MTH210: Lab 10

MM Algorithm

1. Run the code in `classDemonstration.R` which contains the MM-algorithm implementation for the location Cauchy example. Change the seed at the top, and rerun the code for different starting values.

2. **Ridge Regression:** Show that the ridge regression estimator in biased for β but has lower variance than the MLE estimator. (No need to use R for this).

3. **Bridge Regression**: In this problem you will write a function to calculate the Bridge Regression estimate using MM algorithm for any given data set.

   Recall that the Bridge objective function to minimize is

   $$Q_B(\beta) = \frac{(y - X\beta)^T(y - X\beta)}{2} + \frac{\lambda}{2\alpha}\sum_{j=1}^{p}|\beta_j|^\alpha$$

   We found the minorizing function to be

   $$\tilde{f}(\beta|\beta_{(k)}) = \text{constants} + \frac{(y - X\beta)^T(y - X\beta)}{2} + \frac{\lambda}{2\alpha}\sum_{j=1}^{p} m_{j,(k)}\beta_j^2$$

   where $m_{j,(k)} = \alpha|\beta_{j,(k)}|^{\alpha-2}$. Here β_j and $\beta_{j,(k)}$ denoted the jth component of β and $\beta_{(k)}$ respectively. The maximizer of this minorizing function gives us the next iterate of the MM algorithm, which is

   $$\beta_{(k+1)} = \left(X^TX + \frac{\lambda}{\alpha}M_{(k)}\right)^{-1} X^Ty\,,$$

   where $M_{(k)}$ is the diagonal matrix of $m_{j,(k)}$s.

   Now, you will write a function that takes data y, X as input, along with values of λ and α, and returns the Bridge regression estimate. The function should look like:

   ```
   # y = response
   # X = covariate matrix
   # lambda = penlaty term
   # alpha = bridge term
   ```

```
# max.ter = maximum iterations for the MM algorithm.
#         if max iteration has been reached, the function should print: "Maximum iterations reac
# tol = tolerance level for when to stop MM
bridgeReg <- function(y, X, alpha, lambda, max.iter, tol)
{
  ...

  while(... < tol)
  {
    if(iter > max.iter)
    {
      print("Maximum iterations reached")
      stop
    }
    # MM steps
  }

  return(...)
}
```

4. Simulate the following data:

```
set.seed(1)
n <- 100
p <- 5
beta.star <- c(3, 1, .01, -2, -.003)

# X matrix with intercept
X <- cbind(1, matrix(rnorm(n*(p-1)), nrow = n, ncol = (p-1)))

# generating response
y <- X %*% beta.star + rnorm(n)
```

Call your `bridgeReg()` function from the previous question on this dataset for the following values of α and λ, and the print the estimate. See if the estimate is close to the true value of β from which the data was simulated.

```
alpha.vec <- seq(1, 2, length = 5)
lambda <- c(.01, .1, 1, 10, 100)
...
for(i in 1:length(alpha.vec))
{
  for(j in 1:length(lambda))
```

```
  {
    bridge.est <- bridgeReg(......)
    print(bridge.est)
  }
}
```

5. In the previous question, suppose we fix $\lambda = 10$ and for different values of α, we obtain a different estimation of bridge regression coefficient: $\hat{\beta}_{\alpha,\lambda}$. The distance of this estimate from the true β can be measured with $\|\hat{\beta}_{\alpha,\lambda} - \beta\|$.

```
for(i in 1:length(alpha.vec))
{
    bridge.est <- bridgeReg(......)
    norm(bridge.est - beta.star, "2")
}
```

This is a random quantity (since $\hat{\beta}_{\alpha,\lambda}$ is random). Suppose I want to find the average of expected distance of $\hat{\beta}_{\alpha,\lambda}$ from β.

$$E\left[\|\hat{\beta}_{\alpha,\lambda} - \beta\|\right]$$

The expectation is with respect to the distribution of $\hat{\beta}_{\alpha,\lambda}$, which we don't know. But since this is a simulated data setup, we can simulate multiple such datasets (keeping the same value of β), and obtain different realizations of $\hat{\beta}_{\alpha,\lambda}$. Taking an average of all the resulting distances will give us an estimate of the above expectation. Implement this exercise and see which value of α gives the lowest expected distance. Below is code that might help you.

```
alpha.vec <- seq(1, 2, length = 5)
reps <- 100
dist <- matrix(0, nrow = reps, ncol = length(alpha.vec))
for(r in 1:reps)
{
  # generate X and y again here
  for(i in 1:length(alpha.vec))
  {
    bridge.est <- bridgeReg(......)
    dist[r, i] <- norm(bridge.est - beta.star, "2")
  }
}
```