Lab 12 Solution

Problem 1

Find the MLE of (μ, σ^2) for the $\mathcal{N}(\mu, \sigma^2)$ distribution using Gradient Ascent algorithm. Compare with the MLEs.

Solution

```
# Generating sample data
set.seed(1)
x <- rnorm(100, mean = 5, sd = 2)  # true mu = 5, sigma = 2
```

The above code generates a random sample of size 100 from $\mathcal{N}(5, 4)$.

```
# Log-likelihood function
log_likelihood <- function(mu, sigma2, x) {
  n <- length(x)
  -n/2 * log(2*pi) - n/2 * log(sigma2) - sum((x - mu)^2) / (2 * sigma2)
}
```

The above snippet defines the log-likelihood for a random sample from $\mathcal{N}(\mu, \sigma^2)$.

```
# Gradient function
gradient <- function(mu, sigma2, x) {
  n <- length(x)
  dmu <- sum(x - mu) / sigma2
  dsigma2 <- -n / (2 * sigma2) + sum((x - mu)^2) / (2 * sigma2^2)
  return(c(dmu, dsigma2))
}
```

The above snippet defines the gradient ($\frac{\partial \ell}{\partial \mu}$ and $\frac{\partial \ell}{\partial \sigma^2}$) of the log-likelihood for a random sample from $\mathcal{N}(\mu, \sigma^2)$.

```
# Gradient Ascent function
gradient_ascent <- function(x, mu_init = 0, sigma2_init = 1,
                            lr = 0.01, iterations = 10000, tol = 1e-8) {
  mu <- mu_init
  sigma2 <- sigma2_init

  for (i in 1:iterations) {
    grad <- gradient(mu, sigma2, x)
    mu <- mu + lr * grad[1]
    sigma2 <- sigma2 + lr * grad[2]

    # prevent sigma2 from going negative
    if (sigma2 <= 0) sigma2 <- 1e-6

    # Stopping Condition
    if (sqrt(gradient(mu, sigma2, x)[1]^2 +
             gradient(mu, sigma2, x)[2]^2 < tol)){
      return(c(mu, sigma2))
    }
  }
  # Return the last value, if stopping condition is not satisfied
  return(c(mu, sigma2))
}
```

The above snippet defines the gradient ascent algorithm for finding the MLEs of $\mathcal{N}(\mu, \sigma^2)$.

```
# Run gradient ascent
mle_estimates <- gradient_ascent(x)
names(mle_estimates) <- c("mu", "sigma2")

# Analytical MLEs
mu_mle <- mean(x)
sigma2_mle <- var(x)

mle_estimates
```

```
      mu   sigma2
5.217775 3.194798
```

The above are the MLEs of the parameters μ and σ^2, using the Gradient Ascent method.

```r
# Comparing the results obtained

# Absolute Error
abs_error_mu <- abs(mle_estimates["mu"] - mu_mle)
abs_error_sigma2 <- abs(mle_estimates["sigma2"] - sigma2_mle)

# Relative Error
rel_error_mu <- abs_error_mu / abs(mu_mle)
rel_error_sigma2 <- abs_error_sigma2 / abs(sigma2_mle)

# Squared Error
squared_error_mu <- (mle_estimates["mu"] - mu_mle)^2
squared_error_sigma2 <- (mle_estimates["sigma2"] - sigma2_mle)^2

# Total report
comparison_error <- data.frame(
  row.names = c("Closed Form", "Gradient Ascent", "Absolute Error",
                "Relative Error", "Squared Error"),

  mu = c(mu_mle, mle_estimates["mu"], abs_error_mu,
         rel_error_mu, squared_error_mu),

  sigma2 = c(sigma2_mle, mle_estimates["sigma2"], abs_error_sigma2,
             rel_error_sigma2, squared_error_sigma2)
)

print(comparison_error)
```

```
                          mu      sigma2
Closed Form     5.217775e+00 3.227048359
Gradient Ascent 5.217775e+00 3.194797771
Absolute Error  8.881784e-16 0.032250588
Relative Error  1.702217e-16 0.009993835
Squared Error   7.888609e-31 0.001040100
```

From the above values of errors, we can see that the MLEs obtained by Gradient Ascent are quite close to that of the closed form MLEs of μ and σ^2.

Problem 2

Table 1: Data

Income	Age	y	Income	Age	y
45000	2	0	37000	5	1
40000	4	0	31000	7	1
60000	3	1	40000	4	1
50000	2	1	75000	2	0
55000	2	0	43000	9	1
50000	5	1	49000	2	0
35000	7	1	37500	4	1
65000	2	1	71000	1	0
53000	2	0	34000	5	0
48000	1	0	27000	6	0

Fit a logistic regression model for the above data and find the MLEs of coefficients of x_1 and x_2 using Gradient Ascent Algorithm.

