Gradient Steepest Descent

YIK LUN, KEI

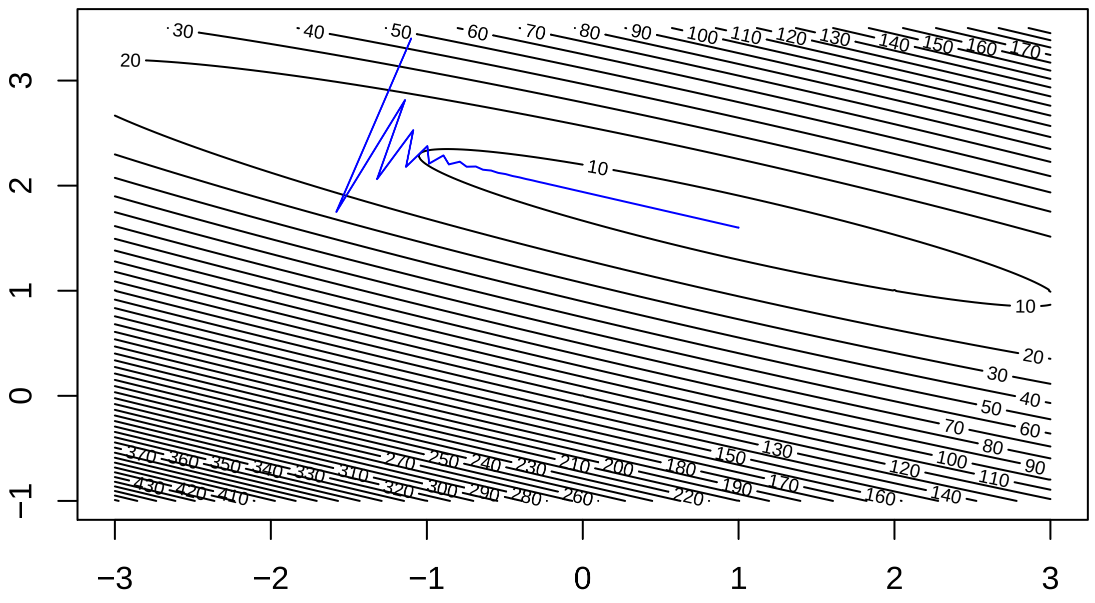
Minimize SSR

```
error<-0
points<-matrix(c(1,2,3,4,2,6,4,8),ncol=2)
toterror<-function(points,b0,b1){
  for(i in 1:dim(points)[1]){
    error <- error + (points[i,2] - (b0+b1*points[i,1]))^2
  }
  error
}

b0_grad<-function(points,b0,b1){
  grad = 0
  for(i in 1:dim(points)[1]){
    grad = grad + -2 * (points[i,2]-(b0+b1*points[i,1]))
  }
  grad
}

b1_grad<-function(points,b0,b1){
  grad = 0
  for(i in 1:dim(points)[1]){
    grad = grad + -2 * points[i,1]*(points[i,2]-(b0+b1*points[i,1]))
  }
  grad
}

b0=-2;b1=1;alpha=0.025
b0hist<-rep(0,1000)
b1hist<-rep(0,1000)

for(i in 1:1000){
  b0_step <- alpha * b0_grad(points,b0,b1)
  b1_step <- alpha * b1_grad(points,b0,b1)
  b0 = b0 - b0_step# update b0 and b1 after both stepings
  b1 = b1 - b1_step
  b0hist[i] <- b0
  b1hist[i] <- b1
}
b0;b1
```

```
## [1] 1
```

```
## [1] 1.6
```

```r
bhist<-cbind(b0hist,b1hist)
b0m<-seq(-3,3, by=0.01)
b1m<-seq(-1,3.5, by=0.01)
z <- outer(b0m,b1m,FUN="toterror",points=points)
contour(b0m,b1m,z,nlevels = 40)
points(bhist,type="l",col="blue")
```



Matrix Form

```r
x0 <- rep(1,4)
x1 <- 1:4
x <- cbind(x0,x1)
y <- as.matrix(c(2,6,4,8))
theta <- matrix(c(-2,1))
thetahist <- matrix(NA,nrow=1000,ncol=2)
costhist <- matrix(NA, nrow=1000,ncol=1)
alpha = 0.025
h<-function(x,theta) x %*% theta # fitted y
cost <- function(x,y,theta){sum((x %*% theta - y)^2)}
grad<-function(x,y,theta) 2 * t(x) %*% (x %*% theta - y)

solve(t(x) %*% x) %*% t(x) %*% y;lm(y~x + 0)
```

