Optimization Methods for Data Science, Machine Learning and AI
Prof. Yinyu Ye

Homework Assignment 3 Sample Solution

1. (10') In most real applications, the (first-order) Lipschitz constant β is unknown. Furthermore, we like to use a localized Lipschitz constant β^k at iteration k such that

$$f(\mathbf{x}^k + \alpha\mathbf{d}^k) - f(\mathbf{x}^k) - \nabla f(\mathbf{x}^k)^T(\alpha\mathbf{d}^k) \leq \frac{\beta^k}{2}\|\alpha\mathbf{d}^k\|^2, \tag{1}$$

where $\mathbf{d}^k$ is the steepest descent direction $-\nabla f(\mathbf{x}^k)$. The goal is to decide a step-size $\alpha \approx \frac{1}{\beta^k}$.

Consider the following *forward-backward tracking method*. In the following, assume that $\beta^k \geq 1$ and $\alpha_{\max} \geq 1/\beta^k$. Notice that if $\beta^k < 1$, we can enforce it to satisfy our assumption by replacing it with $\max\{1, \beta^k\}$.

Now start at a initial guess $\alpha > 0$,

i) If $\alpha \leq \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$, then doubling the step-size: $\alpha \leftarrow 2\alpha$, stop as soon as the inequality is reversed or $\alpha > \alpha_{\max}(> 0)$, and select the latest α such that the inequality ($\alpha \leq \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$) holds and $\alpha \leq \alpha_{\max}$.

ii) Otherwise halving the step-size: $\alpha \leftarrow \alpha/2$; stop as soon as $\alpha \leq \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$ and return it.

(a) (4') Let $\bar{\alpha}$ be a step-size generated by the scheme. Show that $\bar{\alpha} \geq \frac{1}{2\beta^k}$.

(b) (3') Prove that the above scheme will terminate in finite steps.

(c) (3') Show that $f(\mathbf{x}^k + \bar{\alpha}\mathbf{d}^k) \leq f(\mathbf{x}^k) - \frac{1}{4\beta^k}\|\mathbf{d}^k\|_2^2$.

Solution:

(a) Based on the above definition, the procedure will only terminate in the following two cases:

- Case I: $2\bar{\alpha} > \alpha_{\max}$. In this case we clearly have $\bar{\alpha} > \frac{1}{2}\alpha_{\max}$ and thus $\bar{\alpha} \geq \frac{1}{2\beta^k}$ since $\alpha_{\max} \geq \frac{1}{\beta^k}$ as assumed.
- Case II: $2\bar{\alpha} > \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+2\bar{\alpha}\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$. According to (1) it is the case that

$$f(\mathbf{x}^k) - f(\mathbf{x}^k + 2\bar{\alpha}\mathbf{d}^k) \geq 2\bar{\alpha}\|\mathbf{d}^k\|^2 - \frac{1}{2}(2\bar{\alpha})^2\beta^k\|\mathbf{d}^k\|^2 = 2(\bar{\alpha} - \beta^k\bar{\alpha}^2)\|\mathbf{d}^k\|^2$$

Combining above two inequalities gives

$$2\bar{\alpha} > 4(\bar{\alpha} - \beta^k\bar{\alpha}^2),$$

which implies $\bar{\alpha} \geq \frac{1}{2\beta^k}$ since $\bar{\alpha}, \beta^k > 0$.

(b) Since the scheme is safeguarded above by $\alpha_{\max}$, it suffices to show that the inequality

$$\alpha \leq \frac{2(f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k))}{\|\mathbf{d}^k\|^2} \tag{2}$$

holds for sufficiently small α. We claim that this inequality holds for any $\alpha \leq \frac{1}{\beta^k}$. To see this, recall (1), which gives for any $\alpha > 0$,

$$f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k) \geq \left(\alpha - \frac{\beta^k}{2}\alpha^2\right) \|\mathbf{d}^k\|^2.$$

For $\alpha \leq \frac{1}{\beta^k}$ we have $\beta^k \leq \frac{1}{\alpha}$, and thus

$$f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k) \geq \left(\alpha - \frac{1}{2\alpha}\alpha^2\right) \|\mathbf{d}^k\|^2 = \frac{1}{2}\alpha \|\mathbf{d}^k\|^2,$$

which immediately implies the inequality $\alpha \leq \frac{2(f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$.

(c) Based on the above definition, the $\bar{\alpha}$ generated by the scheme always satisfies the inequality (2). Combining (2) and the fact that $\bar{\alpha} \geq \frac{1}{2\beta^k}$ immediately tells that

$$f(\mathbf{x}^k + \alpha \mathbf{d}^k) - f(\mathbf{x}^k) \leq -\frac{1}{2}\bar{\alpha}\|\mathbf{d}^k\|^2 \leq -\frac{1}{4\beta^k}\|\mathbf{d}^k\|^2.$$

2. (10') (L_2 Regularization and Logarithmic Barrier) Consider the optimization problem

$$\begin{aligned} \text{minimize}_{x_1,x_2} \quad & (x_1 - x_2 + 1)^2 \\ \text{subject to} \quad & x_1 \geq 0 \quad x_2 \text{ "free".} \end{aligned}$$

Then we may combine the L_2-regularization and barrier together, that is, for any $\mu > 0$, consider

$$\text{minimize}_{x_1,x_2} \quad (x_1 - x_2 + 1)^2 + \frac{\mu}{2}(x_1^2 + x_2^2) - \mu \log(x_1)$$

(a) (4') Develop explicit path formula in terms of μ. What is the limit solution as $\mu \to 0$?

