Convex Optimization for Machine Learning and Computer Vision

Lecture: Dr. Tao Wu
Exercises: Emanuel Laude, Zhenzhang Ye
Summer Semester 2018

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 8

Room: 02.09.023
Wednesday, 20.06.2018, 12:15-14:00
Submission deadline: Monday, 18.06.2018, 16:15, Room 02.09.023

Convergence Analysis (10+5 Points)

Exercise 1 (6 Points)**.** We want to use averaged operator to show proximal operator converges to a fixed point. So given a convex function $J : \mathbb{R}^n \to \mathbb{R}$, prove following properties:

1. The subdifferential ∂J is a monotone operator.
2. Show that resolvent operator is nonexpansive i.e. show $(I + \lambda R)^{-1}$ is nonexpansive where R is a monotone operator.
3. Show that proximal operator on a convex function is averaged.
   Hint: we define Cayley operator as $C_{\lambda R} = 2(I + \lambda R)^{-1} - I$. Try to express resolvent by Cayley operator.

Solution. 1. Since J is convex, for two different points $u, v \in \mathrm{dom}(J)$, we have following two equations:

$$\partial J(v)(u - v) \leq J(u) - J(v)$$

$$\partial J(u)(v - u) \leq J(v) - J(u)$$

Summing them up, we get $\langle \partial J(u - v), u - v \rangle \geq 0$.

2. We denote that $u = (I + \lambda R)^{-1}x$, $v = (I + \lambda R)^{-1}y$, by definition we have

$$u + \lambda Ru \ni x, \quad v + \lambda Rv \ni y$$

Subtracting these yields

$$u - v + \lambda(Ru - Rv) \ni x - y$$

Taking inner product with $u - v$ and R is monotone, we have:

$$\begin{aligned}
&\langle u - v, u - v \rangle + \lambda \langle u - v, Ru - Rv \rangle = \langle u - v, x - y \rangle \\
&\|u - v\|^2 \leq \langle u - v, x - y \rangle \leq \|u - v\| \, \|x - y\| \\
&\|u - v\| \leq \|x - y\| \\
&\left\|(I + \lambda R)^{-1}x - (I + \lambda R)^{-1}y\right\| \leq \|x - y\|
\end{aligned}$$

3. $(I+\lambda R)^{-1} = \frac{1}{2}I + \frac{1}{2}C_{\lambda R}$. Now we just need to show $C_{\lambda R}$ is nonexpansive. We denote that $u = (I+\lambda R)^{-1}x$, $v = (I+\lambda R)^{-1}y$.

$$\begin{aligned}\|C_{\lambda R}x - C_{\lambda R}y\|^2 &= \|2(u-v)-(x-y)\|^2 \\ &= 4\|u-v\|^2 - 4\langle x-y, u-v\rangle + \|x-y\|^2 \\ &\leq 4\langle x-y, u-v\rangle - 4\langle x-y, u-v\rangle + \|x-y\|^2 \\ &= \|x-y\|^2\end{aligned}$$

Consider following problem:

$$\min_{u\in\mathbb{R}^{n\times k}} \langle u, f\rangle + \sum_{i=1}^{n} \delta\{u(i,:) \in \triangle_{k-1}\} + \sum_{j=1}^{k} ||Du(:,j)||_1$$

where $f \in \mathbb{R}^{n\times k}$ and $u(i,:)$ is the i-th row of u, $u(:,j)$ is the j-th column. The matrix $D : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator. $\triangle_{k-1}$ is the simplex.

Exercise 2 (9 Points)**.** Write down the explicit update equations for solving above problem with

- PDHG
- ADMM
- Douglas-Rachford Splitting

Hint: for Douglas-Rachford Splitting, you should consider the convex conjugate for the sub problem and inner loop is allowed.

Solution.

- PDHG solution:

$$\begin{aligned}u(i,:)^{k+1} &= \text{proj}_{\triangle_{k-1}}(u(i,:)^k - s(D^\top p(i,:)^k + f(i,:)), \\ p(i,:)^{k+1} &= \text{proj}_{[-1,1]^n}(p(i,:)^k + tD(2u(i,:)^{k+1} - u(i,:)^k)).\end{aligned}$$

