An $\mathcal{O}(n \log n)$ projection operator for weighted ℓ_1-norm regularization with sum constraint

Weiran Wang
Toyota Technological Institute at Chicago
weiranwang@ttic.edu

March 1, 2015

Abstract

We provide a simple and efficient algorithm for the projection operator for weighted ℓ_1-norm regularization subject to a sum constraint, together with an elementary proof. The implementation of the proposed algorithm can be downloaded from the author's homepage.

1 The problem

In this report, we consider the following optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \left\| \mathbf{x} - \mathbf{y} \right\|^2 + \sum_{i=1}^{n} d_i \left| x_i \right|, \\ \text{s.t.} \quad & \mathbf{x}^\top \mathbf{1} = 1, \end{aligned} \tag{1}$$

where $\mathbf{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$, $d_i \geq 0$, $i = 1, \dots, n$, and $\mathbf{1}$ is the n-dimensional vector consisting of all 1's. This is a quadratic program and the objective function is strictly convex (even though it is non-smooth), so there is a unique solution which we denote by $\mathbf{x} = [x_1, \dots, x_n]^\top$ with a slight abuse of notation.

Notice if $d_1 = \cdots = d_n$ and the constraint were absent, the problem has a closed form solution known as the soft-shrinkage operator (see, e.g., Beck and Teboulle, 2009), which is widely used for solving ℓ_1-regularized problem in learning sparse representations. But our problem (1) is more involved due to the constraint that couples all dimensions of $\mathbf{x}$. Nonetheless, we give an efficient algorithm with time complexity $\mathcal{O}(n \log n)$ for this problem using only the KKT theorem.

Remark 1.1. Our motivation for (1) also comes from sparse coding. Yu et al. (2009) propose the local coordinate coding (LCC) algorithm for learning sparse representations induced by locality. Given a data sample $\mathbf{u} \in \mathbb{R}^n$ and a set of landmark points $\{\mathbf{v}_j\}_{j=1}^C$ where $\mathbf{v}_j \in \mathbb{R}^n$, $j = 1, \dots, C$, the LCC algorithm reconstructs $\mathbf{u}$ from the landmark points while enforcing the faraway landmark points to contribute less than nearby landmark points (or to have smaller reconstruction coefficients). Let the reconstruction coefficient of $\mathbf{v}_j$ be w_j, $j = 1, \dots, C$. Then the optimization problem for these coefficients in LCC is

$$\begin{aligned} \min_{\mathbf{w}} \quad & \left\| \mathbf{u} - \sum_{j=1}^{C} w_j \mathbf{v}_j \right\|^2 + \lambda \sum_{j=1}^{C} \left\| \mathbf{u} - \mathbf{v}_j \right\|^2 \left| w_j \right| \\ \text{s.t.} \quad & \sum_{j=1}^{C} w_j = 1, \end{aligned} \tag{2}$$

where $\lambda > 0$ is some trade-off parameter. The constraint in (2) ensures that the representation is translation invariant. There are different ways of solving this problem, e.g., Elhamifar and Vidal (2011) have a similar optimization problem which they solve with Alternating Direction Method of Multipliers (Boyd et al., 2011).

One simple way of solving (2) is to use the gradient proximal algorithm and its Nesterov's acceleration scheme (see Beck and Teboulle, 2009 and the reference therein), where one iteratively takes a short gradient step for the smooth quadratic term and projects the new estimate with the weighted ℓ_1 regularization term subject to the sum constraint, where the projection operator solves exactly (1).

2 The solution

We solve the problem (1) using only the KKT theorem (Nocedal and Wright, 2006), which states the necessary and sufficient condition[1] satisfied by the solution $\mathbf{x}$. The Lagrangian of (1) is

$$\mathcal{L}(\mathbf{x}, \alpha) = \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|^2 + \sum_{i=1}^{n} d_i |x_i| + \alpha(\mathbf{x}^\top \mathbf{1} - 1), \tag{3}$$

where α is the Lagrange multipliers associated with the constraint. And the KKT system of this problem is

$$\begin{aligned} x_i - y_i + d_i + \alpha &= 0, \qquad \text{if} \quad x_i > 0, && \text{(4a)} \\ x_i - y_i - d_i + \alpha &= 0, \qquad \text{if} \quad x_i < 0, && \text{(4b)} \\ -d_i \le -y_i + \alpha &\le d_i, \qquad \text{if} \quad x_i = 0, && \text{(4c)} \\ \sum_{i=1}^{n} x_i &= 1, && \text{(4d)} \end{aligned}$$

where we have used the fact that the sub-differential of $|x|$ is $[-1, 1]$ at $x = 0$ to obtain (4c).

Denote $y_i^- = y_i - d_i$, $y_i^+ = y_i + d_i$, $i = 1, \dots, n$, which can be computed beforehand. We can then rewrite (4) in terms of α:

$$\begin{aligned} \alpha < y_i^- &\iff x_i > 0, && \text{(5a)} \\ \alpha > y_i^+ &\iff x_i < 0, && \text{(5b)} \\ y_i^- \le \alpha \le y_i^+ &\iff x_i = 0, && \text{(5c)} \\ \sum_{i:\, x_i > 0} (y_i^- - \alpha) + \sum_{i:\, x_i < 0} (y_i^+ - \alpha) &= 1. && \text{(5d)} \end{aligned}$$

Obviously, the Lagrange multiplier α is the key to our problem. Once the value of α is determined, we can easily obtain the optimal solution by setting

$$\begin{aligned} x_i &= y_i^- - \alpha \qquad \text{if } \; y_i^- > \alpha, && \text{(6a)} \\ x_i &= y_i^+ - \alpha \qquad \text{if } \; y_i^+ < \alpha, && \text{(6b)} \\ x_i &= 0 \qquad\qquad\;\; \text{otherwise.} && \text{(6c)} \end{aligned}$$

We can sort all dimensions of y_i^- and y_i^+ together (a total of $2N$ scalars) into an ascending z-sequence:

$$z_1 \le z_2 \le \dots \le z_{2N}. \tag{7}$$

An important observation is that the z-sequence partitions the real axis into $4N + 1$ disjoint sets, each being either a single point set $\{z_j\}$, $j = 1, \dots, 2N$ or an open interval of the form $(-\infty, z_1)$, (z_j, z_{j+1}), $j = 1, \dots, 2N - 1$, or (z_{2N}, ∞) and the Lagrange multiplier α for the solution must lie in one of them.