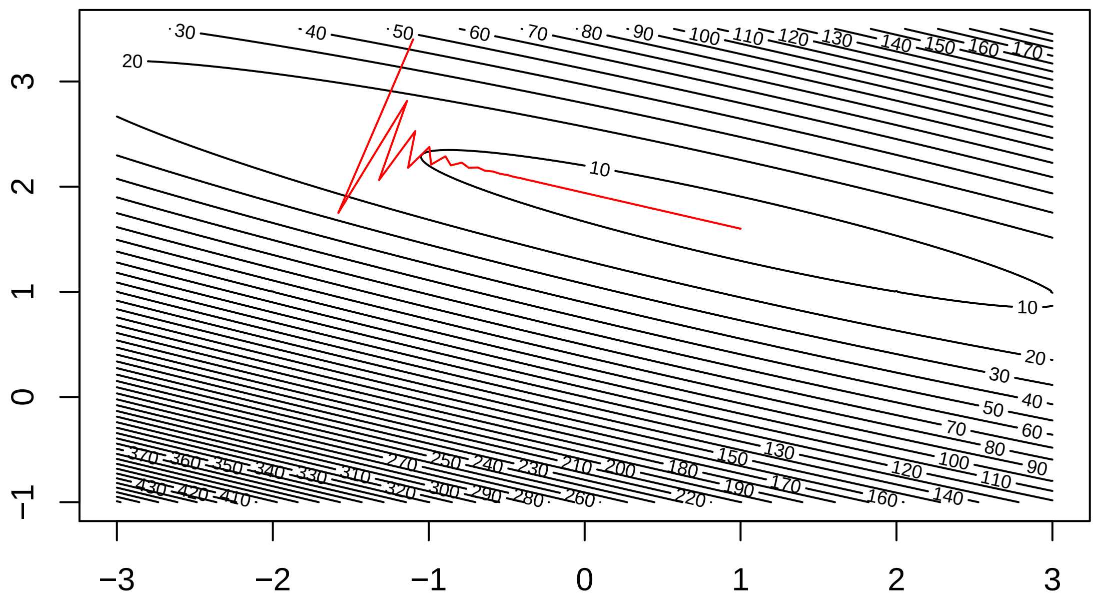
```
##     [,1]
## x0   1.0
## x1   1.6
```

```
##
## Call:
## lm(formula = y ~ x + 0)
##
## Coefficients:
## xx0  xx1
## 1.0  1.6
```

```
for(i in 1:1000){
  theta <- theta - alpha * grad(x,y,theta)
  thetahist[i,] <- t(theta)
  costhist[i,] <- cost(x,y,theta)
}

theta
```

```
##     [,1]
## x0   1.0
## x1   1.6
```

```
b0m<-seq(-3,3, by=0.01)
b1m<-seq(-1,3.5, by=0.01)
z <- outer(b0m,b1m,FUN="toterror",points=points)
contour(b0m,b1m,z,nlevels = 40)
points(thetahist,type="l",col="red")
```



Standardized data makes convergence faster

```
x.scaled<-x
x.scaled[,2] <- (x[,2] - mean(x[,2])) / sd(x[,2])
solve(t(x.scaled) %*% x.scaled) %*% t(x.scaled) %*% y;lm(y~x.scaled[,2])
```

```
##        [,1]
## x0 5.000000
## x1 2.065591
```

