Review Session 7

Jingyu He

2/18/2017

Gradient Descent

In many parametric statistics problems, we want to estimate the "best" parameters to fit data. Usually we write down a loss (or a likelihood) which are a function of data and parameters, then optimize the function with respect to parameters. Many statistical estimation problems can be summarized to an optimization problem in this way. Gradient descent is one of the most frequent used optimization methods. Suppose $f(\mathbf{b})$ is the function to minimize, we caculate the gredient $\nabla f(\mathbf{b})$, then update the coefficients in the opposite direction.

$$\mathbf{b}_{i+1} = \mathbf{b}_i - \alpha \nabla f(\mathbf{b}_i)$$

α is the learning rate, which controls the step length. We repeat this iteration step untill the norm of $\nabla f(\mathbf{b}_i)$ is smaller than a threshold. We also specify maximum number of iterations to avoid infinity loop. If the iteration counts exceeds the maximum, the algorithm stops no matter converge or not.

Three parameters need to be specified for gradient descent:

1. Learning rate α. This is a critical parameter. If α is too large, the algorithm jumps around the optimal but never converge. On the contrary, if α is too small, it takes a long time to converge.
2. Threshold ε and max number of iteration m. The algorithm will stop if the Eculidean norm of $\nabla l(\mathbf{b}_m)$ is smaller than ε or we reach the maximum number of iterations.

We also need to specifiy two functions:

1. $l(\mathbf{b})$, object function $l(\mathbf{b})$ to be **minimized**. If you want to maximize a function, just simply add a negative sign to the function.
2. $\nabla l(\mathbf{b})$, gradient of the object function with respect to parameters.

SGD for logistic regression

To do gradient descent, we need to specify the object function to minimize. In the setting of logistic regression, we have a response $\mathbf{y}$ that is binary, $\mathbf{y} = (y_1, \cdots, y_n)^T$ with $y_i \in \{0, 1\}$. $\mathbf{X}$ denotes a set of covariates, a $n \times p$ matrix with each row $\mathbf{x}_i$. We want to estimate a regression coefficient vector $\mathbf{b} \in \mathbb{R}^p$. In the logistic model, we assume the probability of $y_i = 1$ given $\mathbf{Xb}$ is

$$P(y_i = 1 \mid \mathbf{Xb}) = \frac{\exp(\mathbf{x}_i^T \mathbf{b})}{1 + \exp(\mathbf{x}_i^T \mathbf{b})}$$

We estimate the regression coefficient $\mathbf{b}$ by maximizing the likelihood function. Recall the distribution function of binomial distribution, the likelihood is

$$L(\mathbf{b}) = \prod_{i=1}^{n} P(y_i = 1 \mid \mathbf{Xb})^{y_i} (1 - P(y_i = 1 \mid \mathbf{Xb}))^{1-y_i}$$

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood. Take logarithm and add a negative sign to the likelihood above, we get

$$l(\mathbf{b}) = \sum_{i=1}^{n} \left(-y_i \mathbf{x}_i^t \mathbf{b} + \log \left[1 + \exp \left(\mathbf{x}_i^t \mathbf{b} \right) \right] \right)$$

Take derivative with respect to b_j, which is the j-th entry of $\mathbf{b}$

$$\frac{l(\mathbf{b})}{\partial b_j} = \sum_{i=1}^{n} -y_i(\mathbf{x}_i)_j + \frac{(\mathbf{x}_i)_j}{1+\exp\left(-\mathbf{x}_i^t\mathbf{b}\right)} = \sum_{i=1}^{n} \left(p_i - y_i\right)(\mathbf{x}_i)_j$$

where $(\mathbf{x}_i)_j$ is the j-th entry of $\mathbf{x}_i$ and $p_i = \frac{1}{1+\exp(-\mathbf{x}_i^t\mathbf{b})}$. Write everything in matrix multipication form, let $\mathbf{p} = (p_1 \quad p_2 \cdots p_n)$.

$$\frac{l(\mathbf{b})}{\partial b_j} = -\mathbf{x}_i^t(\mathbf{y}-\mathbf{p})$$

The gradient of $l(\mathbf{b})$ is

$$\nabla l(\mathbf{b}) = \begin{pmatrix} \frac{l(\mathbf{b})}{\partial b_1} \\ \vdots \\ \frac{l(\mathbf{b})}{\partial b_n} \end{pmatrix} = \begin{pmatrix} -\mathbf{x}_1^t(\mathbf{y}-\mathbf{p}) \\ \vdots \\ -\mathbf{x}_n^t(\mathbf{y}-\mathbf{p}) \end{pmatrix} = -\mathbf{X}^t(\mathbf{y}-\mathbf{p})$$

