Convex Optimization Homework

Kwa Jie Hao

12/5/2017

We have the original regression equation $y = X\beta + \epsilon$. In this case, we know that the optimal β^* contains zero values (the second parameter is zero). To recover the original support, we face the constrained optimization problem: Minimize $||y - X\beta||^2$ subject to $||\beta|| \leq t$. Or to write it in its lagrangian form,

$$min_\beta \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_1$$

In this case, we have been given $\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$ as the covariance matrix of the data.

$||y - X\beta||^2 + \lambda||\beta||_1$ can be split into a differentiable and non-differentiable portion. Let $f(\beta) = ||y - X\beta||^2$ and $g(\beta) = ||\beta||_1$. Then we know that the subdifferential of $f + g$ is given by $\nabla f + \delta g = \nabla f + S(\beta)$ where $S(\beta)$ is the subdifferential of $g(\beta)$. We have eventually:

$$0 \, \epsilon \, \nabla f + \delta g = (y - X\beta)^T X + \lambda S(\beta)$$

.

For the purposes of solving this assignment, I define recovering the correct support to be correctly identifying the second coefficient to be zero and the first coefficient to be non-zero (or zero if a is specified to be zero). This is because I expect estimates of the βs to be either shrinked to zero or of reasonable magnitude. Thus, it is sufficient to test for whether the parameter is zero or not.

For each parameter, we can consider a priori how they will affect the probability of recovering the proper support.

For ρ, when it is close to 0, it means that the Xs generated are independent. Then, we can solve for each β_i independently (dependent on λ value) in the above sub-differential equation. (When the columns of X satisfy $< X_i, X_i >= 1$, we know that if $\lambda > | < x, y > |$ then the solution is 0, and if $\beta_{ols,i} > \lambda$, $\hat{\beta} = \beta_{ols,i} - \lambda$ and vice-versa for $\beta_{ols,i} < -\lambda$.) Thus, I would expect the probability of recovering the correct support to be higher when rho approaches 0.

For ρ close to 1, then the columns of X become more and more similar to each other. The OLS solution becomes uncomputable at $\rho = 1$ since the covariance matrix is uninvertible. My guess is that it chooses one of the dimensions of X and calculates and estimate for that while ignoring the other (or setting it to zero).

For n, I expect performance to be better when it is large since the randomness from the data generation process is averaged out.

I suspect that when a is zero, it performs the worst since it is trying to get a signal from what is essentially noise from the data generating process (especially since I have set the default X generation process to initiate from zero). For example, if X are large and Y is very small, then it would be easy to detect that elements of β are either very small or close to zero. As such, I intentionally choose some values of a to be tested to be very small to see how LASSO performs in those cases.

Below, I include the sub-function which I will call to generate my simulations.

```
library(glmnet)
library(MASS)
library(ggplot2)
library(reshape2)
```

```
library(gridExtra)
library(grid)

generator <- function(a, rho, n){
  count <- 0
  for (i in 1:100){
    cov <- c(1,rho,rho,1)
    cov <- matrix(cov,2,2)
    x <- mvrnorm(n, mu=rep(0,2), Sigma=cov)

    beta <- c(a,0)

    y <- beta %*% t(x) + rnorm(n)
    tryCatch({
      fit = coef(cv.glmnet(x,y,intercept=FALSE))
        if (a == 0){
          if (fit[2] == 0 & fit[3] == 0){
           count <- count + 1
          }
         } else{
          if (fit[2] != 0 & fit[3] == 0){
           count <- count + 1
           }
         }
    },error=function(e){})
  }
  return(count)
}
```

Notes:

I choose to analyze the results which use the lambda.1se value (the default). I ran the simulations using both lambda.min (results3) and lambda.1se (results4) and and they are available for checking in the code chunks. There trends across both set of results are similar, but I choose to use lambda.1se because theoretically it gives a more regularized result.

One thing I did not understand was why lambda.min performed much better than lambda.1se at recovering the correct support. The value of λ is larger for lambda.1se (by definition) than for lambda.min. This means that technically we should expect more sparsity for lambda.1se than lambda.min, and thus recover the correct support more often. This however turned out to be the opposite.

The glmnet function failed when value of a was large and negative (< -20) and n was small. This was because there were not enough observations fold and calculate the cross-validation results. I resolved this by choosing a larger minimum value of n.