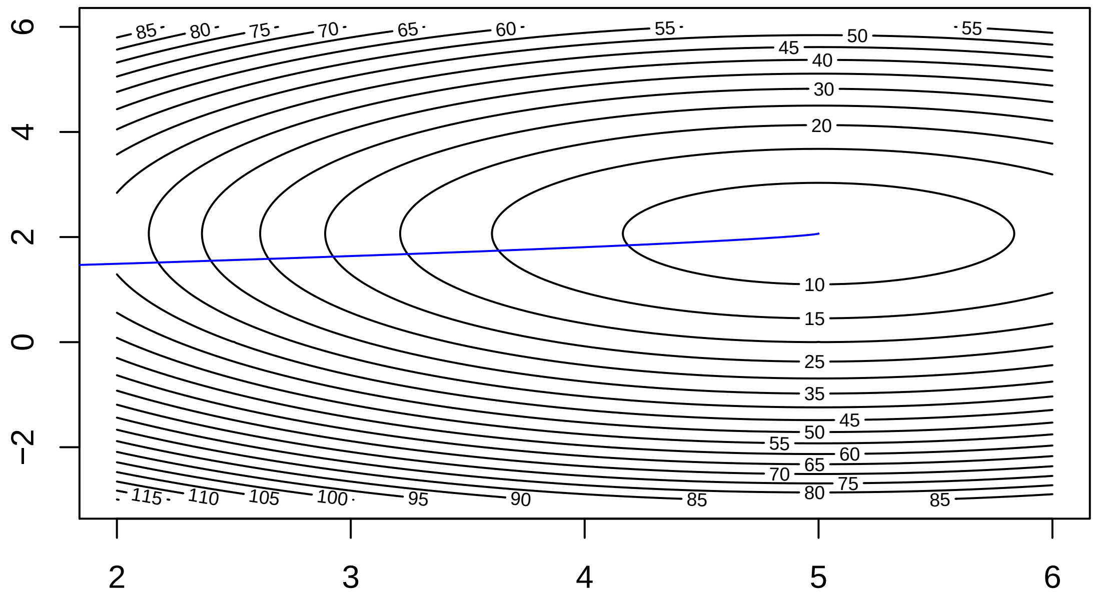
```
##
## Call:
## lm(formula = y ~ x.scaled[, 2])
##
## Coefficients:
##   (Intercept)  x.scaled[, 2]
##         5.000          2.066
```

```
theta <- matrix(c(-2,1))
thetahist <- matrix(NA,nrow=200,ncol=2)
costhist.scaled <- matrix(NA,nrow=200,ncol=1)
alpha = 0.025
```

```r
h<-function(x,theta) x %*% theta # fitted y
grad<-function(x,y,theta) 2 * t(x) %*% (x %*% theta - y)


for(i in 1:200){
  theta <- theta - alpha * grad(x.scaled,y,theta)
  thetahist[i,] <- t(theta)
  costhist.scaled[i,] <- cost(x.scaled,y,theta)
}

theta
```

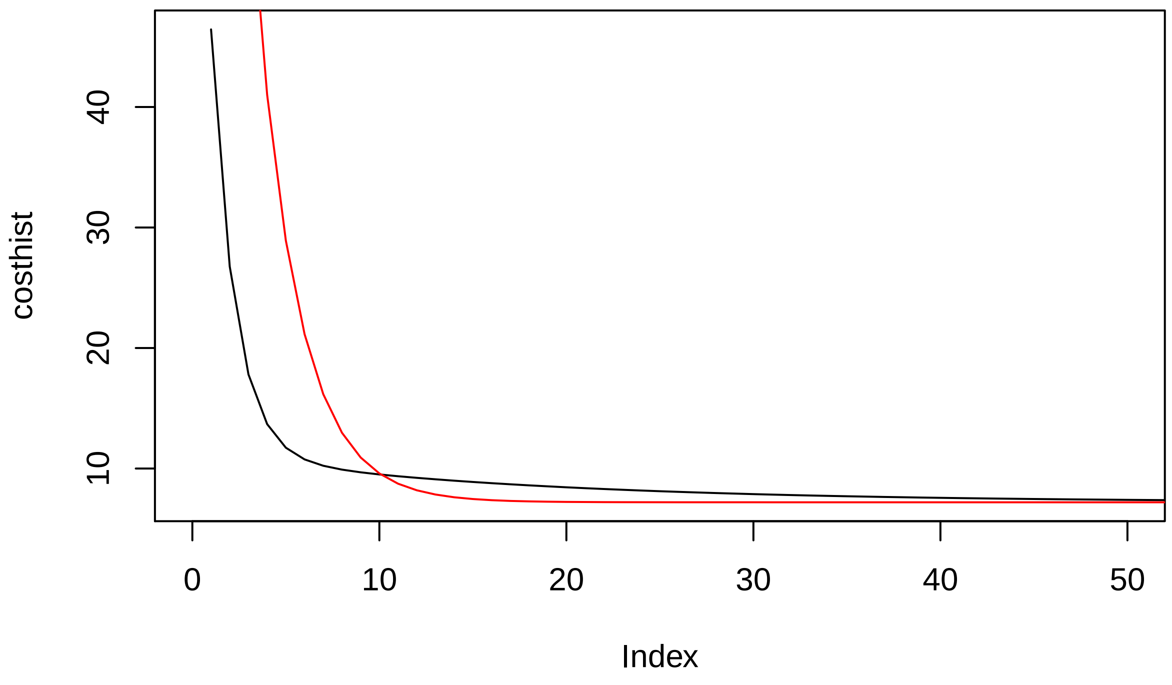
```
##         [,1]
## x0 5.000000
## x1 2.065591
```

```r
b0m<-seq(2,6, by=0.01)
b1m<-seq(-3,6, by=0.01)
points<-cbind(x.scaled[,2],y)
z <- outer(b0m,b1m,FUN="toterror",points=points)
contour(b0m,b1m,z,nlevels = 40)
points(thetahist,type="l",col="blue")
```



