Lectures_6_to_8

November 25, 2016

1 Statistical modeling and Inference

1.1 José Fernando Moreno Gutiérrez (GSE-MDS138478)

1.1.1 Problem set # 2 from slides

1.1.2 Slides 6

Exercise 2.1 If $t = 1$

$$\begin{aligned} p(t = 1|\mathbf{x}) &= \frac{1}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \\ &= \frac{1}{1 + \exp^{-t(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \end{aligned}$$

If $t = -1$

$$\begin{aligned} p(t = -1|\mathbf{x}) &= 1 - \frac{1}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \\ &= \frac{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} - \frac{1}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \\ &= \frac{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)} - 1}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \\ &= \frac{\exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}}{1 + \exp^{-(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \\ &= \frac{1}{1 + \exp^{\mathrm{w}^T\Phi(\mathbf{x})+b}} \\ &= \frac{1}{1 + \exp^{-t(\mathrm{w}^T\Phi(\mathbf{x})+b)}} \end{aligned}$$

Hence:

$$p(t|\mathbf{x}) = \frac{1}{1 + \exp^{-t(\mathrm{w}^T\Phi(\mathbf{x})+b)}}$$

Exercise 2.2 We are going to minimise:

$$\frac{1}{N}\sum_n L\left(t_n(\Phi(\mathbf{x}_n)^T\mathbf{w}+b)\right)+\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Where the loss function $L(u)=\log(1+\exp^{-u})$. Therefore the large margin w is:

$$\mathrm{w}_{LM}=\frac{1}{N}\sum_n \log\left(1+\exp^{-t_n(\Phi(\mathbf{x}_n)^T\mathbf{w}+b)}\right)+\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Then similar as logistic regression likelihood with gaussian prior:

$$p(\mathbf{w}|\mathbf{t})\propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w})$$

We have:

$$\begin{aligned}\log(p(\mathbf{w}))&=-\frac{1}{2}(\mathbf{w}-\mathbf{0})^T\lambda N(\mathbf{w}-\mathbf{0})\\&=-\frac{1}{2}\lambda N\mathbf{w}^T\mathbf{w}\end{aligned}$$

$$\begin{aligned}\log(p(\mathbf{t}|\mathbf{w}))&=\sum_n\log\left(\frac{1}{1+\exp^{-\mathbf{t}_n(\Phi(x_n)^T\mathbf{w}+b)}}\right)\\&=\sum_n-\log\left(1+\exp^{-\mathbf{t}_n(\Phi(x_n)^T\mathbf{w}+b)}\right)\end{aligned}$$

So:

$$\log(p(\mathbf{w}|\mathbf{t}))=\sum_n-\log\left(1+\exp^{-\mathbf{t}_n(\Phi(x_n)^T\mathbf{w}+b)}\right)-\frac{1}{2}\lambda N\mathbf{w}^T\mathbf{w}+c$$

The stationary point of this (w.r.t w) will be the same as for the large margin classifier.

1.1.3 Slides 7

Exercise 2

$$\begin{aligned}p(z_k=1|\mathbf{x}_{N+1})&=\frac{p(\mathbf{x}_{N+1}|z_k=1)p(z_k=1)}{\sum_{j=1}^K p(\mathbf{x}_{N+1}|z_j=1)p(z_j=1)}\\&=\frac{\pi_k N(\mathbf{x}_{N+1}|\mu_k,\mathbf{Q}_k^{-1})}{\sum_{j=1}^K\pi_j N(\mathbf{x}_{N+1}|\mu_j,\mathbf{Q}_j^{-1})}\\&=\frac{\pi_k|\mathbf{Q}_k|^{\frac{1}{2}}e^{-\frac{1}{2}(\mathbf{x}_{N+1}-\mu_k)^T\mathbf{Q}_k(\mathbf{x}_{N+1}-\mu_k)}}{\sum_{j=1}^K\pi_j|\mathbf{Q}_j|^{\frac{1}{2}}e^{-\frac{1}{2}(\mathbf{x}_{N+1}-\mu_j)^T\mathbf{Q}_j(\mathbf{x}_{N+1}-\mu_j)}}\end{aligned}$$

