Power Comparisons Tutorial

Emily Winn-Nunez

2023-10-20

This script will demonstrate the power of distributional centrality via RATE measures. Specifically it compares: (1) L1- regularized lasso regression; (2) The combined regularization utilized by the elastic net; (3) RATE, a KL-divergence based variable selection model (4) A genome scan with individual single nucleotide polymorphisms (SNPs) fit via a univariate linear model (SCANONE). (5) GOALS, a measure to assess global and local variable importance.

NOTE: This script is based on the genetics simulations from Crawford et al. (2018). Here, we simulate genotype data for $n = 500$ individuals with $p = 1000$ independent genetic variants. We randomly assume that $j = 30$ predictors are causal and have true association with the generated (continuous) phenotype y. We then assume that the j predictor variables explain a fixed $H^2\%$ (phenotypic variance explained; PVE) of the total variance in the response $V(y)$. This parameter H^2 can alternatively be described as a factor controlling the signal-to-noise ratio.The parameter ρ represents the proportion of H^2 that is contributed by additive effects versus interaction (epistatic) effects. Namely, the additive effects make up $\rho\%$, while the pairwise interactions (epistatic effects) make up the remaining $(1 - \rho)\%$.

NOTE: We also include a scenario where population stratification effects are introduced into the simulations by allowing the top 5 genotype principal components (PCs) Z to make up an additional percentage of the overall variation in the trait (see variable `pc.var`). The effect sizes for these stratification effects are also drawn from a standard normal distribution. Alternatively, one can think of the combined effect of the PCs as structured noise.

```r
### Load in the RATE R functions (only need for comparisons) ###
source("RATE.R")
### Load in the C++ BAKR functions ###
sourceCpp("BAKRGibbs.cpp")
#Register cores
cores = detectCores()
registerDoParallel(cores=cores)
```

Define the Compute Power Function

```r
compute.power <- function(pvals,SNPs){
  nsnps = length(pvals)
  Pos = SNPs #True Positive Set
  Negs = names(pvals)[which(names(pvals)%in%SNPs==FALSE)] #True Negative Set
  x = foreach(i = 1:nsnps)%dopar%{
    v = sort(pvals,decreasing = TRUE)[1:i] #Test Positives
    z = pvals[which(names(pvals)%in%names(v)==FALSE)] #Test Negatives

    TP = length(which(names(v)%in%Pos==TRUE))
    FP = length(which(names(v)%in%Pos==FALSE))
    TN = length(which(names(z)%in%Negs==TRUE))
    FN = length(which(names(z)%in%Negs==FALSE))
    TPR = TP/(TP+FN); FPR = FP/(FP+TN); FDR = FP/(FP+TP)
    c(TPR,FPR,FDR)
  }
  return(matrix(unlist(x),ncol = 3,byrow = TRUE))
}
```

Simulate Data

```r
### Set the random seed to reproduce research ###
set.seed(11151990)

### Specify the Number of Samples and Genetic Markers ###
n = 500; p = 1e3;

### Set up simulation parameters ###
pve=0.3; rho=0.5; pc.var = 0; ncausal = 30

### The Number of Causal Variables ###
ncausal1= ncausal/6 #Set 1 of causal SNPs
ncausal2 = ncausal-ncausal1 #Set 2 of Causal SNPs

### Generate the data ###
maf <- 0.05 + 0.45*runif(p)
X   <- (runif(n*p) < maf) + (runif(n*p) < maf)
X   <- matrix(as.double(X),n,p,byrow = TRUE); Geno = X
Xmean=apply(X, 2, mean); Xsd=apply(X, 2, sd); Xc=t((t(X)-Xmean)/Xsd)
s=sample(1:p,ncausal,replace = FALSE)

#Select Causal SNPs
s1=sample(s, ncausal1, replace=F)
s2=sample(s[s%in%s1==FALSE], ncausal2, replace=F)

#Generate the ground-truth regression coefficients for the variables (X).
#Adjust the effects so that the variables (SNPs) explain x percent of the variance in the outcome.

Xcausal1=X[,s1]; Xcausal2=X[,s2];
Xepi=c()
for(i in 1:ncausal1){
  Xepi=cbind(Xepi,Xcausal1[,i]*Xcausal2)
}
dim(Xepi)
```

```
## [1] 500 125
```

```r
# Marginal Effects Only
Xmarginal=cbind(X[,s])
beta=rnorm(dim(X[,s])[2])
y_marginal=c(Xmarginal%*%beta)
beta=beta*sqrt(pve*rho/var(y_marginal))
y_marginal=Xmarginal%*%beta

#Pairwise Epistatic Effects
beta=rnorm(dim(Xepi)[2])
```

```r
y_epi=c(Xepi%*%beta)
beta=beta*sqrt(pve*(1-rho)/var(y_epi))
y_epi=Xepi%*%beta

### Compute the Top PCs ###
PCs = ComputePCs(X,10)

### Define the effects of the PCs ###
beta=rnorm(dim(PCs)[2])
y_pcs=c(PCs%*%beta)
beta=beta*sqrt(pc.var/var(y_pcs))
y_pcs=PCs%*%beta

### Error Term ###
y_err=rnorm(n)
y_err=y_err*sqrt((1-pve-pc.var)/var(y_err))

### Simulate the Response ###
y=c(y_marginal+y_epi+y_pcs+y_err) #Full Model
y=(y-mean(y))/(sd(y)) #Centered and Scaled

s = c(s1,s2)
colnames(X) = paste("SNP",1:ncol(X),sep="")
```