(b) (3') Using $\mu = 1$ and $\mathbf{x}^0 = (1,\ 0)$, apply one step of SDM with step-size 1/5 to compute the next iterate.

(c) (3') Using $\mu = 1$ and $\mathbf{x}^0 = (1,\ 0)$, apply one step of Newton's Method to compute the next iterate.

Solution:

(a) Let $f(\mathbf{x}; \mu) := (x_1 - x_2 + 1)^2 + \frac{\mu}{2}(x_1^2 + x_2^2) - \mu \log(x_1)$. Then the gradient of f is

$$\nabla_{\mathbf{x}} f(\mathbf{x}; \mu) = \begin{bmatrix} 2(x_1 - x_2 + 1) + \mu x_1 - \mu/x_1 \\ 2(x_2 - x_1 - 1) + \mu x_2 \end{bmatrix}.$$

Setting the gradient to 0 (KKT conditions), we obtain that $x_2 = \frac{2x_1+2}{2+\mu}$, and $x_1 = \frac{\mu+2}{\mu+4}$. And as a result, we obtain that the path formula of $\mathbf{x}(\mu)$ is

$$\mathbf{x}(\mu) = \begin{bmatrix} (\mu+2)/(\mu+4) \\ (4\mu+12)/(\mu^2+6\mu+8) \end{bmatrix}.$$

Taking $\mu \to 0$, we obtain that the limit solution is $x_1 = 1/2,\ x_2 = 3/2$.

(b) By definition of SDM, we have

$$\mathbf{x}^1 = \mathbf{x}^0 - \frac{1}{5}\nabla_{\mathbf{x}} f(\mathbf{x}^0; 1) = \begin{bmatrix} 1/5 \\ 4/5 \end{bmatrix}.$$

(c) By simple calculation, we have

$$\nabla^2_{\mathbf{x}} f(\mathbf{x}; \mu) = \begin{bmatrix} 2 + \mu + \mu/x_1^2 & -2 \\ -2 & 2 + \mu \end{bmatrix}.$$

And so with $\mathbf{x}^0 = (1, 0)$ and $\mu = 1$, we have

$$\nabla^2_{\mathbf{x}} f((1,0); 1) = \begin{bmatrix} 4 & -2 \\ -2 & 3 \end{bmatrix}.$$

By definition of Newton's method, we have

$$\mathbf{x}^1 = \mathbf{x}^0 - \nabla^2_{\mathbf{x}} f(\mathbf{x}^0; 1)^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}^0; 1) = \begin{bmatrix} 1/2 \\ 1 \end{bmatrix}.$$

3. (20') (L_2 Path-Following) Consider a convex function $f : R^n \to R$ in C^2 that is twice continuously differentiable. Assume that its value is bounded from below and that it has a minimizer. For any given positive parameter $\mu > 0$, consider the regulated minimization problem

$$\text{minimize } f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x}\|^2. \tag{3}$$

Prove the following claims:

(a) (2') Write down the first-order optimality condition of (3). Is it sufficient for $\mathbf{x}$ to be a minimizer?

(b) (5') The minimizer, denoted by $\mathbf{x}(\mu)$, of (3) is unique in μ.

(c) (5') $f(\mathbf{x}(\mu))$ is an increasing function of μ (i.e., $f(\mathbf{x}(\mu)) \geq f(\mathbf{x}(\mu'))$ if $\mu \geq \mu' > 0$), and $\|\mathbf{x}(\mu)\|$ is a decreasing function of μ.

(d) (5') As $\mu \to 0^+$ (i.e., μ decreases to 0), $\mathbf{x}(\mu)$ converges to the minimizer of $f(\mathbf{x})$ with the minimal Euclidean norm.

(e) (3') Consider the specific example

$$\text{minimize}_{x_1,x_2}\ (x_1 - x_2 - 1)^2,$$

where the optimal solution set is unbounded. Write out the explicit path formula of $\mathbf{x}(\mu) = (x_1(\mu), \ldots, x_n(\mu))$ in terms of μ. What is the limit solution as $\mu \to 0$?

Solution:

(a) The first-order optimality condition is $\nabla f(x) + \mu x = 0$. Yes, because the problem is convex, and hence KKT conditions are sufficient (with no other requirements).

(b) In the following, apart from proving the uniqueness (we only require you to prove this in the homework), we also prove that $\mathbf{x}(\mu)$ is continuous in μ, which is similar to part of the proof in (d) below. The uniqueness is simply due to strict (actually strong) convexity. To show continuity, notice that $\nabla f(x(\mu)) + \mu x(\mu) = 0$ for any $\mu > 0$. Consider any $\mu, \mu' > 0$. Then subtracting their optimality conditions, we get

$$-\mu x(\mu) + \mu' x(\mu') = \nabla f(x(\mu)) - \nabla f(x(\mu')).$$

Multiplying $x(\mu) - x(\mu')$ on both sides, we get

$$(x(\mu) - x(\mu'))^T(-\mu x(\mu) + \mu' x(\mu')) \geq (x(\mu) - x(\mu'))^T(\nabla f(x(\mu)) - \nabla f(x(\mu'))) \geq 0$$

where the inequality comes from convexity of $f(\cdot)$. Hence we have

$$\mu\|x(\mu) - x(\mu')\|^2 \leq (\mu' - \mu)(x(\mu) - x(\mu'))^T x(\mu').$$

Now we make use of the monotonicity *to be established* in (c) (which does not depend on the result in (b) here), and observe that if we fix $\mu > 0$, and assume that

$|\mu' - \mu| \le C$ for some $C > 0$, then the monotonicity of $\|x(\mu)\|$ ensures that $\|x(\mu')\|$ is bounded by some constant that may depend on μ (which is fixed now). Hence by taking the limit $\mu' \to \mu$, we obtain that the RHS above converges to 0 (using in addition that $x^T y \le \|x\|\|y\|$). Hence we conclude that $\|x(\mu) - x(\mu')\| \to 0$ as $\mu' \to \mu > 0$. Notice again that μ is fixed.