Exercise 5 In robust regression model we consider a Gamma prior $p(\eta_n) = \text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2} - 1\right)$ and a normal likelihood $p(\mathbf{t}|\mathbf{X}, \mathbf{w}, q, \eta) = \mathcal{N}(\mathbf{\Phi}\mathbf{w}, (q\text{diag}(\eta))^{-1})$. To compute the posterior we use Bayes theorem, obtaining

$$p(\eta_n|t_n, \nu) = \frac{p(t_n|\eta_n, \nu)p(\eta_n|\nu)}{p(t_n|\nu)} \propto p(t_n|\eta_n, \nu)p(\eta_n|\nu) = \text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2} - 1\right)\mathcal{N}(\mathbf{\Phi}(x_n)\mathbf{w}, (q\eta_n)^{-1}).$$

The pdf of the gamma distribution is $\text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2} - 1\right) \propto \eta_n^{\frac{\nu}{2}-1} e^{-(\frac{\nu}{2}-1)\eta}$ and the pdf of Gaussian is $\mathcal{N}(\mathbf{\Phi}(x_n)\mathbf{w}, (q\eta_n)^{-1}) \propto \eta_n^{\frac{1}{2}} e^{\frac{1}{2}((t_n - \mathbf{\Phi}(x_n)\mathbf{w})^T q\eta_n(t_n - \mathbf{\Phi}(x_n)\mathbf{w}))}$. So, the posterior would be

$$p(\eta_n|t_n, \nu) \ \propto \ \eta_n^{\frac{\nu}{2}-1} e^{-(\frac{\nu}{2}-1)\eta_n} \eta_n^{\frac{1}{2}} e^{\frac{1}{2}(s_n q\eta_n s_n)} = \eta_n^{\frac{\nu}{2}+\frac{1}{2}-1} e^{-(\frac{\nu}{2}-1)\eta_n + \frac{1}{2}(s_n^2 q\eta_n)} = \eta_n^{\frac{\nu}{2}+\frac{1}{2}-1} e^{(\frac{\nu+qs_n^2}{2}-1)\eta_n}.$$

where $s_n = t_n - \mathbf{\Phi}(x_n)\mathbf{w}$. The last form is a $\text{Gamma}\left(\frac{\nu+1}{2}, \frac{\nu+qs^2}{2} - 1\right)$, so

$$p(\eta_n|t_n, \nu) \ \sim \ \text{Gamma}(\frac{\nu+1}{2}, \frac{\nu+qs^2}{2} - 1).$$

$z_n|\mathbf{w}, \mathbf{x}_n, t_n$ in the probit model:

$$p(z_n|\mathbf{w}, \mathbf{x}_n, t_n) = \frac{p(t_n|\mathbf{w}, \mathbf{x}_n, z_n)p(z_n|\mathbf{w}, \mathbf{x}_n)}{\int p(t_n|\mathbf{w}, \mathbf{x}_n, z_n)p(z_n|\mathbf{w}, \mathbf{x}_n)dz_n}$$
$$= \frac{[p(z_n > 0|\mathbf{w}, \mathbf{x}_n)^{t_n}(1 - p(z_n > 0|\mathbf{w}, \mathbf{x}_n))^{1-t_n}]N(z_n|\Phi(\mathbf{x}_n)^T\mathbf{w}, q)}{\int[p(z_n > 0|\mathbf{w}, \mathbf{x}_n)^{t_n}(1 - p(z_n > 0|\mathbf{w}, \mathbf{x}_n))^{1-t_n}]N(z_n|\Phi(\mathbf{x}_n)^T\mathbf{w}, q)dz_n}$$

