Software Implementation Project for "Convex Optimization"

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

November 21, 2021

Consider the composite optimization problem

$$\min_x \quad f(x) + h(x), \tag{1}$$

where $f(x)$ is differentiable and $h(x)$ is a function whose proximal operator is easily available. Both $f(x)$ and $h(x)$ may be nonconvex. Let $h(x)$ be a proper and close function, and $\inf_{x\in \mathrm{dom} h} h(x) > -\infty$. The proximal operator of $h(x)$ is defined as

$$\mathrm{prox}_h(x) = \arg\min_x h(u) + \frac{1}{2}\|u - x\|^2.$$

1 Proximal Gradient Method for Composite Program

The proximal gradient method to solve (1) is performed as

$$x^{k+1} = \mathrm{prox}_{t_k h}(x^k - t_k \nabla f(x^k)),$$

where t_k is a chosen step size. Your tasks are listed as follows:

1. Download our optimization package "OptSuite" from PKU Disk.
   Read the builtin implementation of the solver module `Composite::Solver:ProxGBB`.

2. Find more scenarios of $f(x)$, and calculate their gradient and implement these functions in the `Base` module of OptSuite. **Your code should support at least two of them, see also tasks 5 and 6. The least squares has already been implemented**.

   - Least squares: $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ or $f(x) = \frac{1}{2}\|Ax - b\|_F^2$.
   - Variant of Huber loss:
   $$(f(x))_i = \begin{cases} \frac{1}{2\delta}x_i^2, & |x_i| < \delta, \\ |x_i| - \frac{\delta}{2}, & \text{otherwise}, \end{cases}$$
   where $\delta > 0$ is a positive parameter.
   - Logistic regression:
   $$f(x) = \frac{1}{m}\sum_{i=1}^{m} \log(1 + \exp(-b_i a_i^T x)).$$

Of course, the choices of $f(x)$ also depends on the selection of $h(x)$. Other functions not in the list are also welcome.

3. A few typical scenarios of $h(x)$ are listed as follows. **Your code should support at least three of them, see also tasks 5 and 6**. Note that x can either be a vector or a matrix. Implement the corresponding proximal operators of h in the `Base` module of OptSuite. For some $h(x)$ a closed-form solution may not be available. In this case, a simple algorithm for computing $\mathrm{prox}_h(x)$ should be implemented.

   - General functions
     - vectors: ℓ_0-norm
     - vectors: sum-of-norms (for group lasso) $\sum_{g\in\mathcal{G}} \|x_g\|_2$
     - elastic-net: $\|x\|_1 + (\lambda/2)\|x\|_2^2$
     - log-barrier and its variant: $-\sum_{i=1}^n \log x_i$ and $-\sum_{i=1}^n \log x_i - (\lambda/2)\|x\|_2^2$
     - vectors: ReLU function: $\sum_{i=1}^n \max(0, x_i)$
     - vectors: quadratic function $(1/2)x^TAx + b^Tx + c$, where A is positive semidefinite
     - maximal function: $\max_i x_i$
     - conjugate of a proper closed function: f^*
     - 1D, 2D TV-norm: $\sum_t |x_t - x_{t-1}|$ and $\sum_{t,s} |x_{t,s} - x_{t-1,s}| + |x_{t,s} - x_{t,s-1}|$
   - Indicator functions $1_C(x)$, where C can be
     - vectors: ℓ_0, ℓ_1, ℓ_2, ℓ_∞-ball
     - vectors: simple box $l \le x \le u$ (including the special cases: nonnegative and nonpositive orthant)
     - vectors: binary set $\{x \mid x_i \in \{l_i, u_i\}\}$
     - vectors: affine set $\{x \mid Ax = b\}$
     - vectors: half space $\{x \mid a^Tx \le b\}$
     - vectors: probability simplex $\{x \mid 1^Tx = 1,\ x \ge 0\}$
     - vectors: second-order cone $\{(x,t) \in \mathbb{R}^{n+1} \mid \|x\|_2 \le t\}$
     - matrices: nuclear norm ball $\|X\|_* \le r$, spectral norm ball $\|X\|_2 \le r$
     - matrices: Stiefel manifold $\{X \in \mathbb{R}^{n\times p} \mid X^TX = I\}$
     - matrices: positive semidefinite cone $\{X \mid X \succeq 0\}$
     - matrices: rank constraints $\mathrm{rank}(X) \le r$
     - matrices: convex spectral set $\{X \succeq 0 \mid \mathrm{tr}(X) = 1\}$

4. **(Optional)** Implement the following accelerated proximal gradient method based on the solver `ProxGBB`: FISTA, Nesterov's 2nd method. Your implementation should also include some strategies to choose the step size.