Solution

```r
## Defining the vectors

# The covariate vectors
income <- c(45000, 40000, 60000, 50000, 55000, 50000, 35000,
            65000, 53000, 48000, 37000, 31000, 40000, 75000,
            43000, 49000, 37500, 71000, 34000, 27000)
age <- c(2, 4, 3, 2, 2, 5, 7, 2, 2, 1,
         5, 7, 4, 2, 9, 2, 4, 1, 5, 6)

# The response vector
y <- c(0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 0)
```

```r
# Feature scaling (Helpful for gradient ascent convergence)
x1 <- scale(income)
x2 <- scale(age)

# Combine into a matrix X with intercept
```

```
X <- cbind(1, x1, x2)  # Adding the intercept term
y <- as.numeric(y)
```

The above code scales the covariates, to bring them into a similar range and combines them into a matrix for future tasks.

The scaling of the covariates is done as below:

$$\text{scaled }_x = \frac{x - \bar{x}}{\sigma_x}$$

```
# Sigmoid function
sigmoid <- function(z) {
  1 / (1 + exp(-z))
}
```

The above code defines the sigmoid function, $\sigma(x)$, which will be used in finding

$$\begin{aligned} p_i &= \frac{e^{x_i^\top \beta}}{1 + e^{x_i^\top \beta}} \\ &= \frac{1}{1 + e^{-x_i^\top \beta}} \end{aligned}$$

```
# Log-likelihood function
log_likelihood <- function(X, y, beta) {
  z <- X %*% beta
  sum(y * z - log(1 + exp(z)))
}
```

The above snippet defines the log-likelihood, $\ell(\beta)$ of the given data

The likelihood is given by:

$$L(\beta|x) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}$$

Then, the log likelihood is given as:

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{n} [y_i \log(p_i) + (1-y_i)\log(1-p_i)] \\
&= \sum_{i=1}^{n} \left[y_i \log(\frac{p_i}{1-p_i}) + \log(1-p_i) \right] \\
&= \sum_{i=1}^{n} \left[y_i x_i^\top \beta + \ \log(\frac{1}{1+e^{x_i^\top \beta}}) \right] \\
&= \sum_{i=1}^{n} \left[y_i x_i^\top \beta \text{ - } \ \log(1+e^{x_i^\top \beta}) \right] \\
&= \sum_{i=1}^{n} [y_i z_i \text{ - } \ \log(1+e^{z_i})]
\end{aligned}$$

```
# Gradient of the log-likelihood
gradient <- function(X, y, beta) {
  p <- sigmoid(X %*% beta)
  t(X) %*% (y - p)
}
```

The above snippet defines the gradient, $\nabla_\beta \ell(\beta)$ of the log-likelihood function, $\ell(\beta)$ and is given as:

$$\begin{aligned}
\nabla_\beta \ell(\beta) &= \sum_{i=1}^{n} \left[y_i x_i - \frac{e^{x_i^\top \beta}}{1+e^{x_i^\top \beta}} x_i \right] \\
&= \sum_{i=1}^{n} [y_i x_i - p_i x_i] \\
&= \sum_{i=1}^{n} (y_i - p_i) x_i \\
&= X^\top (y-p)
\end{aligned}$$

```
# Gradient Ascent Algorithm
gradient_ascent <- function(X, y, lr = 0.01, iterations = 10000, tol = 1e-8) {
  beta <- matrix(0, ncol = 1, nrow = ncol(X))
  for (i in 1:iterations) {
    grad <- gradient(X, y, beta)
    beta <- beta + lr * grad
  }
  # Stopping Condition
```

```r
  if ((sqrt(gradient(X, y, beta)[1]^2 + gradient(X, y, beta)[2]^2 +
            gradient(X, y, beta)[3])^2) < tol){
    return(beta)
  }
  # Return the last value, if stopping condition is not satisfied
  return(beta)
}
```

The above snippet defines the gradient ascent algorithm for finding the MLEs of coefficients of x_1 and x_2.

```r
# Running gradient ascent
beta_hat <- gradient_ascent(X, y, lr = 0.1, iterations = 10000)

# Displaying estimated coefficients
names(beta_hat) <- c("Intercept", "Income", "Age")
beta_hat
```

```
          [,1]
[1,] 0.1472001
[2,] 0.9656863
[3,] 2.2176829
attr(,"names")
[1] "Intercept" "Income"    "Age"
```

The above are the MLE estimates of the coefficients of x_1 and x_2 for the given data using the Gradient Ascent algorithm.

```r
# Creating a data frame from given data
df <- data.frame(
  y = y,
  x1 = x1,
  x2 = x2
)

## Fitting a logistic regression model to the given data
## .. using pre- defined functions
model <- glm(y ~ x1 + x2, data = df, family = "binomial")

# Finding the coefficients for this method
coef(model)
```

```
(Intercept)          x1          x2
  0.1472001   0.9656863   2.2176829
```

The above snippet uses **Generalised Linear Model** to fit logistic regression to the given data and returns the intercept and coefficients of x_1 and x_2.

```r
# Comparing both results
beta_hat - coef(model)
```

```
             [,1]
[1,] 5.983825e-13
[2,] 1.664335e-12
[3,] 3.582468e-12
```

On comparing the differences between the coefficients obtained in both methods, we can see that the differences are quite close to 0 and hence the Gradient Ascent algorithm works very well for estimating the MLE estimates of the coefficients of x_1 and x_2.