$\mathbf{z}_n|\mathbf{x}_n, \mu_{1:K}, \mathbf{Q}_{1:K}$ in the Gaussian mixture model:

$$p(\mathbf{z}_n|\mathbf{x}_n, \mu_{1:K}, \mathbf{Q}_{1:K}) = \frac{p(\mathbf{x}_n|\mathbf{z}_n, \mu_{1:K}, \mathbf{Q}_{1:K})p(\mathbf{z}_n)}{\sum_{j=1}^K p(\mathbf{x}_n|z_{nj} = 1, \mu_j, \mathbf{Q}_j)p(z_{nj} = 1)}$$
$$= \frac{\Pi_{i=1}^K[\pi_i p(\mathbf{x}_n|\mathbf{z}_n, \mu_i, \mathbf{Q}_i^{-1})]^{z_{nk}}}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n|z_{nj} = 1, \mu_j, \mathbf{Q}_j)}$$
$$= \frac{\Pi_{i=1}^K[\pi_i N(\mathbf{x}_n|\mu_i, \mathbf{Q}_i^{-1})]^{z_{nk}}}{\sum_{j=1}^K \pi_j N(\mathbf{x}_n|\mu_j, \mathbf{Q}_j^{-1})}$$

$\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}, \mathbf{\Sigma}$ in the factor model:

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}, \mathbf{\Sigma}) = \frac{p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{W}, \mathbf{\Sigma})p(\mathbf{z}_n)}{p(\mathbf{x}_n)}$$
$$= \frac{N(\mathbf{x}_n|\mu + \mathbf{W}\mathbf{z}_n, \mathbf{\Sigma})N(\mathbf{0}_k, \mathbf{I}_k)}{N(\mu, \mathbf{W}\mathbf{W}^T + \mathbf{\Sigma})}$$

which will be Gaussian distribution.

1.1.4 Slides 8

Exercise 1 See Slides 7 exercise 5.

Exercise 2 We define $Q(\theta, \theta') = \mathtt{E}[\log(p(\mathbf{t}, \eta|\theta)|\theta', \mathbf{x}]$. Applying conditional probabilities we get that

$$Q(\theta, \theta') = \mathtt{E}[\log(p(\mathbf{t}|\eta, \theta)) + \log p(\eta|\theta)|\theta, \mathbf{x}].$$

$p(\eta)$ only depends on ν and θ', so it is constant over $\boldsymbol{\theta}$. Then, expanding the expectation we get

$$\begin{aligned}\mathtt{E}\log(p(\mathbf{t}|\boldsymbol{\eta}, \theta)|\boldsymbol{\theta}', \mathbf{x}] &= \mathtt{E}\left[\frac{1}{2}\log q - \frac{q}{2}(\mathbf{t} - \mathbf{\Phi w})^T \mathrm{diag}(\boldsymbol{\eta})(\mathbf{t} - \mathbf{\Phi w})\right] + C \\ &= \frac{N}{2}\log q - \frac{q}{2}(\mathbf{t} - \mathbf{\Phi w})^T \mathrm{diag}(\mathtt{E}[\boldsymbol{\eta}|\mathbf{t}, \mathbf{X}, \boldsymbol{\theta}'](\mathbf{t} - \mathbf{\Phi w})) + C,\end{aligned}$$

where $p(\boldsymbol{\eta}|\mathbf{t}, \nu) = \mathrm{Gamma}(\frac{\nu+1}{2}, \frac{\nu + q'(\mathbf{t} - \mathbf{\Phi w}')^2}{2} - 1)$, so his expectation will be a vector with the elements (applying expectation of a Gamma distribution)