Finally, since μ is arbitrary, we have proved that $x(\mu)$ is continuous in μ.

(c) Let $0 < \mu' < \mu$. Then by the optimality of the solutions, we have

$$f(x(\mu')) + \mu'/2\|x(\mu')\|^2 \le f(x(\mu)) + \mu'/2\|x(\mu)\|^2$$

and

$$f(x(\mu)) + \mu/2\|x(\mu)\|^2 \le f(x(\mu')) + \mu/2\|x(\mu')\|^2.$$

Adding the above inequalities, we see that

$$\frac{\mu - \mu'}{2}\|x(\mu')\|^2 \ge \frac{\mu - \mu'}{2}\|x(\mu)\|^2.$$

Since $\mu - \mu' > 0$, we have $\|x(\mu')\|^2 \ge \|x(\mu)\|^2$, showing that $\|x(\mu)\|$ is a decreasing function of μ. Finally, using any one of the first inequality above, we see that $f(x(\mu')) \le f(x(\mu)) + \mu'/2(\|x(\mu)\|^2 - \|x(\mu')\|^2) \le f(x(\mu))$, and hence $f(x(\mu))$ is an increasing function of μ.

(d) Let x^* be an arbitrary minimizer of $f(x)$. Then we have $\nabla f(x^*) = 0$, and together with the fact that $\nabla f(x(\mu)) + \mu x(\mu) = 0$, we obtain that

$$\nabla f(x(\mu)) - \nabla f(x^*) + \mu x(\mu) = 0.$$

Similar to (b), multiplying both sides with $x(\mu) - x^*$, we have by convexity of f that

$$-\mu(x(\mu) - x^*)^T x(\mu) = (x(\mu) - x^*)^T \nabla f(x(\mu)) \ge 0.$$

Hence we have that $\|x(\mu)\|^2 \le x(\mu)^T x^* \le \|x^*\|\|x(\mu)\|$, which proves that $\|x(\mu)\| \le \|x^*\|$ (for any $\mu > 0$).

Now notice that to prove the claim in the question, we only need to show that for any sequence $\mu_k \to 0+$, $x(\mu_k)$ converges to a minimizer $\bar{x}$ of $f(x)$ with minimal Euclidean norm.

To see this, firstly notice that $\{x(\mu_k)\}_{k=1}^{\infty}$ is bounded, and hence it has a limit point, i.e. there exists $\bar{x}$ s.t. $\exists$ a subsequence k_i with $x(\mu_{k_i}) \to \bar{x}$. Moreover, for any limit point $\bar{x}'$ of $x(\mu_k)$, since f is C^2, $\nabla f(\cdot)$ is continuous, and hence we have $\nabla f(\bar{x}') = \lim_{i\to\infty} \nabla f(x(\mu_{l_i})) = -\lim_{i\to\infty} \mu_{l_i} x(\mu_{l_i}) = 0$ for some subsequence l_i. The last equality comes from the fact that $x(\mu_{l_i})$ is bounded and $\mu_{l_i} \to 0+$. Hence we see that any limit point of $x(\mu_k)$ is an optimal solution. In addition, it also comes with minimum Euclidean norm since $\|x(\mu)\| \le \|x^*\|$ for any minimizer x^*,

which implies that $\|\bar{x}'\| \leq \|x^*\|$ also holds and hence $\bar{x}'$ is a minimal Euclidean norm minimizer of $f(x)$.

Finally, it's easy to show that the minimal norm solution is unique. To see this, simply notice that if there are two minimal norm solutions $x_1 \neq x_2$, then $(x_1 + x_2)/2$ is also a minimizer of $f(x)$ but has a smaller norm (unless $x_1 = x_2$), which is a contradiction. Hence we see that any limit point of $x(\mu_k)$ is the same minimal norm minimizer $\bar{x}$, and since μ_k is an arbitrary sequence with $\mu_k \to 0+$, we conclude that as $\mu \to 0+$, $x(\mu)$ converges to the unique minimizer of $f(x)$ with the minimal Euclidean norm.

(e) By the first-order optimality conditions, we have $x_1 = -x_2 = \frac{2}{\mu+4}$. As $\mu \to 0$, the limit solution is $x_1 = -x_2 = \frac{1}{2}$.

4. (15') (Affine-Scaling Interior-Point SD) Consider the conic constrained optimization problem

$$\min_{x} \quad f(\mathbf{x}) \qquad \text{s.t.} \quad \mathbf{x} \geq \mathbf{0} \tag{4}$$

where we assume the objective function f is first-order β-Lipschitz. Starting from $\mathbf{x}^0 = \mathbf{e} > 0$, consider the affine-scaling interior-point method as follows: at iterate $\mathbf{x}^k > 0$ let diagonal scaling matrix D be

$$D_{ii} = \min\{1,\ x_i^k\}$$

and

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k D^2 \nabla f(\mathbf{x}^k),$$

with step-size

$$\alpha^k = \min\left\{\frac{1}{\beta},\ \frac{1}{2\|D\nabla f(\mathbf{x}^k)\|_\infty}\right\}. \tag{5}$$

(a) (3') Show that $D^2\nabla f(\mathbf{x}^k)$ is a descent direction.

