Convex Optimization - Homework 3

Report plots, comments and theoretical results in a pdf file. Send your code together with the requested functions and a main script reproducing all your experiments. You can use Matlab, Python or Julia.

Given $x_1, \dots, x_n \in \mathbf{R}^d$ data vectors and $y_1, \dots, y_n \in \mathbf{R}$ observations, we are searching an explanatory variable $w \in \mathbf{R}^d$ which fits data inputs to the observations y by minimizing their squared difference. In high dimensional settings (when $n \ll d$) a ℓ_1 penalty norm is often used on the regression coefficients w in order to enforce sparsity of the solution (so that w will only have a few non-zeros entries). Such penalization has well known statistical properties, and makes the model more interpretable and faster at test time.

From an optimization point of view we want to solve the following problem called LASSO (which stands for Least Absolute Shrinkage Operator and Selection Operator) :

$$\text{minimize} \quad \tfrac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_1 \tag{LASSO}$$

in the variable $w \in \mathbf{R}^d$, where $X = (x_1^T, \dots, x_n^T) \in \mathbf{R}^{n\times d}, y = (y_1, \dots, y_n) \in \mathbf{R}^n$ and $\lambda > 0$ is a regularization parameter.

1 Second order methods for dual problem

1. Derive the dual problem of LASSO and write it as a general Quadratic Problem :

$$\begin{array}{ll} \text{minimize} & v^T Q v + p^T v \\ \text{subject to} & Av \preceq b \end{array} \tag{QP}$$

in variable $v \in \mathbf{R}^n$, where $Q \succeq 0$.

2. Implement the barrier method to solve QP.

- Write a function `v_seq = centering_step(Q,p,A,b,t,v0,eps)` which implements the Newton method to solve the centering step given the inputs (Q, p, A, b), the barrier method parameter t (see lectures), initial variable v_0 and a target precision ϵ. The function outputs the sequence of variables iterates $(v_i)_{i=1,\dots,n_\epsilon}$, where n_ϵ is the number of iterations to obtain the ϵ precision. Use a backtracking line search with appropriate parameters.
- Write a function `v_seq = barr_method(Q,p,A,b,v0,eps)` which implements the barrier method to solve QP using precedent function given the data inputs (Q, p, A, b), a

feasible point v_0, a precision criterion ϵ. The function outputs the sequence of variables iterates $(v_i)_{i=1,\ldots,n_\epsilon}$, where n_ϵ is the number of iterations to obtain the ϵ precision.

3. Test your function on randomly generated matrices X and observations y with $\lambda = 10$. Plot objective function values versus iterations. Plot precision criterion and gap $f(v_t) - f^*$ in semilog scale (using the best value found for f as a surrogate for f^*). Repeat for different values of the barrier method parameter $\mu = 2, 15, 50, 100$. What would be an appropriate choice for μ ?

2 First order methods for primal problem

1. Write a function `w_seq = subgrad(X,y,lambda,eps)` which implements the sub-gradient descent algorithm for LASSO. Test it for different step-size strategies. Plot objective function values versus iterations. Plot $f(w_t) - f^*$ in semilog scale (using the best value found for f as a surrogate for f^*).

2. Write a function `w_seq = coord_descent(X,y,lambda,eps)` which implements the coordinate descent algorithm for the LASSO dual. Plot convergence in semilog scale (using the best value found for f as a surrogate for f^*). Compare it with the sub-gradient method in terms of iterations and total CPU time to obtain a gap of $\epsilon = 10^{-3}, 10^{-6}$ and 10^{-10}.

3 Proximal methods for primal problem

Problem LASSO belongs to the general class of composite optimization problems which can be written

$$\text{minimize} \quad \phi(x) = f(x) + h(x) \tag{CP}$$

in variable $x \in \mathbf{R}^d$, where f is a differentiable smooth convex function with smooth parameter L and h is a non-differentiable convex function.

1. *(optional)* In the case of LASSO for any sizes of the problem (n, d), is the function f strongly convex or smooth? If yes what are the corresponding parameters? What happens for $n \ll d$? In the following we will use the proximal operator of h defined for $x \in \mathbf{R}^d$ and a real positive number $P > 0$ as

$$\text{prox}_{h,P}(x) = \arg\min_z \ \frac{P}{2}\|z - x\|_2^2 + h(z)$$

2. *(optional)* What is the proximal operator of an indicator function of a convex set ? Derive it for $h(x) = \|x\|_1$.

In order to minimize the objective function we will minimize an upper bound of ϕ at each iterate x_t for $t \geq 0$. For a fixed $x \in \mathbf{R}^d$ and a positive constant $M > 0$, we define the gradient mapping function :

$$g_{x,M}(z) = f(x) + \nabla f(x)^T(z - x) + \frac{M}{2}\|z - x\|_2^2 + h(z)$$

3. *(optional)* For which values of M is $g_{x,M}$ an upper bound on ϕ ? Derive in terms of gradient of f and proximal operator of h, the iteration scheme :

$$x_{t+1} = \arg\min_z g_{x_t,M}(z)$$

What do we obtain for $h = 0$ or h the indicator function of a convex set ? Without further assumptions on the functions f and h this algorithm is proven to converge in at most $O(1/\epsilon)$ iterations to reach an ϵ precision.

4. *(optional)* Write a function `w_seq = prox_method(X,y,lambda,eps,f_min)` which implements the proximal method developed above for the LASSO problem. Plot $f(w_t) - f^*$ in semilog scale (using the best value found for f as a surrogate for f^*). Compare this with the other first order methods in terms of iterations and total CPU time to obtain a gap of $\epsilon = 10^{-3}, 10^{-6}, 10^{-12}$ (include time to compute L).

5. *(optional)* Prove that the proximal gradient method is a point-fix algorithm.

5. *(optional)* Code a function `w_seq = prox_acc_method(X,y,lambda,eps,f_min)` which implements the accelerated version of the proximal gradient method. You can use the FISTA scheme:

$$y_t = x_t + \theta_t(\theta_{t-1}^{-1} - 1)(x_t - x_{t-1})$$
$$x_{t+1} = \text{prox}_{h,L}\left(y_t - \frac{1}{L}\nabla f(y_t)\right)$$

with $\theta_0 = \theta_{-1} = 0$ and $\theta_t = \frac{2}{k+2}$. Without further assumptions on the functions f and h this algorithm is proven to converge in at most $O(1/\sqrt{\epsilon})$ iterations to reach an ϵ precision. Plot the same graphs as for the other algorithms and compare them with the other first order methods in terms of iterations and total CPU time to obtain a gap of $\epsilon = 10^{-3}, 10^{-6}, 10^{-12}$ (include time to compute L).