Discussion:

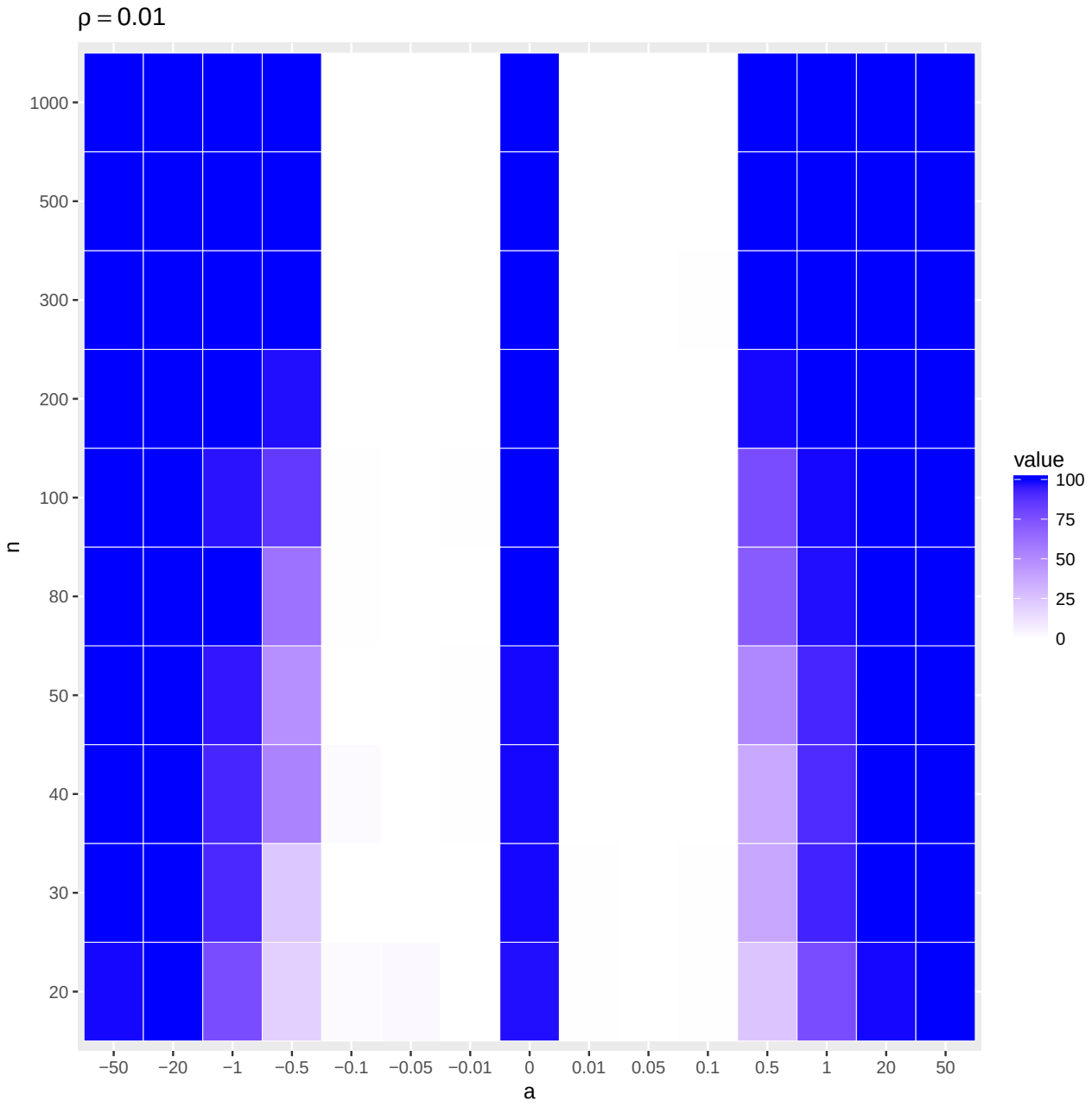
I have chosen to use heatmaps to represent my results. In this way I can express the 4 dimensional data by generating multiple plots when holding one dimension constant. For example, I can generate heatmaps of how probability varies with value of n and ρ for different values of a. Here, I choose to present plots for each value of ρ so that we can see how performance changes as data becomes more dependent. I've plotted the first one larger for better viewing. On the x-axis we have the different values of a and on the y-axis, n. "Value" represents the probability of recovering correct support multiplied by 100.

```
probability_table <- function(ns, rhos, as){
  table <- array(rep(0, length(ns)*length(rhos)*length(as)),
                 c(length(ns), length(rhos), length(as)))
```