(b) (3') Show that $\mathbf{x}^{k+1} > \mathbf{0}$ for all $k = 0, 1, ...$

(c) (6') Show that

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq \min\left\{-\frac{1}{2\beta}\|D\nabla f(\mathbf{x}^k)\|_\infty^2, -\frac{1}{4}\|D\nabla f(\mathbf{x}^k)\|_\infty\right\}$$

(d) (3') Derive a iterative complexity bound for $\|D\nabla f(\mathbf{x}^k)\|_\infty \leq \epsilon$.

Solution: Here we treat the gradient vector as a column vector.

(a) Denote $\mathbf{d}_k = -D^2\nabla f(\mathbf{x}^k)$. By definition of D and the fact that $\mathbf{x}^k > 0$ one clearly have $D \succ \mathbf{0}$, and thus $D^2 \succ \mathbf{0}$. Hence $\mathbf{d}_k^T \nabla f(\mathbf{x}^k) = -\nabla f(\mathbf{x}^k) D^2 \nabla f(\mathbf{x}^k) < 0$ if $\nabla f(\mathbf{x}^k) \neq 0$. Hence $\mathbf{d}^k$ is a descent direction if $\nabla f(\mathbf{x}^k) \neq 0$.

(b) We will prove the fact that $\mathbf{x}^k > \mathbf{0}$ for all $k = 0, 1, \ldots$ by induction. Clearly for $k = 0$ one have $\mathbf{x}^0 = \mathbf{e} > 0$ by assumption. Assume this fact holds for k, i.e., $\mathbf{x}^k > 0$, consider the $(k+1)$-th iteration. It follows that the i-th component of $\mathbf{x}^{k+1}$ satisfies

$$x_i^{k+1} = x_i^k - \alpha^k (D^2\nabla f(\mathbf{x}^k))_i \geq x_i^k - \alpha^k |(D^2\nabla f(\mathbf{x}^k))_i| = x_i^k - \alpha^k (D_{ii}) |(D\nabla f(\mathbf{x}^k))_i|$$

Since $D_{ii} = \min\{1, x_i^k\}$ one have $D_{ii} \leq x_i^k$, and thus

$$x_i^k - \alpha^k (D_{ii}) |(D\nabla f(\mathbf{x}^k))_i| \geq x_i^k - \alpha^k x_i^k |(D\nabla f(\mathbf{x}^k))_i| = x_i^k (1 - \alpha^k (D\nabla f(\mathbf{x}^k))_i|)$$

Note that $D_{ii}|\nabla f(\mathbf{x}^k)|_i \leq \|D\nabla f(\mathbf{x}^k)\|_\infty$, by definition of α^k we obtain $\alpha^k D_{ii}|\nabla f(\mathbf{x}^k)|_i \leq \frac{1}{2}$. Combining these results above yields $x_i^{k+1} \geq \frac{1}{2} x_i^k > 0$. This concludes the induction.

(c)(d) Since the function is first-order β-Lipschitz, for each step,

$$\begin{aligned} f(\mathbf{x}^{k+1}) &\leq f(\mathbf{x}^k) - (\alpha^k D^2 \nabla f(\mathbf{x}^k))^T \nabla f(\mathbf{x}^k) + \frac{\beta}{2}(\alpha^k)^2 \|D^2 \nabla f(\mathbf{x}^k)\|_2^2 \\ &= f(\mathbf{x}^k) - \alpha^k \|D \nabla f(\mathbf{x}^k)\|_2^2 + \frac{\beta}{2}(\alpha^k)^2 \|D^2 \nabla f(\mathbf{x}^k)\|_2^2 \end{aligned}$$

Since $D_{ii} \leq 1$ we have $\|D^2 \nabla f(\mathbf{x}^k)\|_2^2 \leq \|D \nabla f(\mathbf{x}^k)\|_2^2$, and therefore

$$\begin{aligned} f(\mathbf{x}^{k+1}) &\leq f(\mathbf{x}^k) - \alpha^k \|D \nabla f(\mathbf{x}^k)\|_2^2 + \frac{\beta}{2}(\alpha^k)^2 \|D \nabla f(\mathbf{x}^k)\|_2^2 \\ &= f(\mathbf{x}^k) - \left(\alpha^k - \frac{\beta}{2}(\alpha^k)^2\right) \|D \nabla f(\mathbf{x}^k)\|_2^2 \end{aligned} \tag{6}$$

Note that $\|D \nabla f(\mathbf{x}^k)\|_2^2 \geq \|D \nabla f(\mathbf{x}^k)\|_\infty^2$. According to the scheme, the inequality $0 \leq \alpha^k \leq \frac{1}{\beta}$ always holds, which implies $\alpha^k - \frac{\beta}{2}(\alpha^k)^2 \in [0, \frac{1}{2\beta}]$. Therefore

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \left(\alpha^k - \frac{\beta}{2}(\alpha^k)^2\right) \|D \nabla f(\mathbf{x}^k)\|_\infty^2 \tag{7}$$

According to the step-size scheme (5), there are two cases:

- Case I: $\alpha^k = \frac{1}{\beta} \leq \frac{1}{2\|D \nabla f(\mathbf{x}^k)\|_\infty}$. In this case (according to (7)),

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \frac{1}{2\beta} \|D \nabla f(\mathbf{x}^k)\|_\infty^2 \tag{8}$$

- Case II: $\alpha^k = \frac{1}{2\|D \nabla f(\mathbf{x}^k)\|_\infty} \leq \frac{1}{\beta}$. In this case (according to (7)),

$$\begin{aligned} f(\mathbf{x}^{k+1}) &\leq f(\mathbf{x}^k) - \left(1 - \frac{\beta}{2}\alpha^k\right) \frac{1}{2\|D \nabla f(\mathbf{x}^k)\|_\infty} \|D \nabla f(\mathbf{x}^k)\|_\infty^2 \\ &= f(\mathbf{x}^k) - \frac{1}{2}\left(1 - \frac{\beta}{2}\alpha^k\right) \|D \nabla f(\mathbf{x}^k)\|_\infty \\ &\leq f(\mathbf{x}^k) - \frac{1}{4} \|D \nabla f(\mathbf{x}^k)\|_\infty \end{aligned} \tag{9}$$

  where in the last inequality we used the fact that $\alpha^k \leq \frac{1}{\beta}$.