Then we update $\mathbf{b}$ by

$$\mathbf{b}_{m+1} = \mathbf{b}_m - \alpha\nabla l(\mathbf{b}_m)$$

where m denotes the m-th iteration. We repeat this updating procedure several times until convergence, that is the Eculidean norm of $\nabla l(\mathbf{b}_m)$ is sufficiently close to zero (smaller than a pre-setting threshold).

Code

First we simulate data

```
# generate data
set.seed(12345)
n <- 100
p <- 10
X <- matrix(rnorm(n*p),n,p)
b <- matrix(rnorm(p),p,1)
e <- 0.5*matrix(rnorm(n),n,1)
z <- X%*%b + e
y <- as.vector((plogis(z) <= runif(n)) + 0)
```

Object and gradient functions

```
# The object function
l <- function(X,y,b) {
  -t(y)%*%(X%*%b) + sum(log(1+exp(X%*%b)))
}

# Define the gradient function
grad_l <- function(X,y,b){
  -t(X)%*%(y-plogis(X%*%b))
}
```

We use the **gdescent** function in pacakge **gettothebottom**.

```
# If you don't have this pacakge on your computer
# Install it !
# install.packages("gettingtothebottom")
library(gettingtothebottom)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: grid
```

```
## Loading required package: Matrix
```

Run gradient descent.

```
# Choose alpha, try more alpha by yourself!
alpha = 0.1
logistic_ex <- gdescent(l,grad_l,X,y,alpha=alpha,iter=15000)
```

```
## Minimum function value:
##  22.33875
##
## Intercept:
##  -0.6917839
##
## Coefficient(s):
##  -3.06784109 -0.67155928 1.97323830 1.36163002 0.42670615 3.77991403 0.04289983 2.29479711 -0.011411
```

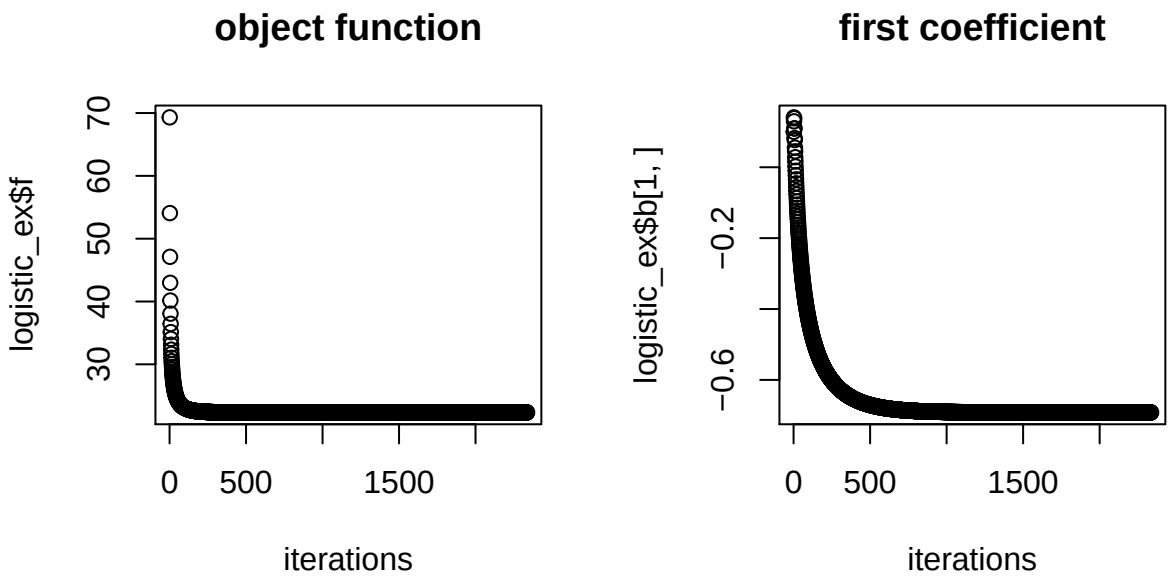
Compare our result with the R built-in function glm.

```
# compare of R built-in function glm
summary(glm(y ~ X, family = binomial))$coef[,1]
```

```
## (Intercept)          X1          X2          X3          X4          X5
## -0.69178496 -3.06784406 -0.67156012  1.97324011  1.36163106  0.42670634
##          X6          X7          X8          X9         X10
##  3.77991811  0.04289958  2.29479965 -0.01141058 -0.60526117
```