Running Bayesian Gaussian Process (GP) and GOALS

Create the Nonlinear Covariance Matrix with the Gaussian Kernel

This function takes on two arguments: (1) The Genotype matrix X. This matrix should be fed in as a $p \times n$ matrix. That is, predictor are the rows and subjects/patients/cell lines are the columns. (2) The bandwidth (also known as a smoothing parameter or lengthscale) h. For example, the Gaussian kernel can be specified as $k(u,v) = \exp||u-v||^2/2h^2$.

```r
n = dim(X)[1] #Sample size
p = dim(X)[2] #Number of markers or genes

### Find the Approximate Basis and Kernel Matrix; Choose N <= D <= P ###
sigma2 = 1e-3
B = GaussKernel(t(X)); diag(B)=1
A <- B + sigma2*diag(1,nrow=n,ncol=n)
A_svd <- svd(A)
Ainv = nearPD(A_svd$v%*%diag(1/A_svd$d, nrow=n, ncol=n)%*%t(A_svd$u))$mat
Aiy <- Ainv%*%y
BAiy <- B %*% Aiy
IAiB <- (diag(1,nrow=n, ncol=n)-Ainv%*%B)
BIAiB <- B%*%IAiB

#Calculate Delta

delta = ComputeESAFast(X, B, as.vector(Aiy), as.vector(BAiy), cores=cores)

global_delta = abs(colMeans(delta))
```

Running the Bayesian Gaussian Process (GP) Regression Model

Here, we do the same as above but with the centered data instead of the raw data.

```
Kn = GaussKernel(t(Xc)); diag(Kn) = 1

v=matrix(1, n, 1)
M=diag(n)-v%*%t(v)/n
Kn=M%*%Kn%*%M
Kn=Kn/mean(diag(Kn))

### Gibbs Sampler ###
sigma2 = 1e-3
fhat = Kn %*% solve(Kn + diag(sigma2,n), y)
fhat.rep = rmvnorm(5e3,fhat,Kn - Kn %*% solve(Kn+diag(sigma2,n),Kn))

### Run the RATE Function ###
beta.tilde = t(apply(fhat.rep,1,function(x) return(cov(Xc,x))))
```

NOTE: We formally define the effect size analogue as the result of projecting the design matrix X onto the nonlinear response vector f, where $\beta = \mathrm{Proj}(X, f) = X^{\dagger} f with X^{\dagger}$ symbolizing the Moore-Penrose generalized inverse.

Compute the First Order Centrality of each Predictor Variable

The function results in a list with: (1) The raw Kullback-Leibler divergence measures (RATEKLD); (2)$Therelativecentralityme$
(3) The entropic deviance from uniformity (RATE$Detla); and (4) The calibrating approximate effect sample size (ESS) measures from importance sampling (Gruber and West, 2016, 2017)

```
nl = NULL
res = RATE(X=X,beta.draws = beta.tilde, rank.r=n/4, low.rank=TRUE, snp.nms = colnames(X),cores = cores)

### Get the Results ###
rates = res$RATE
```

Run other Methods

For comparisson, we run LASSO, ElasticNet, and SCANONE.

```
### LASSO ###
fit= cv.glmnet(X, y,intercept=FALSE,alpha=1)
lasso = as.matrix(coef(fit,s = fit$lambda.1se))
lasso = c(lasso[-1,])

### Elastic Net ###
fit= cv.glmnet(X, y,intercept=FALSE,alpha=0.5)
enet = as.matrix(coef(fit,s = fit$lambda.1se))
enet = c(enet[-1,])

### Scan One ###
lp_scanone = sapply(1:ncol(X),function(i) -log10(summary(lm(y~X[,i]))$coef[2,4]))
```