5. Consider the LASSO problem

$$\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1.$$

   - Perform numerical experiments on the following datasets using the standard (**optional:** and the accelerated) proximal gradient method:
     - Data generated via DCT. See section 4.1.1 of
       https://link.springer.com/content/pdf/10.1007/s10915-017-0624-3.pdf

  - Data generated from LIBSVM. See section 4.1 of
    https://epubs.siam.org/doi/pdf/10.1137/16M1097572

- Replace $\|x\|_1$ with other types of regularization terms that preserves sparsity and repeat the numerical tests. You can refer to this article for a list of regularization. Compare the results with the ℓ_1 case.

6. Consider the logistic regression problem

$$\min_x \quad \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-b_i a_i^T x)) + \lambda\|x\|,$$

where $\|\cdot\|$ can be either $\|\cdot\|_1$ or $\|\cdot\|_2$. Perform numerical experiments on the binary classification problems of the LIBSVM dataset. See
https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

7. **(Optional)** Consider the matrix completion problem with rank-constraints

$$\min_x \quad \frac{1}{2}\|\mathcal{P}_\Omega(X-M)\|_F^2, \quad \text{s.t.} \quad \operatorname{rank}(X) \le r,$$

where $\Omega \subset [n]\times[n]$ denotes the observation set, and $\mathcal{P}_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is defined as

$$(\mathcal{P}_\Omega(X))_{ij} = \begin{cases} x_{ij}, & (i,j)\in\Omega, \\ 0, & \text{otherwise.} \end{cases}$$

Perform numerical experiments on the following dataset:

- Synthetic data. First, generate random Gaussian matrices $U_0 \in \mathbb{R}^{m\times r_0}$ and $V_0 \in \mathbb{R}^{n\times r_0}$, and set $X_0 = U_0 V_0^T$. Second, randomly choose pmn indices ($0 < p < 1$) to form Ω, and populate the observation matrix M by letting $M_{ij} = (X_0)_{ij} + e_{ij}$, where e_{ij} is Gaussian noise with variance σ^2. Choose different m, n, r_0, p, σ and test with your proximal gradient algorithms. Required ranges: $m, n \in [500, 10000]$, $r_0 \in [10, 50]$, $p \in [0.02, 0.5]$, $\sigma \in [0, 0.2]$.
- Real data: jester and movie-len. Downloadable from here.

Since the true r_0 is never known in reality, some strategies are needed to dynamically adjust the target rank r.

8. **(Optional)** Consider the balanced wavelet model

$$\min_x \quad \lambda\|x\|_1 + \frac{\kappa}{2}\|(I-WW^T)x\|_2^2 + \frac{1}{2}\|\mathcal{A}W^T x - b\|_2^2,$$

where the variable x stands for the wavelet coefficients, W is a given wavelet transform, and $\mathcal{A}$ is a given linear operator.

- Implement at least one type of W in your codes. You can use some existing library directly such as GSL or wavelib. Alternatively you can provide a C++ implementation based on this MATLAB code.
- Perform numerical experiments on the image deblur and denoise tasks using different types of the proximal gradient method. Let $u \in \mathbb{R}^{n\times n}$ be the original gray-scale image. The observation b is given by

$$b = \mathcal{A}u + e,$$

where $\mathcal{A}$ can be a given convolution operator with Gaussian kernel (deblurring) or identity (denoising), and e is Gaussian noise. Note: the size of b must be at least 256×256.

2 Requirements and Hints

Requirements:

1. All code implementations should base on the framework "OptSuite" (any version since `20211121`).
2. Prepare a report including
   - detailed description of the design of each module (self-implemented functions, solvers, etc)
   - detailed answers to each question
   - tables of numerical results (including the total number of iterations, the optimality measures, the CPU time and etc) and their interpretation
3. Pack the report and all of your codes in one file named as "comp-StudentID-date.zip" and send it to TA: `pkuopt@163.com`
4. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

Hints:

1. For any questions about installation, please contact Liu Haoyang (liuhaoyang@pku.edu.cn). Some tutorial videos are available at PKU Disk.
2. The codes "OptSuite" already contain the implementation of several functions and the proximal gradient solver. See `src/Base/func` and `src/Composite/Solver` for reference. Also read the source codes in `example` folder for the usage of the solvers.