1 Convex sets and functions.

During class (Lecture 3), I claimed that the following set is convex:

$$C = \{x \in \mathbb{R}^k : x_1 A_1 + \cdots + x_k A_k \preceq B\}$$

where $A_1, \ldots, A_k, B$ are $n \times n$ symmetric matrices. Prove this result using the definition of a convex set.

Solution:

Let $x, y \in C$. Then $x_1 A_1 + \cdots + x_k A_k \preceq B$ and $y_1 A_1 + \cdots + y_k A_k \preceq B$. So for $t \in (0,1)$, $tx_1 A_1 + \cdots + tx_k A_k \preceq tB$ and $(1-t)y_1 A_1 + \cdots + (1-t)y_k A_k \preceq (1-t)B$, and therefore $(tx_1 + (1-t)y_1)A_1 + \cdots + (tx_k + (1-t)y_k)A_k \preceq B$. Thus C is convex.

2 Duality.

2.1

Derive the dual of a general LP (note the solution in the notes):

$$\begin{aligned} \min_x \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & Gx \le h. \end{aligned}$$

Solution:

We have

$$\begin{aligned} Ax = b & \qquad u^\top Ax = u^\top b \\ Gx \le h & \quad \Rightarrow \quad v^\top Gx \le v^\top h \\ & \qquad v \ge 0 \end{aligned}$$

Therefore,

$$\begin{aligned} & u^\top Ax + v^\top Gx u^\top b + v^\top h \\ \Rightarrow \quad & -(A^\top u + G^\top v)^\top x \ge -b^\top u - h^\top v \end{aligned}$$

Thus, the dual is given by

$$\begin{aligned} \max_{u,v} \quad & -b^\top u - h^\top v \\ \text{subject to} \quad & -A^\top u - G^\top v = c \\ & v \ge 0. \end{aligned}$$

2.2

Consider the simpler LP

$$\begin{aligned}\min_x \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0,\end{aligned}$$

along with the related problem

$$\begin{aligned}\min_x \quad & c^\top x - \tau \sum_i \log(x_i) \\ \text{subject to} \quad & Ax = b.\end{aligned}$$

The second version is sometimes called the log barrier function, and acts as a 'soft' inequality constraint, because it will tend to positive infinity as any of the x_i tend to zero from the right. Throughout, assume that $\{x : x > 0,\ Ax = b\}$ and $\{y : A^\top y > -c\}$ are non-empty. i.e. the primal LP and its dual are both strictly feasible.

i. Derive the dual and the KKT conditions for the original problem.
ii. Derive the dual and the KKT conditions for the log barrier problem (note the implicit constraint on x given by the domain of the objective).
iii. Describe the differences in the two KKT conditions. (Hint: what can you observe about the second set of KKT conditions when τ is taken to be small?)

Solution:

i. Using the previous part, the dual is

$$\begin{aligned}\max_{u,v} \quad & -b^\top u \\ \text{subject to} \quad & -A^\top u + v = c \\ & v \geq 0.\end{aligned}$$

The KKT conditions are:

1. $0 = \partial\left(c^\top x - u^\top x + v^\top(Ax - b)\right) = c - u + A^\top v$ (Stationarity)
2. $u_i x_i = 0,\ \forall i$ (complementary slackness)
3. $x \geq 0$, $Ax = b$ (primal feasibility)
4. $v \geq 0$ (dual feasibility)

ii. The dual is a bit hard to get, but simplifies nicely. Assuming $A \in \mathbb{R}^{m\times n}$:

$$\max_u \quad \tau \sum_i^n \log\left(\frac{(A^\top u)_i + c_i}{\tau}\right) - n - b^\top u.$$

This is an unconstrained concave maximization, and the primal had no (explicit) inequality constraints.

The KKT conditions are therefore:

1. $0 = \partial\left(c^\top x - \tau \sum_i \log(x_i) + v^\top(Ax - b)\right) = c - \tau \sum_i \frac{1}{x_i} + A^\top v$ (Stationarity)
2. (complementary slackness is vacuous)

3. $Ax = b,\ x > 0$ (primal feasibility, the constraint on x is implicit in the domain of log)
4. (dual feasibility is vacuous)

iii. Consider the second set of KKT conditions, and define $u_i = 1/(\tau x_i)$. Then $u_i x_i = 1/\tau$. Compare this to the complementary slackness condition for the first problem. Thus, for large τ, the log barrier version gives feasible points which nearly satisfy the KKT conditions for the original optimization. This technique is the basis of the "barrier methods" for handling equality constriants.