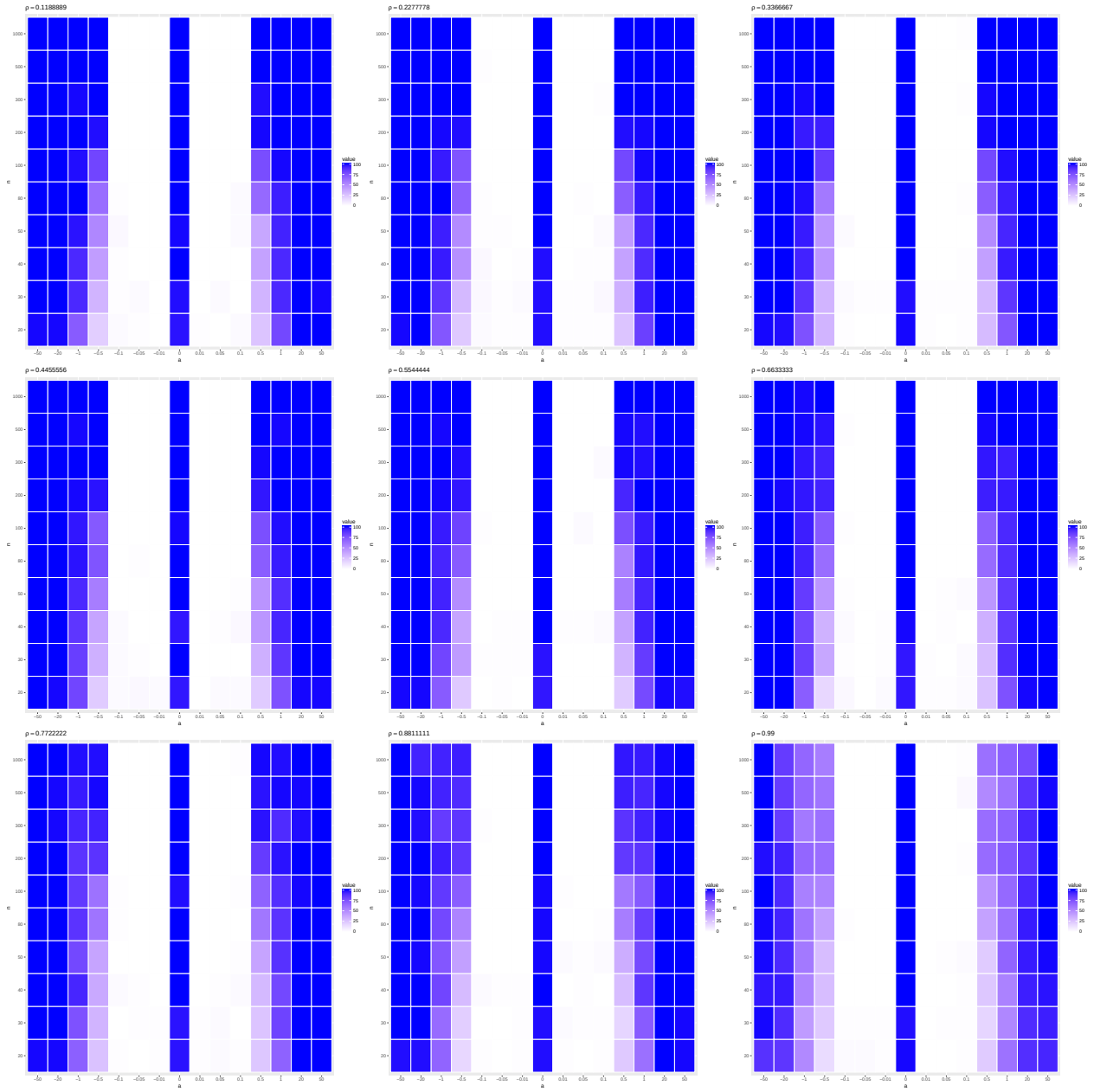
```
  for (i in 1:length(ns)){
    for (j in 1:length(rhos)){
      for (k in 1:length(as)){
        table[i,j,k] <- generator(as[k],rhos[j],ns[i])
        print(c(i,j,k))
      }
    }
  }
  return(table)
}

rho_list <- seq(0.01, 0.99, length=10)
n <- c(20, 30, 40, 50, 80, 100, 200, 300, 500, 1000)
a <- c(-50, -20, -1, -0.5, -0.1, -0.05, -0.01, 0, 0.01, 0.05, 0.1, 0.5, 1, 20, 50)

# results4 <- probability_table(n,rho_list,a)

getwd()
```

```
## [1] "/Users/kwajiehao/Documents/GitHub/sup/1/Deterministic Models/convex optimization/hw"
```

```
path <- "/Users/kwajiehao/Documents/Github/sup/1/Deterministic Models/convex optimization/hw/"
setwd(path)
load("results4.Rda")

generate_map <- function(data, index){

  # convert output into dataframe
  temp <- as.data.frame(data[,index,])
  names(temp) <- a
  rownames(temp) <- n
  temp <- cbind(temp, n)
  # make n a factor so that we can reshape temp
  temp$n <- as.factor(temp$n)
  temp <- melt(temp, id.vars="n")

  titl <- bquote(rho == .(rho_list[index]))

  plot_ <- ggplot(temp, aes(variable, n)) + geom_tile(aes(fill = value),colour = "white") +
    scale_fill_gradient(low = "white", high = "blue") +
    labs(title = titl, x="a")

  return(plot_)
}

for(i in 1:length(rho_list)){
  assign(paste0("df",i), generate_map(results4,i))
}

print(df1)
```




```
54
grid.newpage() # Open a new page on grid device
pushViewport(viewport(layout = grid.layout(3, 3)))
print(df2, vp = viewport(layout.pos.row = 1, layout.pos.col = 1))
print(df3, vp = viewport(layout.pos.row = 1, layout.pos.col = 2))
print(df4, vp = viewport(layout.pos.row = 1, layout.pos.col = 3))
print(df5, vp = viewport(layout.pos.row = 2, layout.pos.col = 1))
print(df6, vp = viewport(layout.pos.row = 2, layout.pos.col = 2))
print(df7, vp = viewport(layout.pos.row = 2, layout.pos.col = 3))
print(df8, vp = viewport(layout.pos.row = 3, layout.pos.col = 1))
print(df9, vp = viewport(layout.pos.row = 3, layout.pos.col = 2))
print(df10, vp = viewport(layout.pos.row = 3, layout.pos.col = 3))
```