$$\frac{\nu + 1}{\nu + \frac{q'}{2}(t_n - \Phi(x_n)^T \mathbf{w}')^2 - 1}$$

Exercise 3 The following is the code used to produce the graphics and the proposals for exercise 3

```
In [ ]: # Author: José Fernando Moreno Gutiérrez

        # load libraries and functions needed
        library(dplyr)
        library(ggplot2)
        library(ggthemes)
        library(grid)
        library(gridExtra)

        m <- 30 # number of variables taken for the model

        # load data - Find it on https://1drv.ms/t/s!Ai0XbELt7PXquFJtWenDbxvksoXM #
        data <- read.table(file = "~/Documents/OneDrive/Documents/BGSE/First_Term/S
                           nrow = 300)[,1:(m + 1)]

        # Initial parameters
        w0 <- rep(1,31)
        q0 <- 1
        nu <- 10
        t <- as.vector(data[, 1])
        X <- as.matrix(cbind(rep(1,nrow(data)), as.matrix(data[,2:31])))
        N <- length(t)

        mle_estimator_lm <- function (w, t_vector, phi) {
          if(is.vector(t_vector) == FALSE)
            t_vector <- as.vector(t_vector)
          if(is.matrix(phi) == FALSE)
```

```
    phi <- as.matrix(phi)
  M <- ncol(phi)
  phi_w <- phi %*% w[1:M]
  Sig <- w[(M + 1):(M + 1)]
  sum(-(1 / 2) * log(2 * pi) - (1 / 2) * log(Sig ^ 2) - (1 / (2 * Sig ^ 2))
}

mle_result <- function (mle_estimation_result, t_vector, phi) {
  phi          <- as.matrix(phi)
  w            <- as.matrix(mle_estimation_result$par[-length(mle_estimation_
  sigma        <- last(mle_estimation_result$par)
  variance     <- -solve(mle_estimation_result$hessian)
  w_se         <- sqrt(diag(variance))[1:length(w)]
  sigma_se     <- last(sqrt(diag(variance)))
  z_value      <- w/w_se
  p_value      <- 2 * (1 - pnorm(abs(z_value)))
  t_hat        <- phi %*% w
  e            <- t_vector - t_hat
  e_st         <- e/sigma_se
  leverage     <- NULL
  for(i in 1:nrow(phi)){
    leverage <- c(leverage, t(phi[i,]) %*% solve(t(phi) %*% phi) %*% phi[i,
  }
  dev <- -2 * ((1 / 2) * log(2 * pi) - (1 / 2) * log(sigma ^ 2) - (1 / (2 *
  # leverage is equivalent to diag(phi %*% solve(t(phi) %*% phi) %*% t(phi)
  # diagonal of the hat matrix
  return(list(w = w, sigma = sigma, variance = variance, w_se = w_se, sigma
              z_value = z_value, p_value = p_value, t_hat = t_hat, e = e, e
              leverage = leverage, dev = dev, n = nrow(phi)))
}

mle_plot <- function(mle_results){
  ci_plot_data <- data.frame(coeff_number = seq(1:nrow(mle_results$w))) %>%
    bind_cols(data.frame(w = mle_results$w)) %>%
    mutate(lower_bound = w - 1.96 * mle_results$w_se) %>%
    mutate(upper_bound = w + 1.96 * mle_results$w_se) %>%
    mutate(color_ci = ifelse(mle_results$p_value < 0.05, "blue", "red"))

  ci_plot <- ggplot(data = ci_plot_data, aes(x = as.factor(coeff_number), y
    geom_errorbar(aes(ymin = lower_bound, ymax = upper_bound),
                  color = ci_plot_data$color_ci, width = 0.3) +
    geom_point() + ggtitle("Maximum Likelihood Estimator") +
    labs(x = "Coefficient", y = "w, w +/- 1.96 s.e.") + theme_excel()

  de_plot_data <- data.frame(observation_number = seq(1:length(mle_results$
    bind_cols(data.frame(dev = mle_results$dev)) %>%
    mutate(color_de = ifelse(dev > as.numeric(quantile(mle_results$dev, c(0
                             dev < as.numeric(quantile(mle_results$dev, c(0
```