3 Algorithms.

Recall the lasso problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

Use any programming language you like to implement the following tasks.

1. Generate data. Let $X \in \mathbb{R}^{n\times p}$ consist of iid entries from a normal distribution with mean 0 and variance 1. Take $n = 100$ and $p = 25$. Set $\beta = (5, 5, 5, 0, \ldots, 0)$ and let $y = X\beta + \epsilon$ with ϵ containing iid entries with mean 0 and variance 0.1. To do this in `R` to match my solutions use

2. Estimate the lasso using the following four techniques: (a) subgradient descent, (b) proximal gradient descent (here, this is called ISTA for Iterated Soft Thresholding Algorithm), (c) coordinate descent, and (d) ADMM. For each method, track f for 50 iterations.

3. Repeat (1) and (2) 100 times.

4. Produce a plot with the iterations on the x-axis and $\log_{10}(f - f^*)$ on the y-axis. You should plot all 400 lines (thin) as well as the mean of each method (thick). My result is shown below so that this is clear.

Solution:

My functions are included here.

```
soft <- function(arg, thresh){
  sgn = sign(arg)
  adj = pmax(abs(arg)-thresh, 0)
  return(sgn*adj)
}

lasso_gd <- function(yy, Xy, XX, lambda, t0=.02, b0 = rep(c(1,-1),length=25), maxit=1e2){
  iter = 1
  beta = b0
  f = double(maxit)
  while(iter <= maxit){
    f[iter] = (yy - 2*t(beta) %*% Xy + t(beta) %*% XX %*% beta)/2
    beta = beta + t0/iter * ((Xy - XX %*% beta) + sign(beta))
    iter = iter + 1
  }
  return(list(f=f, beta=beta))
}
```

```r
lasso_ista <- function(yy, Xy, XX, lambda, t0=.2, b0 = rep(c(1,-1),length=25), maxit=1e2){
  iter = 1
  beta = b0
  f = double(maxit)
  while(iter <= maxit){
    f[iter] = (yy - 2*t(beta) %*% Xy + t(beta) %*% XX %*% beta)/2
    t = t0/iter
    beta = soft(beta + t * (Xy - XX %*% beta), lambda*t)
    iter = iter + 1
  }
  return(list(f=f, beta=beta))
}


lasso_cd <- function(yy, Xy, XX, lambda, b0 = rep(c(1,-1),length=25), maxit=1e2){
  p = length(Xy)
  iter = 1
  beta = b0
  f = double(maxit)
  while(iter <= maxit){
    f[iter] = (yy - 2*t(beta) %*% Xy + t(beta) %*% XX %*% beta)/2
    for(j in 1:p){
      num = Xy[j] - XX[j,-j] %*% beta[-j]
      beta[j] = soft( num / XX[j,j], lambda / XX[j,j])
    }
    iter = iter + 1
  }
  return(list(f=f,beta=beta))
}

lasso_admm <- function(yy, Xy, XX, lambda, rho=10, maxit=1e2){
  iter = 1
  f = double(maxit)
  p = ncol(XX)
  XXinv = solve(XX+rho*diag(p))
  XXinvXy = XXinv %*% Xy
  alpha = double(p)
  w = double(p)
  while(iter <= maxit){
    beta = XXinvXy + rho*XXinv %*% (alpha-w)
    alpha = soft(beta + w, lambda/rho)
    w = w + beta - alpha
    f[iter] = (yy - 2*t(beta) %*% Xy + t(beta) %*% XX %*% beta)/2
    iter = iter + 1
  }
  return(list(f=f, beta=beta))
}
```