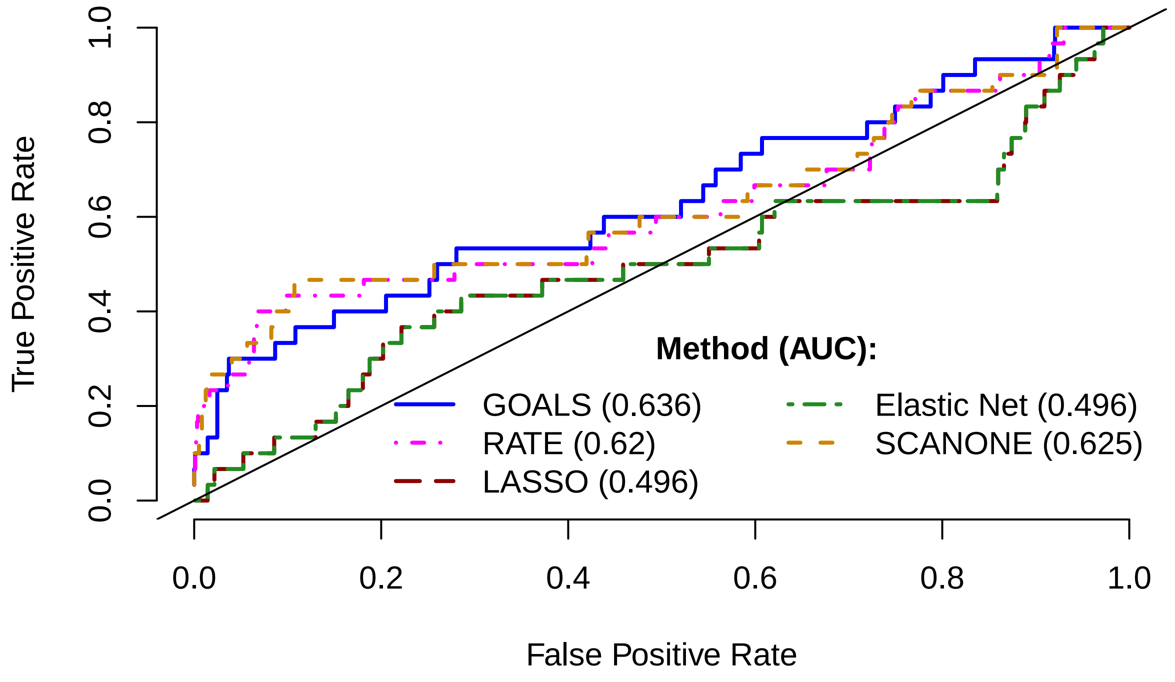
Compute Power of Each Metric

```
b = global_delta; names(b) = colnames(X)
power.goals = compute.power(b,colnames(X)[s])
```

```
b = rates; names(b) = colnames(X)
power.rate = compute.power(b,colnames(X)[s])

b = abs(lasso); names(b) = colnames(X)
power.lasso= compute.power(b,colnames(X)[s])

b = abs(enet); names(b) = colnames(X)
power.enet= compute.power(b,colnames(X)[s])

b = lp_scanone; names(b) = colnames(X)
power.scanone = compute.power(b,colnames(X)[s])
```

Plot Results

```
PP = cbind(power.goals,power.rate,power.lasso,power.enet,power.scanone)
auc = c(trapz(PP[,2],PP[,1]),trapz(PP[,5],PP[,4]),trapz(PP[,8],PP[,7]),trapz(PP[,11],PP[,10]),trapz(PP[
m = paste(c("GOALS","RATE","LASSO","Elastic Net","SCANONE")," (", round(auc,3), ") ",sep = "")

### Visualize these power comparisons ###
plot(PP[,2],PP[,1],type = "l",lty = 1, lwd = 2, col = "blue", xlab = "False Positive Rate",ylab = "True
lines(PP[,5],PP[,4],type = "l",lty = 4, lwd = 2, col = "magenta")
lines(PP[,8],PP[,7],type = "l",lty = 5, lwd = 2, col = "dark red")
lines(PP[,11],PP[,10],type = "l",lty = 6, lwd = 2, col = "forest green")
lines(PP[,14],PP[,13],type = "l",lty = 2, lwd = 2, col = "orange3")
legend("bottomright",legend = m, lty =c(1,4,5,6,2), lwd = c(2,2,2,2,2),col = c("blue","magenta","dark r
abline(a=0, b=1,col="black")
```


True Positive Rate
False Positive Rate
Method (AUC):
GOALS (0.636)
RATE (0.62)
LASSO (0.496)
Elastic Net (0.496)
SCANONE (0.625)