Coordinate descent does not require knowledge of or computation of the derivative of the objective function; rather, it only considers the coordinates of the function itself. Whereas gradient descent moves in the direction of steepest descent, coordinate descent searches for the minimum of the function by separately moving along the axes of its coordinates. Gradient descent can be conceptualized by picturing a ball rolling down a hill until it reaches the bottom, while coordinate descent is more like someone starting at one corner of a city and following streets arranged in a grid pattern in order to reach the other side. Coordinate descent is preferred when computation and evaluation of a function's derivative is time consuming. Coordinate descent does not require knowledge of or computation of the derivative of the objective function; rather, it only considers the coordinates of the function itself. Whereas gradient descent moves in the direction of steepest descent, coordinate descent searches for the minimum of the function by separately moving along the axes of its coordinates. Gradient descent can be conceptualized by picturing a ball rolling down a hill until it reaches the bottom, while coordinate descent is more like someone starting at one corner of a city and following streets arranged in a grid pattern in order to reach the other side. Coordinate descent is preferred when computation and evaluation of a function's derivative is time consuming. When it comes to function minimization, it's time to open a book of optimization and linear algebra. I am currently working on variable selection and lasso-based solutions in genetics. What lasso does is basically minimizing the loss function and an penalty in order to set to zero some regression coefficients and select only those covariates that are really associated with the response. Pheew, the shortest summary of lasso ever!We all know that, provided the function to be minimized is convex, a good direction to follow, in order to find a local minimum, is towards the negative gradient of the function. Now, my question is how good or bad is following the negative gradient with respect to a coordinate descent approach that loops across all dimensions and minimizes along each?There is no better way to try this with real code and start measuring. Hence, I wrote some code that implements both gradient descent and coordinate descent.The comparison might not be completely fair because the learning rate in the gradient descent procedure is fixed at 0.1 (which in some cases might be slower indeed). But even with some tuning (maybe with some linear search) or adaptive learning rates, it's quite common to see that coordinate descent overcomes its brother gradient descent many times.This occurs much more often when the number of covariates becomes very high, as in many computational biology problems. In the figure below, I plot the analytical solution in red, the gradient descent minimisation in blue and the coordinate descent in green, across a number of iterations.A small explanation is probably necessary to read the function that performs coordinate descent. For a more mathematical explanation refer to the original post.It's quite common to see that coordinate descentovercomes its brother gradient descentCoordinate descent will update each variable in a Round Robin fashion. Despite the learning rate of the gradient descent procedure (which could indeed speed up convergence), the comparison between the two is fair at least in terms of complexity.Coordinate descent needs to perform operations for each coordinate update. Gradient descent performs the same number of operations . The R code that performs this comparison and generates the plot above is given belowlibrary(lassoshooting) # make data rm(list = ls(all = TRUE)) # make sure previous work is clear ls() p = 70; n = 300 x