Combining the two cases immediately tells that the method will identify an $\mathbf{x}^k$ such that $\|D \nabla f(\mathbf{x}^k)\|_\infty \leq \varepsilon$ within $\max\left\{\frac{4(f(\mathbf{x}^0) - f^*)}{\varepsilon}, \frac{2\beta(f(\mathbf{x}^0) - f^*)}{\varepsilon^2}\right\}$ steps.

Computational Homework:

5. (10') There is a simple nonlinear least squares approach for Sensor Network Localization:

$$\min \quad \sum_{(ij)\in N_x} \left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2\right)^2 + \sum_{(kj)\in N_a} \left(\|\mathbf{a}_k - \mathbf{x}_j\|^2 - d_{kj}^2\right)^2 \qquad (10)$$

which is an unconstrained nonlinear minimization problem.

(a) (5') Apply the Steepest Descent Method, starting with either the origin or a random solution as the initial solution for model (10), to solve few selected SNL instances you created in HW1. Does it work?

(b) (5') Apply the same Steepest Descent Method, starting from the SOCP or SDP solution (which may not have errors) as the initial solution for model (10), to solve the same instances in (a). Does it work? Does the SOCP or SDP initial solution make a difference?

Solution:

(a) [1] For the data from HW2, we obtain the following result:

Case#	Iteration#	$\|g\|$	$\|x - x^*\|$
0	95	7.02937300001e-07	1.31526462014
1	96	8.49323630054e-07	0.4132505402
2	66	7.91874506287e-07	2.06534769155
3	46	7.36718181322e-07	0.41958131873
4	211	8.77641007091e-07	0.932793325729
5	2692	9.97830126784e-07	0.176389149702
6	91	7.82269895272e-07	1.23784084017
7	227	9.45751060917e-07	1.18611529159
8	61	7.82206062276e-07	1.21552072733
9	308	9.77360601169e-07	1.4142135577e-08
10	41	6.78613735484e-07	0.175096086606
11	209	9.99876636385e-07	0.806543186885
12	357	9.99499535795e-07	0.575181360423
13	1039	9.76516842978e-07	0.219059563814
14	104	8.72745816205e-07	0.849779558494
15	293	9.5217523952e-07	6.32455531876e-08
16	24	5.27755778653e-07	0.982261236774
17	2965	9.95270899503e-07	2.23606798625e-08

[1] Adapted from student solutions last year. Thanks to Kailai Xu and Vivienne Liu.

18	65	7.77092719272e-07	3.51732255203
19	27	9.54216094557e-07	0.815901373748
20	378	9.87688019158e-07	1.99999999895e-08
21	195	8.62342659222e-07	0.724194649238
22	1552	9.91882305809e-07	2.99999999287e-08
23	285	9.97067679627e-07	1.12545484854
24	89	9.75161710255e-07	1.59704634344
25	147	9.72828975705e-07	0.740145519253
26	99	8.13831949701e-07	1.50224641879
27	309	9.79323149256e-07	1.02081475561
28	226	9.2885990418e-07	0.721344277403
29	654	9.79829401167e-07	5.99999999684e-08
30	988	9.94532217468e-07	0.109547519934
31	364	9.68201976257e-07	0.590655437691
32	510	9.93023652421e-07	0.490007256859
33	77	9.07378070571e-07	0.61726817244
34	2134	9.95071239091e-07	0.0630439420689
35	2390	9.92962934133e-07	0.309758188886
36	163	9.82627021908e-07	0.0
37	516	9.84861710495e-07	4.12310563018e-08
38	204	9.48124819317e-07	0.282016214286
39	215	9.18247954374e-07	0.880287444236
40	190	9.99061485568e-07	0.153819671152
41	519	9.54957321284e-07	0.246353191122
42	53	6.69096706371e-07	1.02024518653
43	98	8.84224787798e-07	5.00000000292e-08
44	66	8.19316713216e-07	1.1314524184
45	160	9.58536768051e-07	0.449272985057
46	666	9.74917392674e-07	3.16227766509e-08
47	181	9.27183274439e-07	0.323240948876
48	67	9.77719512522e-07	1.22810067024
49	53	8.86323005847e-07	9.99999993923e-09

We can see from the table:

- **The algorithm do converge for all the cases.**
- **The average iteration number is larger in the latter case (HW2).**
- **However, the algorithm do not necessarily converge to the real location. Only in a few cases(those in RED) turn out to be exact. This depends on the locality of the original problem.**

(b) Starting from the solutions of SOCP/SDP relaxation obtained from CVX as we

did in HW2, we see that if SOCP/SDP has already (almost) exactly recovered the true solution, then the SDM applied to the unconstrained problem here does improve the accuracy further. Also, starting from SDP tends to give more accurate results (and also more cases where we can refine the solution, because SDP exact recovery is more common than SOCP, as we have seen in HW2 problem 9). However, if SOCP/SDP already fails miserably, then SDM also doesn't help at all. This is expected as we first used a more accurate method (CVX solvers, which are typically second-order algorithms) and then try to refine it with a first-order solver SDM. The code is listed in Appendix.