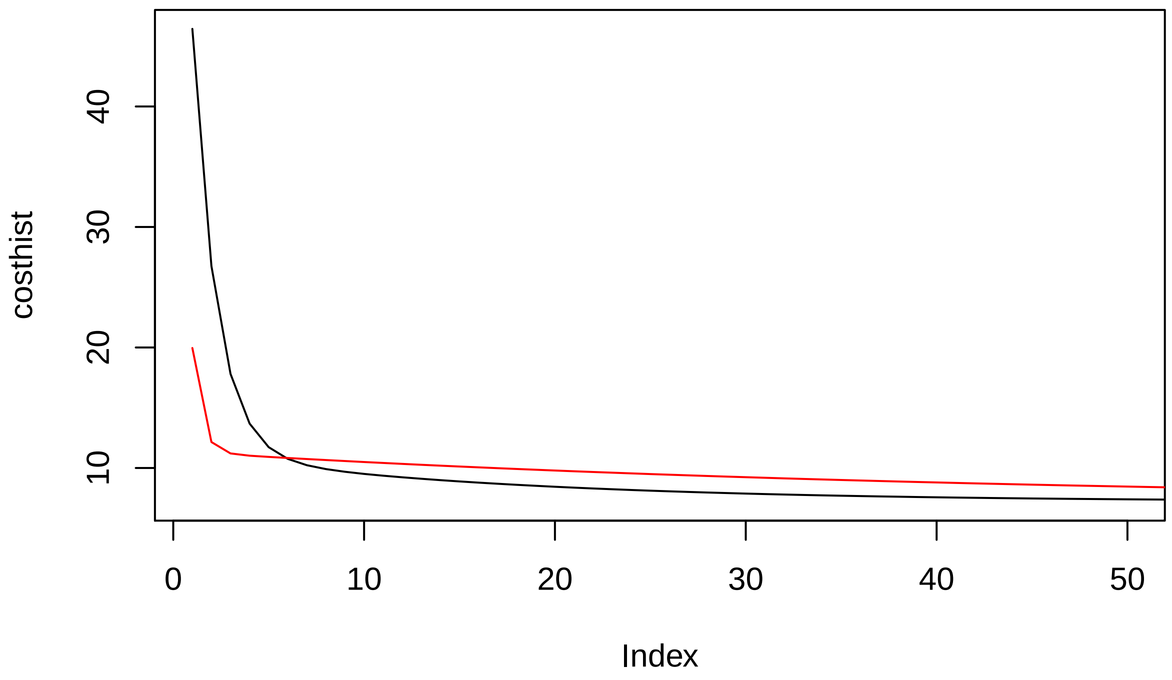
Compare: scaled data

```r
plot(costhist,type="l",xlim=c(0,50))
lines(costhist.scaled,type="l",col="red")
```



Compare: alpha

```r
alpha2 = 0.01
theta2 <- matrix(c(-2,1))
thetahist2 <- matrix(NA,nrow=1000,ncol=2)
costhist2 <- matrix(NA,nrow=1000,ncol=1)
for(i in 1:1000){
  theta2 <- theta2 - alpha2 * grad(x,y,theta2)
  thetahist2[i,] <- t(theta2)
  costhist2[i,] <- cost(x,y,theta2)
}
plot(costhist,type="l",xlim=c(1,50))
lines(costhist2,type="l",col="red")
```


costhist
10
20
30
40
0
10
20
30
40
50
Index