This is what I ran to evaluate the algorithms 100 times.

```r
n = 100
p = 25
sig = 0.1
```

```
toreplicate <- function(n, p, sig){
  beta = c(5,-5,5,-5,rep(0,p-4))
  X = matrix(rnorm(n*p),nrow=n)
  epsilon = rnorm(n,sd=sig)
  y = X %*% beta + epsilon
  XX = crossprod(X)
  Xy = drop(crossprod(X,y))
  yy = drop(crossprod(y))
  gd = lasso_gd(yy, Xy, XX, 1, t0=.015, maxit = 50)
  ista = lasso_ista(yy, Xy, XX, 1, t0=.015, maxit=50)
  cd = lasso_cd(yy, Xy, XX, 1, maxit=50)
  admm = lasso_admm(yy, Xy, XX, 1, rho=100, maxit=50)
  fstar = min(gd$f, ista$f,cd$f, admm$f)
  return(cbind(gd=gd$f, ista = ista$f, cd=cd$f, admm=admm$f)-fstar)
}
set.seed(20170926)
res = replicate(100, toreplicate(n,p,sig))
m = apply(res, 1:2, mean)
```

The resulting figure:

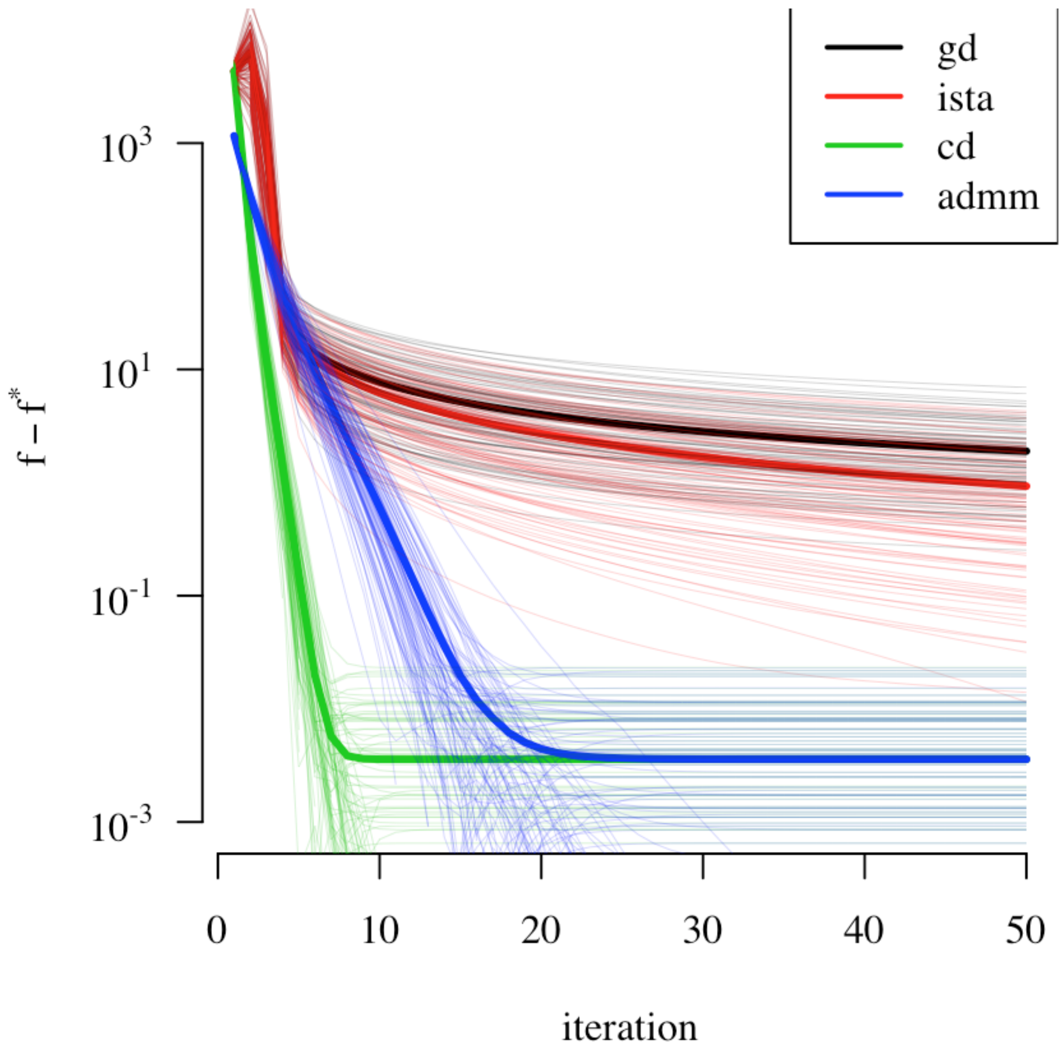

FIGURE 1: *My output.*