(zoom for clearer view of the plots)

Firstly, it's clear that my hypothesis on performance improving with n holds. The darker shades of blue are located higher up on the heatmaps and lighter shades at lower values of n. This is true across all values of a and ρ. However, at lower values of ρ, the probability of recovering the correct support is still very high even at low n. The caveat is that this happens only at higher values of a. In fact, looking at the plots, it's clear that a has a very strong role in influencing recovery of support.

For a, we see that when it hits a sufficiently large value (greater than 1 in magnitude), performance is very good and LASSO has no problems identifying the correct support. It even does a decent job at small n. However, when a is close to zero (at values like 0.01), the probability of recovering the correct support becomes very low. This is a pattern across all the plots. Interestingly, there is a spike in performance when a is zero. LASSO correctly recovers the support at all values of ρ and n when a is zero but not when a is close to zero. If I had to make a guess for why this is the case, I would correct my previous hypothesis and say that $a = 0$, it is easy to predict that there is not relation between either of the predictors on the response and that

contribution from the random noise is very small. However, at a close to zero, the LASSO either finds it hard to tell which β is 0 and possibly distributes the contribution of each predictor evenly so that β is non-zero.

When data is less correlated, I find that performance is better. This is in line with my reasoning that when ρ is 0 the covariates are independent and we can solve for the β individually. Again, this is subject to high enough values of a. However, we see that performance goes down when ρ becomes larger and closer to 1. This is ilustrated by the patch of lighter purple growing out and infecting more of the heatmap as we go from low ρ values to high ρ values. Looking across the plots, it appears that low ρ and large n can mitigate the effect of a small a but only to a certain degree.

Ultimately, beyond $n = 50$, a seems to have the largest impact on performance. Across all combinations of parameter values, probability of correct recovery goes to zero when a is near-zero but not zero. This is followed by ρ, which mitigates the effect of a smaller a when ρ is smaller. In an ideal scenario, to guarantee high probability of correct support, we would hope that n is large, ρ is sufficiently small and that the values of the β are sufficiently differentiated from each other (not close to zero if one of them is zero).