Remark. We will accept any reasonable solutions showing sufficient efforts, and the results above are just for references.

6. (30') (Multi-Block ADMM)

Part I Implement the ADMM to solve the divergence example:

$$\begin{aligned} \text{minimize} \quad & 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 \\ \text{subjectto} \quad & \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{0} \end{aligned}$$

(a) (5') Try $\beta = 0.1$, $\beta = 1$, and $\beta = 10$, respectively. Does the choice of β make a difference?

(b) (5') Add the objective function to minimize

$$0.5(x_1^2 + x_2^2 + x_3^2)$$

to the problem, and retry $\beta = 0.1$, $\beta = 1$, and $\beta = 10$, respectively. Does the choice of β make a difference?

(c) (5') Set $\beta = 1$ and apply the randomly permuted updating-order of $\mathbf{x}$ (discussed in class) to solving each of the two problems in (a) and (b). Does the iterate converge?

Part II Generate some (feasible) convex QP problems with linear equality constraints, say 30 variables and 10 constraints (i.e., $A \in R^{10\times 30}$),

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\mathbf{x}^T Q \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

(d) (5') Divide the variables of $\mathbf{x}$ into 5 blocks and apply the ADMM with $\beta = 1$. Does it converge? (You may construct 5 different blocks and conduct the experiments.)

(e) (5') Apply the randomly permuted updating-order of the 5 blocks in each iteration of the ADMM. Does it converge? Convergence performance?

(f) (5') Consider the following scheme – random-sample-without-replacement: in each iteration of ADMM, randomly sample 6 variables for update, and then randomly select 6 variables from the remaining 24 variable for update, and... , till all 30 variables are updated; then update the multipliers as usual. Does it converge? Convergence performance?

Solution: First of all, we would like to comment that since for each sub-step, the minimization problem is a convex QP, and hence we can simply solve it by taking the gradient and setting it to 0, which reduces to the problem to solving a linear system. Hence in this problem, either you apply a gradient step to the sub-problem or solve it directly, you never need to use some other general optimization solvers.

(a) We Implement the ADMM to solve the divergence example in Lecture 15. As shown in figure 1, the choice of β doesn't really make a difference. For all three choices of β, the procedure diverges, especially they diverge in a very similar geometric rate.

```
rng(1); A = [1,1,1;1,1,2;1,2,2]; LA = tril(A^2); UA = LA-A^2;
figure();
for beta = [0.1,1,10]
    M_lhs = [LA*beta, zeros(3,3);A*beta,eye(3)]; M_rhs = [UA*beta, A; ...
        zeros(3,3),eye(3)];
    M = inv(M_lhs)*M_rhs; x = rand(3,1); y = rand(3,1); xnorm = [];
    for time = 1:1000
        xynew = M*[x;y]; x = xynew(1:3); y = xynew(4:6); xnorm = [xnorm, norm(x)];
    end
    semilogy(xnorm); hold on
end
legend('\beta = 0.1','\beta = 1','\beta = 10');
xlabel('Iterations'); ylabel('x Norm'); ylabel('x Norm');
t = title('ADMM: divergence example in Lecture 15');
```

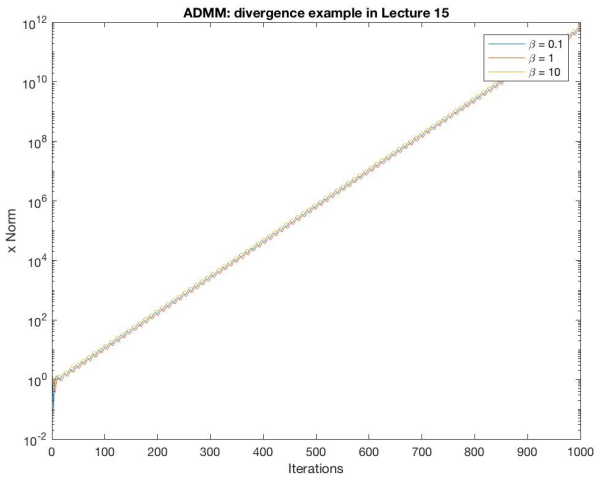

Figure 1: ADMM (a): the divergence example in Lecture 15

(b) After we add $0.5(x_1^2 + x_2^2 + x_3^2)$ to the objective function, the choice of β makes a difference. In figure 2, it appears that the procedure diverges for $\beta = 10$, and converges for $\beta = 0.1$ and $\beta = 1$. This time the procedure can converge for some β and hence, to a certain extent, better than the previous one in (a). It is because the objective function becomes strictly convex. The step size corresponds to $\beta = 10$ is too large so that the procedure does not converge. At the same time, $\beta = 0.1$ corresponds to a much smaller step size, making the convergence rate not as good as the $\beta = 1$ one. Still, both cases converge geometrically.

```
rng(1); A = [1,1,1;1,1,2;1,2,2]; LA = tril(A^2); UA = LA-A^2;
figure();
for beta = [0.1,1,10]
    M_lhs = [LA*beta + eye(3), zeros(3,3);A*beta,eye(3)]; M_rhs = [UA*beta, A; ...
        zeros(3,3),eye(3)];
    M = inv(M_lhs)*M_rhs; x = rand(3,1); y = rand(3,1); xnorm = [];
    for time = 1:1000
        xynew = M*[x;y]; x = xynew(1:3); y = xynew(4:6); xnorm = [xnorm, norm(x)];
    end
    semilogy(xnorm); hold on
end
legend('\beta = 0.1','\beta = 1', '\beta = 10'); xlabel('Iterations'); ylabel('x ...
    Norm');
t = title('ADMM: Adding 0.5(x_1^2 + x_2^2 + x_3^2) to objective');
```