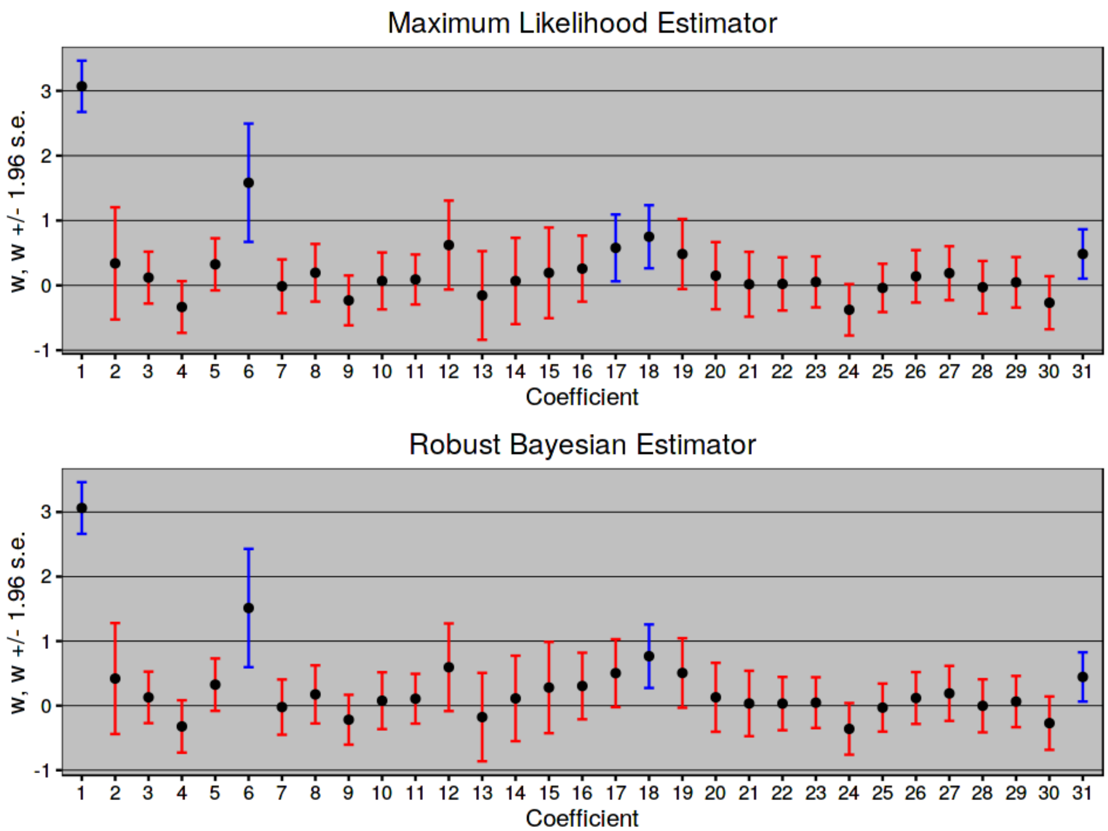
```
  de_plot <- ggplot(data = de_plot_data, aes(x = observation_number, y = de
    geom_point(color = de_plot_data$color_de) +  geom_hline(yintercept =  c

    geom_hline(yintercept =  quantile(mle_results$dev, c(0.005, 0.995))[2])
    ggtitle("Maximum Likelihood Estimator") +
    labs(x = "Observation", y = "Deviance Error") + theme_excel()

  return(list(ci_plot = ci_plot, de_plot = de_plot))
}

# mle_estimation
mle_estimation_result <- optim(runif(m + 2, 0, 1), mle_estimator_lm, phi =
                               t_vector = t, method = "BFGS",
                               control = list(trace = 1, maxit = 10000, fns
                               hessian = TRUE)

# mle_results
mle_results <- mle_result(mle_estimation_result, t, X)

# mle graphics
mle_graphics <- mle_plot(mle_results)

# EM algorithm functions
e_step <- function(t, X, q, w, nu) {
  eta <- (nu + 1) / (nu + q * (t - X %*% w) ^ 2 - 2)
  return(eta)
}

m_step <- function(t, X, q0, w0, eta) {
  w <- solve(t(X) %*% diag(as.vector(eta)) %*% X, t(X) %*% diag(as.vector(e
  q <- (N / ( t(t - X %*% w) %*% diag(as.vector(eta)) %*% (t - X %*% w)))
  return(list(w = as.vector(w), q = as.numeric(q)))
}

rob_reg <- function(t, X, q0, w0, nu, max_iter = 100) {
  q <- q0
  w <- w0
  q_conv      <- NULL
  e_conv      <- NULL
  loglik_conv <- NULL
  for(i in 1:max_iter) {
    eta    <- e_step(t, X, q, w, nu)
    wq     <- m_step(t, X, q, w, eta)
    q      <- wq$q
    w      <- wq$w
    loglik <- (length(t) / 2) * log(q) - q/2 * (t((t - X %*% w)) * diag(eta
    q_conv[i]      <- q
```

```r
    e_conv[i]       <- sum((t - X %*% w) ^ 2)
    loglik_conv[i] <- loglik
  }
  e   <- as.vector(t - X %*% w)
  se  <- as.vector(sqrt(diag(solve(q * t(as.vector((nu + 1) * (nu - 2 - q *
                                                    ((nu + q * e ^ 2 - 2) ^
  dev <- NULL
  for(i in 1:length(t)){
    dev[i] <- -2 * ((1 / 2) * log(q) - q/2 * (t[i] - X[i,] %*% w) * diag(et
  }
  return(list(w = w, q = q, eta = eta, e = e, se = se, e_conv = e_conv, q_c
              loglik_conv = loglik_conv, dev = dev))
}

# Execution
rob_reg_results <- rob_reg(t, X, q0, w0, nu)

rob_reg_plot <- function(rob_reg_results){
  ci_plot_data <- data.frame(coeff_number = seq(1:length(rob_reg_results$w)
    bind_cols(data.frame(w = rob_reg_results$w)) %>%
    mutate(lower_bound = w - 1.96 * rob_reg_results$se) %>%
    mutate(upper_bound = w + 1.96 * rob_reg_results$se) %>%
    mutate(color_ci = ifelse(upper_bound > 0 & lower_bound < 0, "red", "blu

  ci_plot <- ggplot(data = ci_plot_data, aes(x = as.factor(coeff_number), y
    geom_errorbar(aes(ymin = lower_bound, ymax = upper_bound),
                  color = ci_plot_data$color_ci, width = 0.3) +
    geom_point() + ggtitle("Robust Bayesian Estimator") +
    labs(x = "Coefficient", y = "w, w +/- 1.96 s.e.") + theme_excel()

  de_plot_data <- data.frame(observation_number = seq(1:length(rob_reg_resu
    bind_cols(data.frame(dev = rob_reg_results$dev)) %>%
    mutate(color_de = ifelse(dev > as.numeric(quantile(rob_reg_results$dev,
                             dev < as.numeric(quantile(rob_reg_results$dev,

  de_plot <- ggplot(data = de_plot_data, aes(x = observation_number, y = de
    geom_point(color = de_plot_data$color_de) +
    geom_hline(yintercept =  quantile(rob_reg_results$dev, c(0.005, 0.995))
    geom_hline(yintercept =  quantile(rob_reg_results$dev, c(0.005, 0.995))
    ggtitle("Robust Bayesian Estimator") +
    labs(x = "Observation", y = "Deviance Error") + theme_excel()

  tolerance <- 10 ^ -10
  conv_iter <- which((abs(diff(rob_reg_results$e_conv)) < tolerance &
                      abs(diff(rob_reg_results$q_conv)) < tolerance &
                      abs(diff(rob_reg_results$loglik_conv)) < tolerance) =

  conv_plot_data <- data.frame(iter_number = seq(1:conv_iter), e_conv = rob
```

```
                                  q_conv = rob_reg_results$q_conv[1:conv_iter]
                                  loglik_conv = rob_reg_results$loglik_conv[1:

  q_conv_plot <- ggplot(data = conv_plot_data, aes(x = iter_number, y = q_c
    geom_point() + ggtitle("q convergence") +
    labs(x = "iteration", y = "q") + theme_excel()

  loglike_conv_plot <- ggplot(data = conv_plot_data, aes(x = iter_number, y
    geom_point() + ggtitle("loglik convergence") +
    labs(x = "iteration", y = "loglik") + theme_excel()

  e_conv_plot <- ggplot(data = conv_plot_data, aes(x = iter_number, y = e_c
    geom_point() + ggtitle("RSS convergence") +
    labs(x = expression(nu), y = "RSS") + theme_excel()

  return(list(ci_plot = ci_plot, de_plot = de_plot, q_conv_plot = q_conv_pl
              loglike_conv_plot = loglike_conv_plot, e_conv_plot = e_conv_p
}

# robust regression graphics
rob_reg_graphics <- rob_reg_plot(rob_reg_results)
```

