Question 4

Jin Barai

Gradient Descent

```
gradientDescent <- function(theta = 0, rhoFn, gradientFn, lineSearchFn, testConvergenceFn,
maxIterations = 100, tolerance = 1e-06, relative = FALSE, lambdaStepsize = 0.01,
lambdaMax = 0.5) {
converged <- FALSE
i <- 0
while (!converged & i <= maxIterations) {
g <- gradientFn(theta) ## gradient
glength <- sqrt(sum(g^2)) ## gradient direction
if (glength > 0)
g <- g/glength
lambda <- lineSearchFn(theta, rhoFn, g, lambdaStepsize = lambdaStepsize,
lambdaMax = lambdaMax)
thetaNew <- theta - lambda * g
converged <- testConvergenceFn(thetaNew, theta, tolerance = tolerance,
relative = relative)
theta <- thetaNew
i <- i + 1
}
## Return last value and whether converged or not
list(theta = theta, converged = converged, iteration = i, fnValue = rhoFn(theta))
}
```

Linear Search Method

```
LineSearch <- function(theta, pfn, gfn, lambdaStepsize = 0.01, lambdaMax = 1) {
abs(sin(theta))
}
```

Grid Linear Search

```
### line searching could be done as a simple grid search
gridLineSearch <- function(theta, rhoFn, g,
                      lambdaStepsize = 0.01,
                      lambdaMax = 1) {
  ## grid of lambda values to search
  lambdas <- seq(from = 0,
                 by = lambdaStepsize,
                 to = lambdaMax)

  ## line search
  rhoVals <- Map(function(lambda) {rhoFn(theta - lambda * g)},
```

```
                    lambdas)
  ## Return the lambda that gave the minimum
  lambdas[which.min(rhoVals)]
}
```