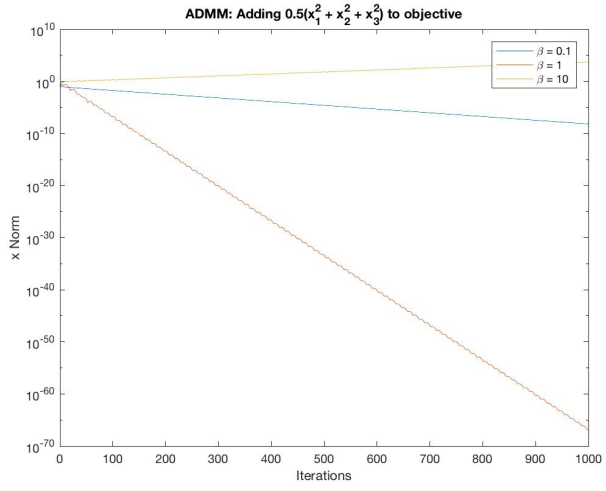

Figure 2: ADMM (b): add $0.5(x_1^2 + x_2^2 + x_3^2)$ to objective

(c) See figure 3 and figure 4 After randomly permuting the updating-order of x, for both problems in (a) and (b), the iterates converge. The one corresponds to (a) converges slower than the that to (b), as the problem is more "convex" for (b). Both converge in roughly a geometric rate.

```
A = [1,1,1;1,1,2;1,2,2]; beta = 1;
x = rand(3,1); y = rand(3,1); xnorm = [];

for time = 1:1000
    r_index = randperm(3);
    A_2 = A^2; LA = tril(A_2(r_index,r_index));
    [t,r_rank] = sort(r_index);
    LA = LA(r_rank, r_rank); UA = LA-A^2;
    M_lhs = [LA*beta, zeros(3,3);A*beta,eye(3)];
    M_rhs = [UA*beta, A; zeros(3,3),eye(3)];
    M = inv(M_lhs)*M_rhs;
    xynew = M*[x;y]; x = xynew(1:3); y = xynew(4:6);
    xnorm = [xnorm, norm(x)];
end
figure();
semilogy(xnorm); legend('\beta = 1'); xlabel('Iterations'); ylabel('x Norm');
t = title('ADMM: randomly permuted updating-order - (a)');

A = [1,1,1;1,1,2;1,2,2]; beta = 1;
x = rand(3,1); y = rand(3,1); xnorm = [];
for time = 1:1000
    r_index = randperm(3);
    A_2 = A^2; LA = tril(A_2(r_index,r_index));
    [t,r_rank] = sort(r_index);
    LA = LA(r_rank, r_rank); UA = LA-A^2; LA = LA+eye(3);
    M_lhs = [LA*beta, zeros(3,3);A*beta,eye(3)];
    M_rhs = [UA*beta, A; zeros(3,3),eye(3)];
    M = inv(M_lhs)*M_rhs;
    xynew = M*[x;y]; x = xynew(1:3); y = xynew(4:6);
    xnorm = [xnorm, norm(x)];
end
figure();
semilogy(xnorm); legend('\beta = 1'); xlabel('Iterations'); ylabel('x Norm');
t = title('ADMM: randomly permuted updating-order - (b)');
```

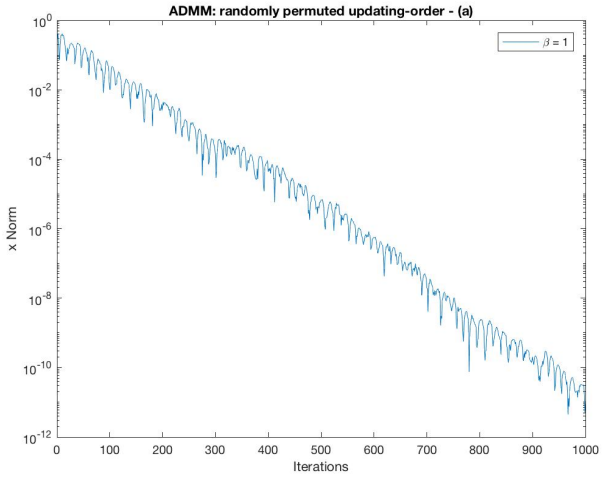

Figure 3: ADMM (c): randomly permuting the updating-order

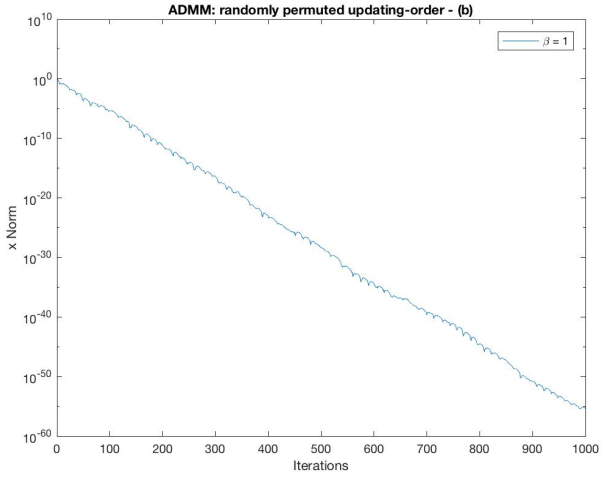

Figure 4: ADMM (c): randomly permuting the updating-order

(d)-(f) In figure 5, we consider three procedures. We'll refer to them as procedure 1,2,3 respectively.