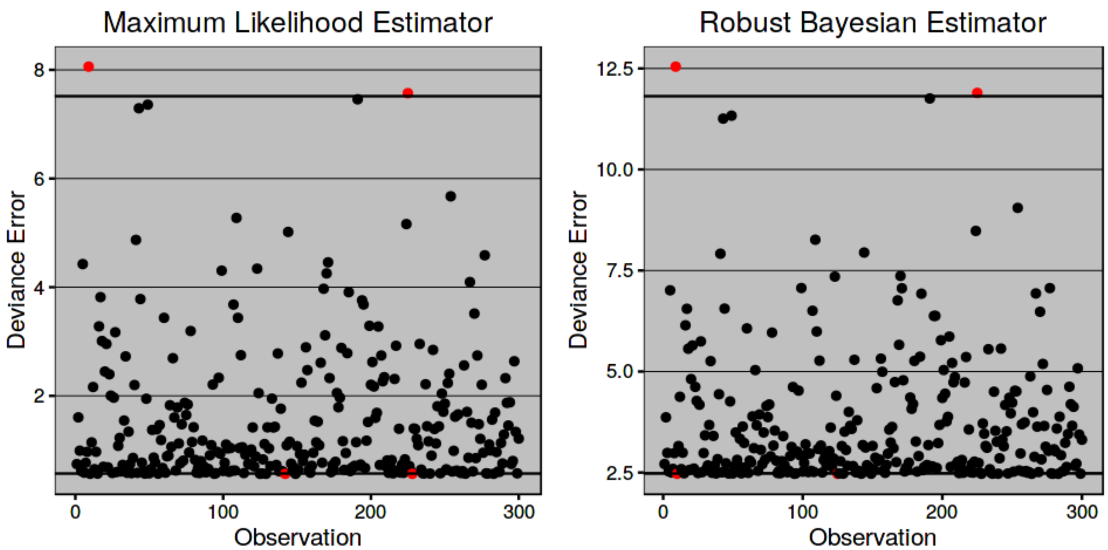
1)

```
In [6]: options(repr.plot.width=8, repr.plot.height=6)
        grid.arrange(mle_graphics$ci_plot, rob_reg_graphics$ci_plot, nrow = 2)
```



