Computing Project - Lasso Algorithm

Adam Lee, Viviana Rosales, Carlos Rodriquez, Katrina Walker

March 22, 2017

Implementation for a Time-Series Analysis

Prior to implementing the GLASSO algorithm on a a panel of financial returns, we created the following function in R:

```
GLASSO <- function(Y, lambda = 0.05, t = 0.001, max_iter = 5000, beta_tol = 0.001){

  # Functions ------------------------------------------------------------------

  # Soft thresholding
  S_func <- function(x, t){
    sign(x) * max((abs(x) - t), 0)
  }

  # update beta
  update_beta <- function(u, V, beta, lambda){
    for (i in seq(1, (p - 1))){
      arg <- u[j] - V[-i,i] %*% beta[-i]
      beta[i] <- S_func(arg, lambda)
    }
    return(beta)
  }

  # GLASSO ---------------------------------------------------------------------

  # determine dimension
  p <- ncol(Y)

  # compute sample covariance matrix
  S <-  cov(Y)

  # compute tolerance
  S_temp <- S
  diag(S_temp) <- rep(0, p)
  tol <- t * mean(abs(S_temp))

  # initialise W & W_new for loop
  W <- S + lambda * diag(p)
  W_new <- W

  # intialise B, Theta
  B <- matrix(data=NA, nrow = (p - 1), ncol = p)
```

```
38    Theta <- matrix(data = NA, nrow = p, ncol = p)
39
40    # set intial error and iteration counter
41    err <- 1
42    it  <-  0
43    beta_err <- 1
44    # loop
45    while (err > tol & it < max_iter){
46
47      for (j in seq(1, p)){
48
49        V <- W[-j, -j]
50        u <- S[-j, j]
51
52        # initalise beta
53        beta <- rep(1, (p - 1))
54
55        # cyclical co-ordinate-descent
56        while (beta_err > beta_tol){
57          beta_new <- update_beta(u, V, beta, lambda)
58          beta_err = norm(as.matrix(beta_new - beta), type = 'f')
59          beta <- beta_new
60        }
61
62        # update w_12
63        W_new[-j, j] <- V %*% beta_new
64        B[,j] <- beta_new
65      }
66
67      # update error and iteration counter
68      err <- mean(abs(W-W_new))
69      it <- it + 1
70
71      # update W
72      W <- W_new
73    }
74
75    # recover Theta
76    for (j in seq(1, p)){
77      Theta[j, j] <- 1 / (W[j, j] - W[-j, j] %*% B[, j])
78      Theta[-j, j] <- -B[, j] * Theta[j, j]
79    }
80
81    return(Theta)
82  }
```

Demo

Problem: Given...
Inputs: ...
Outputs: ...
The Algorithm:

How it Works

The graphical lasso (GLASSO) algorithm is a variant of the least absolute shrinkage and selection operator (LASSO) which estimates a sparse inverse covariance matrix using a lasso (L1) penalty. The L1 penalty constrains the size of the coefficients by adding a penalty equivalent to the absolute value of the magnitude of the coefficients.....BLABLABLA

Specifically,.....BLABLABLA