- The update of ADMM differs if we replaces u with different numbers of v_i. There are several options:
  1) using v to replace u in $\sum_{j=1}^{k} ||Du(:,j)||_1$, which leads to $\sum_{j=1}^{k} ||Dv(:,j)||_1$. In this case, we achieve DRS alogrithm.
  2) replace $\langle u, f\rangle + \sum_{i=1}^{n} \delta\{u(i,:) \in \triangle_{k-1}\}$ with v_1 and $\sum_{j=1}^{k} ||Du(:,j)||_1$ into $\sum_{j=1}^{k} ||v_2(:,j)||$ with v_2. Here, we consider the second strategy, the Augmented Lagrangian is:

$$\begin{aligned}\mathcal{L}_\tau =& \langle v_1, f\rangle + \sum_{i=1}^{n} \delta\{v_1(i,:) \in \triangle_{k-1}\} + \sum_{j=1}^{k} ||v_2(:,j)|| + \langle p_1, u - v_1\rangle \\ &+ \frac{\tau}{2}\|u-v_1\|^2 + \langle p_2, Du - v_2\rangle + \frac{\tau}{2}\|Du - v_2\|^2\end{aligned}$$

$$
\begin{aligned}
u^{k+1}(:,j) &= (I + D^\top D)^{-1}(v_1^k(:,j) + D^\top v_2^k(:,j) - \frac{1}{\tau}(p_1^k(:,j) + D^\top p_2^k(:,j)) \\
v_1^{k+1}(i,:) &= \mathrm{proj}_{\triangle_{k-1}}(u^{k+1}(i,:) + \frac{p_1^k(i,:) - f(i,:)}{\tau}) \\
v_2^{k+1}(:,j) &= \mathrm{prox}_{\frac{1}{\tau}\|\cdot\|_1}(Du^{k+1}(:,j) + \frac{1}{\tau}p_2^k(:,j)) \\
p_1^{k+1} &= p_1^k + \tau(u^{k+1} - v_1^{k+1}) \\
p_2^{k+1} &= p_2^k + \tau(Du^{k+1} - v_2^{k+1})
\end{aligned}
$$

- The update of DRS is following:

$$
\begin{aligned}
u(i,:)^{k+1} &= \mathrm{prox}_{\tau G}(v(i,:)^k) \\
&= \mathrm{argmin}_u \frac{1}{2\tau}\left\|u(i,:) - v^k\right\|^2 + \langle u(i,:), f(i,:)\rangle + \delta\{u(i,:) \in \triangle_{k-1}\} \\
&= \mathrm{prox}_{\tau\triangle_{k-1}}(v(i,:)^k - \frac{1}{\tau}f(i,:)) \\
v(i,:)^{k+1} &= v(i,:)^k - 2u(i,:)^{k+1} + 2\mathrm{prox}_{\tau F}(2u(i,:)^{k+1} - v(i,:)^k)
\end{aligned}
$$

  We denote $\hat{v} = 2u(i,:)^{k+1} - v(i,:)^k$. Updating $v(i,:)$ requires solving a TV problem:

$$
\mathrm{prox}_{\tau F}(\hat{v}) = \mathrm{argmin}_v \frac{1}{2\tau}\|v - \hat{v}\|^2 + \|D\hat{v}\|_1
$$

  We can compute the dual problem of it, which is:

$$
\mathrm{argmin}_p \frac{\tau}{2}\left\|D^\top p\right\|^2 - \langle \hat{v}, D^\top p\rangle + \delta_{\|\|_\infty<1}(p)
$$

  It can be solved by projected gradient descent efficiently.

Semi-supervised learning (Due: 25.06.2018) (12 Points)

Exercise 3. In this programming exercise, you are asked to solve following problem: give you n points. You already know that these points can be classfied into k classes. Parts of these points are already labeled. Your task is to classify the rest unlabeled points. In fact this task can be formulated to following convex problem:

$$\min_{u \in \mathbb{R}^{n \times k}} \langle u, f \rangle + \sum_{i=1}^{n} \delta\{u(i,:) \in \triangle_{k-1}\} + \sum_{j=1}^{k} ||Du(:,j)||_1$$

where $f \in \mathbb{R}^{n \times k}$ is a matrixes represents the labeled points. D is a linear operator constructed by considering the distance of one point to its neighbour points. In case you need to do projection into a simplex, the function *projSimplex* is given.
Your tasks are following:

1. figure out how f should be and construct f.
2. Implement EITHER **ADMM** OR **PDHG**. Use your solution from previous exercise.
3. Think about the other algorithm you do NOT implement, which one would be faster considering running time.

(Reading following is optional). To help you understand the energy function, firstly we explain how D is constructed (D is already given in the code, you do NOT need to implement it).
Assume we consider d neighbour points. Thus, for each point, we get its d closest points (just like in an image, one pixel is connected to four neighbouring pixels). Therefore, one point is connected to d points which can be regarded as d edges. There is a weight on one edge which is reciprocal of the distance (in an image, each edge is weighted as same).
Considering i-th row of f, it represents the label of current point i. If it's known, $\langle u(i,:), f(i,:) \rangle$ should achieves minimal when $u(i,:) = f(i,:)$. Otherwise, $f(i,:)$ should affect nothing.
The indicator function here forces each row of u should be in a simplex. That can be considered as a probability distribution of one point for different classes.