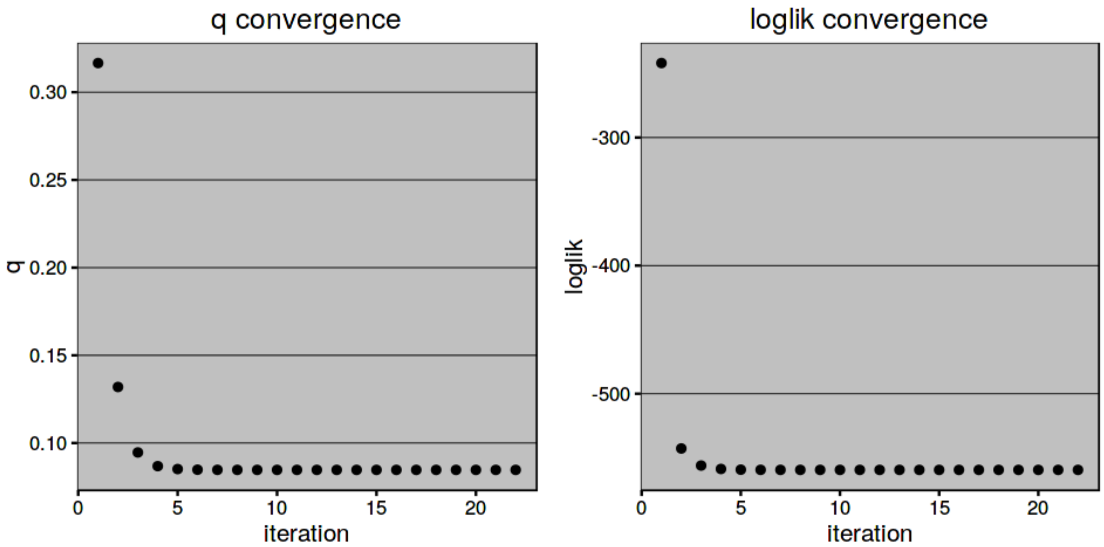
2)

```
In [8]: options(repr.plot.width=8, repr.plot.height=4)
        grid.arrange(mle_graphics$de_plot, rob_reg_graphics$de_plot, nrow = 1)
```



