[["time"]]
temp = df[["temp"]]

pdf("plots/1_2_prior_draws.pdf", width=grid_w, height=grid_h)

plot(t,
     temp,
     type="p",
     col="lightgray",
     ylim=c(-30, 30),
     xlim=c(0,1),
     xlab="Fraction of year",
     ylab="Temperature",
     main="Prior prediction",
     cex.main=.9,
     cex.lab=.9,
     cex.axis=.8)

n_draws = 10
for (iter in 1:n_draws) {
  sigma_sq = rinvchisq(n=1, df=v0, scale=sigma_sq0)
  beta = mvrnorm(n=1, mu=mu0, Sigma=sigma_sq*ginv(covar0))
  error = rnorm(n=1, 0, 1)
  preds = beta[1] + beta[2]*t + beta[3]*I(t)^2 + error
  lines(t, preds)
}

dev.off()

# (c)

# 366x3 (1 t t^2)
X = t(rbind(rep(1,n), t, I(t)^2))
y = as.vector(temp)

X_X = t(X) %*% X

# Least squares approximate of beta
beta_hat = ginv(X_X) %*% t(X) %*% y

# Posterior parameters
mu_n = ginv(X_X + covar0)%*%(X_X%*%beta_hat+covar0%*%mu0)
covar_n = X_X + covar0
v_n = v0 + n
sigma_sq_n = as.double((1/v_n)*(v0%*%sigma_sq0 + (t(y)%*%y + t(mu0)%*%covar0%*%mu0 - t(mu_n)%*%covar_n%

pdf("plots/1_3_posterior_draws.pdf", width=grid_w, height=grid_h)

plot(t,
     temp,
     type="p",
     col="lightgray",
     ylim=c(-30, 30),
     xlim=c(0,1),
     xlab="Fraction of year",
     ylab="Temperature",
     main="Posterior prediction with credibility interval",
     cex.main=.9,
     cex.lab=.9,
     cex.axis=.8)

beta_1s = c()
beta_2s = c()
```

```r
beta_3s = c()

n_draws = 1000
for (iter in 1:n_draws) {
  sigma_sq = rinvchisq(n=1, df=v_n, scale=sigma_sq_n)
  beta = mvrnorm(n=1, mu=mu_n, Sigma=sigma_sq*ginv(covar_n))
  error = rnorm(n=1, 0, sigma_sq)
  # Add beta draws
  beta_1s = c(beta_1s, beta[1])
  beta_2s = c(beta_2s, beta[2])
  beta_3s = c(beta_3s, beta[3])
}

error = 0

# Line using mean values from simulation
beta_1 = mean(beta_1s)
beta_2 = mean(beta_2s)
beta_3 = mean(beta_3s)
preds_mean = beta_1 + beta_2*t + beta_3*I(t)^2 + error
lines(t, preds_mean, lwd=2)

# Equal tail interval
beta_1_eti = quantile(beta_1s, probs=c(0.025, 0.975))
beta_2_eti = quantile(beta_2s, probs=c(0.025, 0.975))
beta_3_eti = quantile(beta_3s, probs=c(0.025, 0.975))

# Lower
lower_beta_1 = beta_1_eti[1]
lower_beta_2 = beta_2_eti[1]
lower_beta_3 = beta_3_eti[1]
preds_low = lower_beta_1 + lower_beta_2*t + lower_beta_3*I(t)^2 + error
lines(t, preds_low, col="blue", lwd=2)

# Higher
higher_beta_1 = beta_1_eti[2]
higher_beta_2 = beta_2_eti[2]
higher_beta_3 = beta_3_eti[2]
preds_high = higher_beta_1 + higher_beta_2*t + higher_beta_3*I(t)^2 + error
lines(t, preds_high, col="red", lwd=2)

legend("bottomright",
       legend = c("Mean","Lower", "Upper"),
       fill = c("black", "blue", "red"),
       inset = 0.02)

dev.off()

# (d)

pdf("plots/1_4_warm_days.pdf", width=grid_w, height=grid_h)

# Could also solve for time by derivation set to zero
```

```r
i = which.max(preds_mean)
max_time = t[i]
max_day = round(366 * max_time)

# Std dev chosen by prediction plot
xGrid = seq(0, 366, 1)
n = dnorm(xGrid, mean=max_time*366, sd=0.1*366)
plot(xGrid,
     n,
     type="l",
     lwd=2,
     xlab="Days",
     ylab="Probability",
     main="Prob. distribution of day with hottest temperature",
     cex.main=.9,
     cex.lab=.9,
     cex.axis=.8)

dev.off()