1. Divide the variables of x into 5 blocks and apply the ADMM with $\beta = 1$. The procedure does converge.

2. Apply the randomly permuted updating-order of the 5 blocks in each iteration of the ADMM. The procedure also does converge.

3. new scheme random-sample-without-replacement: in each iteration of ADMM, randomly sample 6 variables for update, and then randomly select 6 variables from the remaining 24 variable for update, and... , till all 30 variables are updated; then update the multipliers as usual. The procedure also converges.

All of the three converge in geometric rate. The new scheme-random-sample-without-replacement appears to converge the fastest, followed by no permutation .

Note that the yellow line doesn't appear to move after 350 iterations, this may be due to that the "true value" of x we used is not perfectly accurate.

```
%%construct the problem
A = rand(10,30); Q_half = rand(30,30);
x_0 = rand(30,1); b = A*x_0; Q = Q_half'*Q_half; LU = Q + A'*A;
%%use cvx to find the true value
cvx_begin quiet
    variable x_true(30)
    minimize( x_true'*Q*x_true )
    subject to
        A*x_true == b
cvx_end

a=ones(6,6); n=5;
AA=repmat(a,n,1); BB=mat2cell(AA,6*ones(1,n),6);
Diagonal_B=blkdiag(BB{:}); lower_B = tril(ones(30,30)) + triu(Diagonal_B,1);

%%Without random permutation
beta = 1; x = rand(30,1); y = rand(10,1); xnorm1 = [];
for time = 1:1000
    LA = lower_B.*LU; UA = LA-LU;
    M_lhs = [LA, zeros(30,10);A,eye(10)];
    M_rhs = [UA, A'; zeros(10,30),eye(10)];
    M_b = [A'*b; b];
    xynew = inv(M_lhs)*(M_rhs*[x;y]+M_b); x = xynew(1:30); y = xynew(31:40);
    xnorm1 = [xnorm1, norm(x-x_true)];
end

%%With random permutation of block
x = rand(30,1); y = rand(10,1); xnorm2 = [];
for time = 1:1000
    group_index = randperm(5);
    r_index = reshape(6*repmat(group_index,6,1) + repmat((0:5)',1,5)-5,1,30);

    LA = lower_B.*(LU(r_index,r_index)); [t,r_rank] = sort(r_index);
    LA = LA(r_rank, r_rank); UA = LA-LU;

    M_lhs = [LA, zeros(30,10);A,eye(10)];
```

```
    M_rhs = [UA, A'; zeros(10,30),eye(10)];
    M_b = [A'*b; b];

    xynew = inv(M_lhs)*(M_rhs*[x;y]+M_b); x = xynew(1:30); y = xynew(31:40);
    xnorm2 = [xnorm2, norm(x-x_true)];
end

%%new scheme random-sample-without-replacement

x = rand(30,1);y = rand(10,1);xnorm3 = [];

for time = 1:1000
    r_index = randperm(30);

    LA = lower_B.*(LU(r_index,r_index));
    [t,r_rank] = sort(r_index); LA = LA(r_rank, r_rank); UA = LA-LU;

    M_lhs = [LA, zeros(30,10);A,eye(10)];
    M_rhs = [UA, A'; zeros(10,30),eye(10)];
    M_b = [A'*b; b];

    xynew = inv(M_lhs)*(M_rhs*[x;y]+M_b); x = xynew(1:30); y = xynew(31:40);
    xnorm3 = [xnorm3, norm(x-x_true)];
end

%%plot
figure(); semilogy(xnorm1); hold on
semilogy(xnorm2); hold on
semilogy(xnorm3);

legend('No random permutation', 'Random permutation of blocks', 'New scheme: ...
    random-sample-without-replacement');
xlabel('Iterations'); ylabel('||x-x_0||');
t = title('ADMM: With and without random permutations');
```

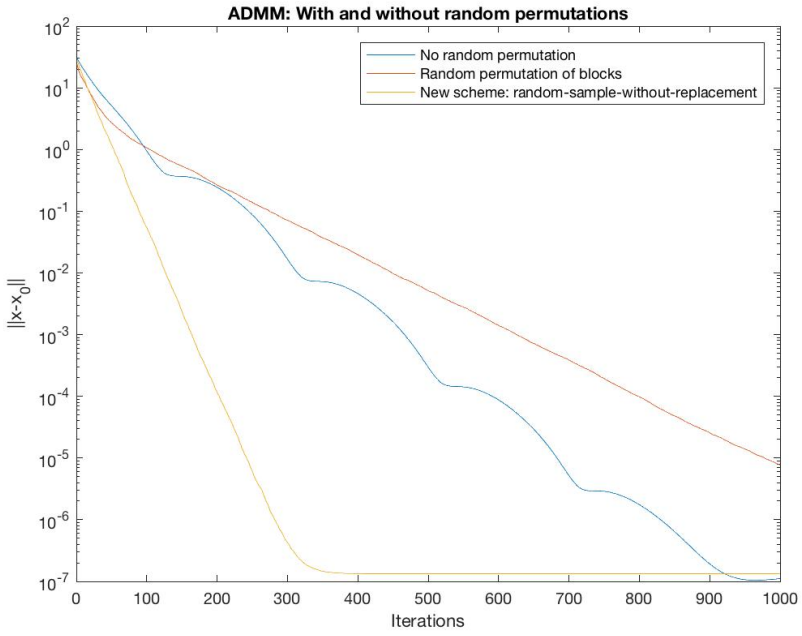

Figure 5: ADMM: Comparison of convergence speed