Draw plots of the object function and one coefficient

```
par(mfrow = c(1,2))
plot(logistic_ex$f, main = "object function", xlab ="iterations")
plot(logistic_ex$b[1,], main = "first coefficient", xlab = "iterations")
```

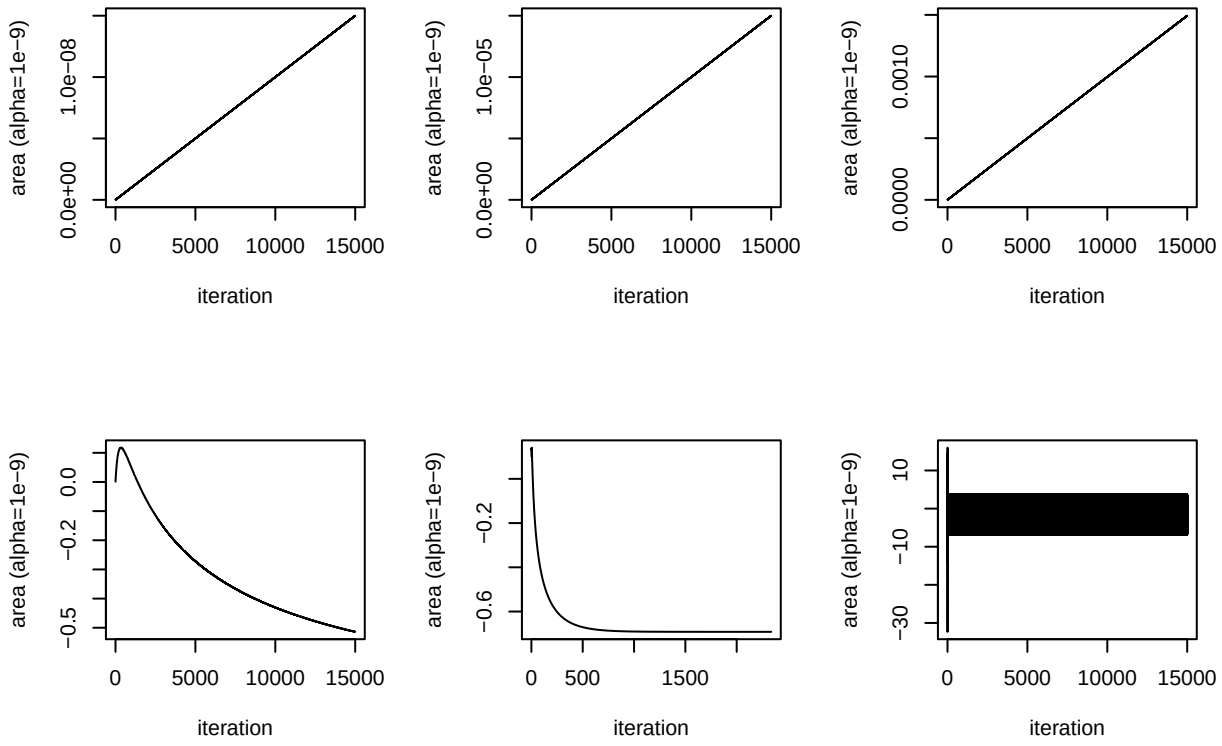


It seems the algorithm converges well from the plots above.

In the previous example, we happened to choose a good α, how about other choices of α?

```r
# Look at output for various different alpha values
vary.alpha <- lapply(c(1e-12, 1e-9, 1e-7, 1e-3, 0.1, 0.9),
                     function(alpha) gdescent(l, grad_l, X, y, alpha=alpha, iter = 15000))

par(mfrow = c(2, 3))
for (j in 1:6) {
    plot(vary.alpha[[j]]$b[1,], ylab="area (alpha=1e-9)", xlab="iteration", type="l")
}
```



The choice of α is critical!