# (e)

# Chooses new mu0 and covar0 as the previous posterior hyper parameters and the additional
# betas set to zero (to combat overfitting).
mu0 = c(mu_n, 0, 0, 0, 0)
covar0 = matrix(0, nrow=8, ncol=8)
covar0[1:3, 1:3] = covar_n
```

Assignment 2

```r
require(MASS)
require(geoR)
library(mvtnorm)
library(LaplacesDemon)


grid_w = 5
grid_h = 4


# Lab 2 - Assignment 2

df = read.table("data/WomenWork.dat.txt", header=TRUE)

# (a)

glmModel <- glm(Work ~ 0 + ., data = df,family = binomial)
summary(glmModel)

# (b)

y = df[["Work"]]
X = as.matrix(df[,2:9])
headers = colnames(df)
n = dim(X)[1]
n_params = dim(X)[2]

tau_sq = 100 # tau = 10

# Initialize prior hyper parameters
mu_prior <- as.vector(rep(0,n_params))
sigma_prior = diag(tau_sq, n_params, n_params)

# Calculate the log posterior
LogPosteriorLogistic <- function(betas, y, X, mu, Sigma){

  # Multiply data by parameters to get predictions
  predictions <- X%*%betas;

  # Log likelihood
  log_likelihood <- sum(predictions*y - log(1 + exp(predictions)));
  if (abs(log_likelihood) == Inf) log_likelihood = -20000;

  # Log prior
  log_prior <- dmvnorm(betas, mu_prior, sigma_prior, log=TRUE);

  # Sum of log likelihood and log prior is log posterior
  return(log_likelihood + log_prior)
}

log_posterior = LogPosteriorLogistic
```

```r
# Initialize as zeros
init_betas = rep(0, n_params)

opt_results = optim(init_betas,
                    log_posterior,
                    gr=NULL,
                    y,
                    X,
                    mu_prior,
                    sigma_prior,
                    method=c("BFGS"),
                    control=list(fnscale=-1),
                    hessian=TRUE)

# Posterior mode (beta hat)
post_mode = opt_results$par
# Posterior covariance (J^-1(beta hat))
post_cov = -solve(opt_results$hessian)
approx_post_std_dev = sqrt(diag(post_cov))

pdf("plots/2_2_nsmallchild_cred_interval.pdf", width=grid_w, height=grid_h)

# Plot NSmallChild parameter
pmode = post_mode[7]
pstd = approx_post_std_dev[7]
beta_grid = seq(pmode - 4*pstd, pmode + 4 * pstd, length=1000)
eti = qnorm(c(0.025, 0.975), pmode, pstd)
dn = dnorm(x=beta_grid, mean=pmode, sd=pstd)
plot(beta_grid,
     dn,
     type = "l",
     lwd = 2,
     main="ETI for NSmallChild parameter",
     ylab = 'Density', xlab=headers[8],
     cex.main=.9,
     cex.lab=.9,
     cex.axis=.8)
lines(eti, rep(0.04, 2), col="black", lwd=2)

dev.off()

# (c)

# Sample to predict
constant = 1
husband = 10
edu_years = 8
exp_years1 = 10
exp_years2 = (exp_years1/10)^2
age = 40
n_small_child = 1
n_big_child = 1
```

```
sample = c(constant,
           husband,
           edu_years,
           exp_years1,
           exp_years2,
           age,
           n_small_child,
           n_big_child)

y_draws = c()
n_draws = 1000
for (i in 1:n_draws){
  # Draw a beta
  beta_draw = as.vector(rmvnorm(n=1, mean=post_mode, sigma=post_cov))
  e = exp(sample%*%beta_draw)
  # Calculate the probability (bernoulli parameter)
  p = e / (1 + e)
  # Draw a y prediction
  y_draw = rbern(n=1, prob=p)
  y_draws = c(y_draws, y_draw)
}

outcomes = table(y_draws)
n_working = outcomes[names(outcomes)==1]
prob_working = n_working / length(y_draws)

pdf("plots/2_3_pred_distr.pdf", width=grid_w, height=grid_h)

prob_density = density(y_draws)
plot(prob_density,
     type="l",
     lwd=2,
     xlim=c(0,1),
     ylab="Density",
     xlab="y (0 = not working, 1 = working)",
     main="Predictive distribution for sample",
     cex.main=.9,
     cex.lab=.9,
     cex.axis=.8)

dev.off()
```