MTH 511a - 2020: Lecture 17

Instructor: Dootika Vats

The instructor of this course owns the copyright of all the course materials. This lecture material was distributed only to the students attending the course MTH511a: "Statistical Simulation and Data Analysis" of IIT Kanpur, and should not be distributed in print or through electronic media without the consent of the instructor. Students can make their own copies of the course materials for their use.

1 Numerical optimization methods

The two optimization techniques we've learned have utilized derivatives. But what if the objective function is not differential? Or the derivatives are too complicated to write down explicitly. What do we do then?

1.1 MM Algorithm

Consider obtaining a solution to

$$\theta_* = \arg\max_{\theta} f(\theta)$$

The "Minorize/Maximize algorithm" algorithm at a current iterate, finds a "minorizing" function at that point, and then maximizes that minorizing function. That is, at any given iteration, consider a *minorizing function* $\tilde{f}(\theta|\theta_{(k)})$ such that:

- $f(\theta_k) = \tilde{f}(\theta_k|\theta_k)$
- $f(\theta) \geq \tilde{f}(\theta|\theta_k)$ for all other θ

Then, $\theta_{(k+1)}$ is obtained as

$$\theta_{(k+1)} = \arg\max_{\theta} f_k(\theta|\theta_{(k)})\,.$$

The algorithm has the ascent property in that every update increases the objective value. That is

$$\begin{aligned} f(\theta_{(k+1)}) &\geq \tilde{f}(\theta_{(k+1)} \mid \theta_{(k)}) \\ &\geq \tilde{f}(\theta_{(k)} \mid \theta_{(k)}) \\ &= f(\theta_{(k)})\,. \end{aligned}$$

When minimizing an objective function, we the opposite: we find a majorizing function and then minimize it.

1.1.1 Bridge Regression (including Lasso)

Example 1 (Bridge regression). Recall the case of the penalized (negative) log-likelihood problem during ridge regression. Ridge regression can be generalized as *bridge regression* when the objective function is

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

for $\alpha \in [1, 2]$ and $\lambda > 0$. When $\alpha = 2$, this is ridge regression, and when $\alpha = 1$, this is lasso regression. Different choices of α, lead to different style of penalization. For a given λ, smaller values of α push the estimates closer towards zero. Specifically, Lasso ($\alpha = 1$) is quite popular.

We need to find $\arg\min Q_B(\beta)$. First note that, $(y-X\beta)^T(y-X\beta)$ is a convex function and $|\beta_i|^\alpha$ is convex for $\alpha \geq 1$. So the objective function is convex, thus a global solution exists.

Note that for $\alpha = 1$, the objective function is not differentiable at 0, and for $\alpha \in (1, 2)$, the function is not twice differentiable at 0. Thus, using Newton-Raphson and gradient descent is not possible. We will instead use an MM algorithm. Since this is a minimization problem, we will find a *majorizing function*.

We will try to find a majorizing function that upper bounds the objective $Q_B(\beta)$, and then minimize the majorizing function. Optimizing the majorizing function will again require derivatives, so we want to get rid of the absolute value.

Consider a function $h(u) = u^{\alpha/2}$ for $u \geq 0$ and see that

$$h'(u) = \frac{\alpha}{2}u^{\alpha/2-1}$$

and

$$h''(u) = \frac{\alpha}{2}\left(\frac{\alpha}{2} - 1\right)u^{\alpha/2-2} \leq 0$$

so $h(u)$ is a concave function. For a concave function, by the "Rooftop theorem"the first order Taylor series creates a tangent line that is above the function. Thus for a u^*,

$$h(u) \leq h(u^*) + \frac{\alpha}{2}(u^*)^{\alpha/2-1}(u - u^*)\,.$$

For any given iteration of the optimization given $\beta_{(k)}$, taking $u = |\beta_i|^2$ and $u^* = |\beta_{i,(k)}|^2$ where β_i is the ith component of the vector β. Then,

$$\begin{aligned}
|\beta_i|^\alpha &\leq |\beta_{i,(k)}|^\alpha + \frac{\alpha}{2}|\beta_{i,(k)}|^{\alpha-2}\left(\beta_i^2 - \beta_{i,(k)}^2\right) \\
&= |\beta_{i,(k)}|^\alpha - \frac{\alpha}{2}|\beta_{i,(k)}|^\alpha + \frac{\alpha}{2}|\beta_{i,(k)}|^{\alpha-2}\beta_i^2 \\
&= \text{constants} + \frac{m_j}{2}\beta_j^2
\end{aligned}$$

where $m_j = \alpha|\beta_{i,(k)}|^{\alpha-2}$. You will see that the constants will not be important.