We then test each of the $4N + 1$ sets as follows. Assuming that α lies in one set, we can use (5a)–(5c) to conjecture the positive, negative, and zero dimensions of a possible solution $\hat{\mathbf{x}}$. After that, we use (5d) to compute a hypothesized value $\hat{\alpha}$ for the Lagrange multiplier, i.e.,

$$\alpha = \frac{\sum_{i:\, \hat{x}_i > 0} y_i^- + \sum_{i:\, \hat{x}_i < 0} y_i^+ - 1}{\sum_{i:\, \hat{x}_i > 0} 1 + \sum_{i:\, \hat{x}_i < 0} 1}. \tag{8}$$

[1] Strictly speaking, our objective is convex and non-smooth, so the condition is that the zero vector $\mathbf{0}$ lies in the sub-differential at the solution $\mathbf{x}$.

If the computed $\hat{\alpha}$ indeed lies in the assumed set (a point or an open interval), we have a KKT point and thus the solution.

Since the problem (1) is strictly convex and there exists a unique global optimum, this procedure will find the exact solution with no more than $4N+1$ tests. We can do this efficiently by sorting y_i^- and y_i^+ separately ($\mathcal{O}(n \log n)$ operations) and gradually merging the two sorted sequences (an $\mathcal{O}(n)$ operation). Therefore the total cost of our procedure for solving (1) is of order $\mathcal{O}(n \log n)$.

Algorithm 1 gives the detailed pseudocode for solving (1), whose MATLAB and `C++` implementation can be downloaded at `https://eng.ucmerced.edu/people/wwang5`.

References

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1): 1–122, 2011.

E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 55–63. MIT Press, Cambridge, MA, 2011.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.

K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 22. MIT Press, Cambridge, MA, 2009.

Algorithm 1 Pseudo-code of our projection operator for (1).

```
Input: y ∈ ℝ^n and d = [d_1, ..., d_n] where d_i ≥ 0, i = 1, ..., n.
  Sort y − d into y⁻: y⁻_1 ≤ y⁻_2 ≤ ··· ≤ y⁻_n. And sort y + d into y⁺: y⁺_1 ≤ y⁺_2 ≤ ··· ≤ y⁺_n.
  i ← 1, j ← 1                      % i/j index of the dimension of y⁻/y⁺ that will be merged next.
  % s_1/s_2 stores the sum of dimensions of y⁻/y⁺ that are strictly greater/smaller than the current estimate of α.
  s_1 ← Σ_{i=1}^{n} y⁻_i, s_2 ← 0, t ← n   % t is the number of nonzero dimensions of the hypothesized x.
  if (s1 + s2) < t · y⁻_1 then
    α ← (s1 + s2)/t; return          % α < y⁻_1, all dimensions of x are positive.
  end if
  while true do
    % Test a single point set.
    if y⁻_i < y⁺_j then
      k ← i                          % y⁻_i is the next value in the z-sequence.
      while (y⁻_k = y⁻_i) && (k ≤ n) do
        s_1 ← s_1 − y⁻_k, t ← t − 1, k ← k + 1   % Skip the contiguous block of identical dimensions in y⁻.
      end while
      if (s_1 + s_2 − 1) = t · y⁻_i then
        α ← y⁻_i; return             % α happens to lie in a single point set.
      else
        left ← y⁻_i, i ← k           % Otherwise, α lies in a open interval with left boundary left.
      end if
    else
      if y⁻_i > y⁺_j then
        % y⁺_j is the next value in the z-sequence.
        if (s_1 + s_2 − 1) = t · y⁺_j then
          α ← y⁺_j; return           % α happens to lie in a single point set.
        else
          left ← y⁺_j                % Otherwise, α lies in a open interval with left boundary left.
          while (y⁺_j = left) && (j ≤ n) do
            s_2 ← s_2 + y⁺_j, t ← t + 1, j ← j + 1   % Skip the contiguous block of identical entries in y⁺.
          end while
        end if
      else
        k ← i                        % y⁻_i = y⁺_j is the next value in the z-sequence.
        while (y⁻_k = y⁻_i) && (k ≤ n) do
          s_1 ← s_1 − y⁻_k, t ← t − 1, k ← k + 1
        end while
        if (s_1 + s_2 − 1) = t · y⁻_i then
          α ← y⁻_i; return
        else
          left ← y⁻_i, i ← k
          while (y⁺_j = left) && (j ≤ n) do
            s_2 ← s_2 + y⁺_j, t ← t + 1, j ← j + 1
          end while
        end if
      end if
    end if
    % Find the right boundary of the open interval and test if it contains α.
    if y⁻_i < y⁺_j then
      right ← y⁻_i
    else
      right ← y⁺_j
    end if
    if t · left < (s_1 + s_2 − 1) && t · right > (s_1 + s_2 − 1) then
      α ← (s_1 + s_2 − 1)/t; return  % α lies in the open interval (left, right).
    end if
  end while
Output: α is the Lagrange multiplier of the problem (1), use (6) to obtain x.
```