Stat 610 Lab 5/Homework 8

Due Wednesday, November 12, 11:59pm

Assignment

Part 1: Problems with Newton's method

The Cauchy location family with location family θ has probability density function

$$f(x;\theta) = \frac{1}{\pi}\left[\frac{1}{1+(x-\theta)^2}\right]$$

Its log likelihood is therefore

$$\ell(\theta) = \sum_{i=1}^{n}(-\log(1+(x_i-\theta)^2) - \log(\pi))$$

In this lab you'll look at ways of estimating θ.

You may recall from your other statistics classes that the Cauchy distribution is very poorly behaved and so many of the normal ways of estimating parameters do not work well.

We will be using Newton's method here, and so we will need to know the first and second derivatives of ℓ with respect to θ. You can verify for yourself that these are:

$$\ell'(\theta) = \sum_{i=1}^{n}\frac{2(x_i-\theta)}{1+(x_i-\theta)^2}$$
$$\ell''(\theta) = \sum_{i=1}^{n}\frac{-2+2(x_i-\theta)^2}{(1+(x_i-\theta)^2)^2}$$

Some R functions that compute these quantities are given below:

```
ell <- function(x, theta) {
    return(sum(log(pi^(-1) * 1 / (1 + (x - theta)^2))))
}
ellprime <- function(x, theta) {
    num = 2 * (x - theta)
    denom = 1 + (x - theta)^2
    return(sum(num / denom))
}
elldoubleprime <- function(x, theta) {
    num = 2 * (x - theta)^2 - 2
    denom = (1 + (x - theta)^2)^2
    return(sum(num / denom))
}
```

1. Suppose you are given the following ten data points:
   $-2.09, -2.68, -1.92, -1.76, -2.12, 2.21, 1.97, 1.61, 1.99, 2.18$. Compute the log likelihood of this dataset for a range of values of θ between -4 and 4. Make a plot of the log likelihood as a function of θ. Are there going to be any problems using Newton's method in this model?

2. Write a function that takes one Newton step. Your function should take a data vector and a value for θ.

3. Use the function that you wrote in the previous part to experiment with starting at different values of θ and taking just a couple (maybe around 10) Newton steps. You can either print out the updated value of $\hat{\theta}$ after each step, or you can add them to the plot you made in question 1 using `abline(v = thetahat)` (this only works if you used base R to plot in question 1).

4. When we implement Newton's method, we usually take Newton steps until either the difference between the current estimate and the previous estimate is very small (we have "converged") or until we have reached some maximum number of steps. Write such a function for estimating θ in the Cauchy location family.

   Try your function out on the values given in question 1. Does it give good results? Is there any extra information you could give a user that would tell them whether the value returned is a local maximum, a local minimum, or something else?

Part 2: Gradient descent

In this part, you will implement gradient descent for logistic regression. Recall that in logistic regression, we are trying to find the coefficient vector β that maximizes the log likelihood of the data, or, equivalently, minimizes the negative log likelihood of the data. If $\mathbf{x}_i, i = 1, \ldots, n$ are p-vectors giving the values of p predictor variables for the ith sample and $y_i, i = 1, \ldots, n$ are 0/1-valued response variables, then the negative log likelihood for the logistic regression model is

$$f(\beta) = -\ell(\beta) = \sum_{i=1}^{n} -[y_i \log(p_i(\beta)) + (1 - y_i) \log(1 - p_i(\beta))]$$

where

$$p_i(\beta) = \frac{\exp(\mathbf{x}_i^T \beta)}{1 + \exp(\mathbf{x}_i^T \beta)}$$

The gradient is then

$$\frac{d}{d\beta} f(\beta) = -\sum_{i=1}^{n} (y_i - p_i(\beta)) \mathbf{x}_i = -X^T (y - p)$$

if X is the $n \times p$ matrix of predictors, and y and p are n-vectors giving the y_i's and $p_i(\beta)$'s, respectively. In standard gradient descent with a fixed step size η we would start with some

guess $\beta^{(0)}$ for the optimal value of β, and then iterate

$$\beta^{(t+1)} = \beta^{(t)} - \eta \frac{d}{d\beta} f(\beta^{(t)})$$

until the gradient is sufficiently small.

You should do the following:

1. Generate data from a logistic regression model like we did in class (https://jfukuyama.github.io/teaching/stat610/notes/lecture19.html#(25)), but with a larger number of points, say, $n = 10^4$.
2. Create a function that takes the predictor matrix `X`, the response vector `y`, and the current value of `beta` and computes the gradient of the negative log likelihood at `beta`.
3. Create a function that performs gradient descent. It should take `X`, `y`, a starting value `beta`, a step size `eta`, and the number of iterations to perform (for the purposes of this assignment, we aren't going to check for convergence, just do a fixed number of iterations).
4. Try your function out with the data you generated and `eta` set to a very small value (e.g. .001). Gradient descent with a fixed step size will only converge if the step size is small enough. If you pick `eta` too big the algorithm will diverge.