Now that we have bounded the the penalty function, we have an upper bound on the full objective function!

So, the objective function can be bounded above by:

$$Q_B(\beta) \leq \text{constants} + \frac{(y - X\beta)^T(y - X\beta)}{2} + \frac{\lambda}{2\alpha}\sum_{j=1}^{p} m_j\beta_j^2 \,.$$

Why is this upper bound useful?

- Remember that at any given iteration, the optimization is with respect to β. Thus, the constants are truly constants.
- The upper bound has no absolute values!
- Recall that we obtained the upper bound function using a derivative of $h(u)$. This derivative is not defined at $u = 0$. However, we're only using the derivative function at $u^* = |\beta_{i,(k)}|^2$, which is the previous iteration. So as long as we DO NOT START at zero, this upper bound is valid.
- Finally, the upper bound is easily optimizable, as it is similar to ridge. (See below)

The objective function is similar to ridge regression, except it is "weighted". Following the same steps as in ridge optimization, you can show that the minimum occurs at

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

where $M_{(k)} = \text{diag}(m_1, m_2, \ldots, m_p)$. Note that here $M_{(k)}$ is what drives the direction of the optimization.

```
################################################
## Bridge regression and the MM algorithm
## Compare for different values of alpha
################################################
set.seed(1)
n <- 100
p <- 5
```

```r
beta.star <- c(0,0,0,rnorm(p-3, sd = 1)) # larger variance than exercise 4.
beta.star # to output
# [1]  0.0000000 0.0000000 0.0000000 -0.6264538 0.1836433

# Making design matrix, first column is 1
X <- cbind(1, matrix(rnorm(n*(p-1)), nrow = n, ncol = (p-1)))

# Generating response
y <-  X %*% beta.star + rnorm(n, mean = 0, sd = 1)

###############
# First MLE
###############
# MLE of beta
beta.mle <- solve( t(X) %*%X ) %*% t(X) %*%y

# Bridge solutions
alpha.vec <- c(1, 1.5, 1.8)

lambda <- 10
beta.bridge <- matrix(0, nrow = p, ncol = 3)
tol <- 1e-5
for(i in 1:length(alpha.vec) ) # loop for each alpha
{
  current <- solve( t(X) %*%X + diag(lambda,p) ) %*% t(X) %*%y # start at
     ridge solution
  iter <- 0
  diff <- 100
  while( (diff > tol) && iter < 1000)
  {
    iter <- iter + 1

    # M matrix diagonals
    ms <- as.vector( lambda/2 *( abs(current))^(alpha.vec[i] - 2) )

    # MM update -- using qr.solve for numerical stability
    update <- qr.solve(t(X) %*% X + diag(ms, p)) %*% t(X) %*% y

    diff <- sum( (current - update)^2 )
    current <- update
  }
  beta.bridge[,i] <- current
}

## Comparing MLE and Bridge for different alpha values
(beta.ridge <- solve( t(X) %*%X + diag(lambda,p) ) %*% t(X) %*%y)
```

```
cbind(beta.mle, beta.bridge, beta.ridge)
#           [,1]       [,2]       [,3]       [,4]       [,5]
#[1,] -0.09512113 -0.0359535 -0.0760993 -0.08483116 -0.08223668
#[2,]  0.20373809 0.1302566 0.1732737 0.18477742 0.17764152
#[3,]  0.16523932 0.1144513 0.1467874 0.15499924 0.15213858
#[4,] -0.62238012 -0.5699536 -0.5834113 -0.58933128 -0.56569626
#[5,]  0.23165539 0.1675901 0.2017733 0.21170945 0.20262618

beta.star
#[1]  0.0000000 0.0000000 0.0000000 -0.6264538 0.1836433
```

2 Questions to think about

- How do you think we can choose α in any given problem?
- Can you try the MM algorithm for the Location-Cauchy example?
- Will the MM algorithm always converge to a global maxima?