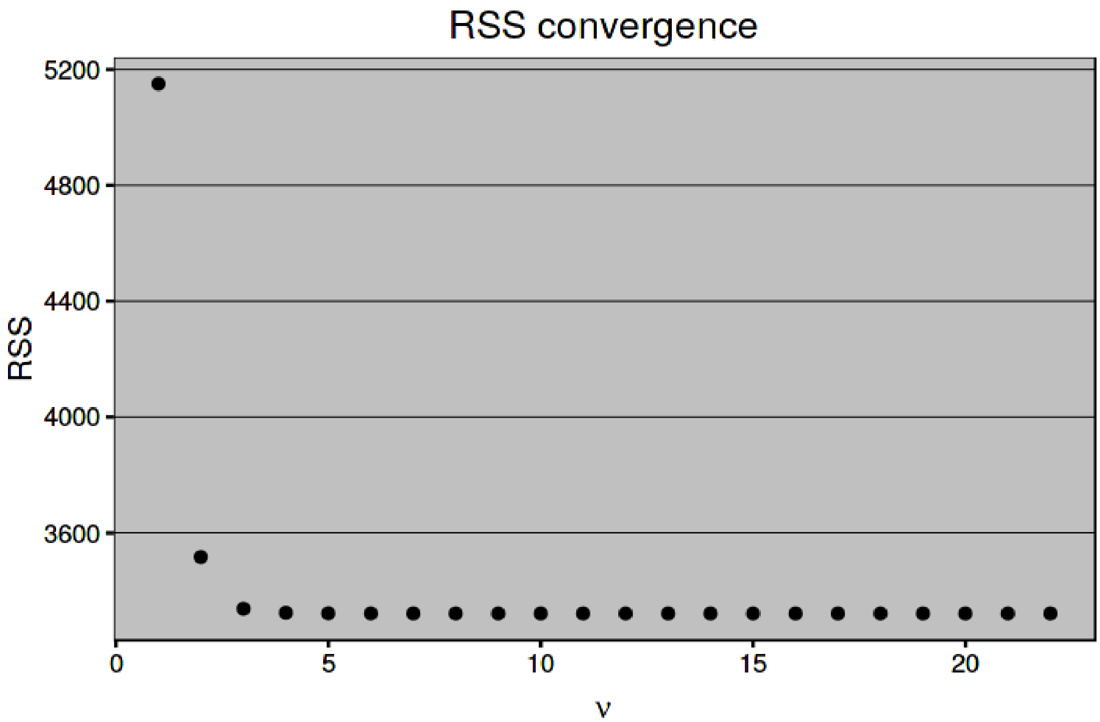
3) To decide the number of iterations for the algorithm, we could compare the values of q and the log likelihood after each iteration to those from the preceding iteration. Once these values are enough equal (taking a tolerance value) after each iteration then you can end the algorithm.

```
In [10]: options(repr.plot.width=8, repr.plot.height=4)
         grid.arrange(rob_reg_graphics$q_conv_plot, rob_reg_graphics$loglike_conv_p
```



4) To decide the degrees of freedom ν, iterate over the EM algorithm until the sum of squared errors has convergence. The graph below shows how the sum of squared errors converges as ν increase.

```
In [14]: options(repr.plot.width=6, repr.plot.height=4)
         rob_reg_graphics$e_conv_plo
```


RSS convergence
RSS
5200
4800
4400
4000
3600
0
5
10
15
20
ν