Projection Onto A Simplex

Yunmei Chen and Xiaojing Ye *

February 9, 2011

Abstract

This mini-paper presents a fast and simple algorithm to compute the projection onto the canonical simplex $\triangle^n$. Utilizing the Moreau's identity, we show that the problem is essentially a univariate minimization and the objective function is strictly convex and continuously differentiable. Moreover, it is shown that there are at most n candidates which can be computed explicitly, and the minimizer is the only one that falls into the correct interval.

Keywords. Nonlinear programming, projection onto a simplex, Moreau's identity, proximity operators.

1 Introduction

The computation of projection onto simplex appears in many imaging and statistics problems, such as multiphase segmentation, diffusion tensor imaging, etc. The problem can be described as follows: for any given vector $y \in \mathbb{R}^n$, the projection of y onto the simplex $\triangle^n$ is to solve the minimization problem

$$x = \arg\min_{x \in \triangle^n} \|x - y\|, \tag{1}$$

where $\|\cdot\|$ denotes the regular Euclidean norm and $\triangle^n$ is the canonical simplex defined by

$$\triangle^n := \left\{ x = (x_1, \cdots, x_n)^T \in \mathbb{R}^n : 0 \le x_i \le 1, i = 1, \cdots, n, \text{ and } \sum_{i=1}^n x_i = 1 \right\}. \tag{2}$$

For example, $\triangle^2$ is the line segment between two points $(1, 0)$ and $(0, 1)$ in $\mathbb{R}^2$, and $\triangle^3$ is the triangle in $\mathbb{R}^3$ with vertexes $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. The solution is nontrivial and it does not yield an explicit form. Early attempts include iterative projections onto affine subspaces of $\mathbb{R}^n$ [3], and Lagrangian method that shows the optimal solution is $(y - t)_+$ for some t followed by the iterative bisection algorithm to find this t, etc. In addition, it is worth pointing out that there are a large number of literatures projection onto the ℓ_1 ball $\{x \in \mathbb{R}^n : \|x\|_1 \le 1\}$, e.g. [2, 5], which has has similar formulation and is in general easier to solve.

In this report, we present a novel, faster and simpler numerical algorithm to solve (1) for any dimension $n \ge 2$.

*Department of Mathematics, University of Florida, PO Box 118105, Gainesville, FL 32611-8105. Phone (352) 392-0281. Fax (352) 392-8357. Email: yun,xye@ufl.edu, Web: http://www.math.ufl.edu/∼yun,xye.

2 Algorithm

The minimization problem in (1) can be rewritten as

$$\min_{x\in\mathbb{R}^n} \pi(x) + \frac{1}{2}\|x-y\|^2, \quad y=(y_1,\cdots,y_n)^T \in \mathbb{R}^n, \tag{3}$$

where π is the indicator function of $\triangle^n$ defined by

$$\pi(x) = \begin{cases} 0 & \text{if } x \in \triangle^n, \\ +\infty & \text{otherwise .} \end{cases} \tag{4}$$

As π is a proper and convex function due to the fact that $\triangle^n$ is close and convex, we know (3), and hence (1), has a unique solution.

In the literature, the solution to (3) is also denoted by the Moreau's proximity operator [4]

$$(I+\partial\pi)^{-1}(y) := \arg\min_{x\in\mathbb{R}^n} \pi(x) + \frac{1}{2}\|x-y\|^2, \tag{5}$$

which satisfies the Moreau's identity [1]

$$y = (I+\partial\pi)^{-1}(y) + (I+\partial\pi^*)^{-1}(y), \tag{6}$$

where π^* is the Fenchel transform of π defined by

$$\pi^*(y) := \sup_{z\in\mathbb{R}^n} \langle z,y\rangle - \pi(z). \tag{7}$$

Note that π^* in (7) has a simple form if π is the indicator function of $\triangle^n$ defined in (4), i.e.

$$\begin{aligned} \pi^*(y) &:= \sup_{z\in\mathbb{R}^n} \langle z,y\rangle - \pi(z) = \sup_{z\in\triangle^n} \langle z,y\rangle \\ &= \max_{z\in\triangle^n} \sum_{i=1}^n z_i y_i = \max_{1\le i\le n}\{y_i\}. \end{aligned} \tag{8}$$

So $\pi^*(y)$ is just the largest component of y. Based on (6), it is suffice to compute the Moreau's proximity operator of π^*

$$\begin{aligned} (I+\partial\pi^*)^{-1}(y) &= \arg\min_{z\in\mathbb{R}^n} \pi^*(z) + \frac{1}{2}\|z-y\|^2 \\ &= \arg\min_{z\in\mathbb{R}^n} \left\{ \max_{1\le i\le n}\{z_i\} + \frac{1}{2}\|z-y\|^2 \right\} \end{aligned} \tag{9}$$

and then (5) can be obtained by

$$(I+\partial\pi)^{-1}(y) = y - (I+\partial\pi^*)^{-1}(y). \tag{10}$$

To find the solution of (9), we first sort the components of $y=(y_1,\cdots,y_n)^T \in \mathbb{R}^n$ in the ascending order as $y_{(1)} \le \cdots \le y_{(n)}$. Then we know the minimization problem in (9) can be written as

$$\min_{z\in\mathbb{R}^n} \left\{ \max_{1\le i\le n}\{z_i\} + \frac{1}{2}\|z-y\|^2 \right\} = \min_{t\in\mathbb{R}} \min_{z\in\mathbb{R}^n} \left\{ t + \frac{1}{2}\|z-y\|^2 : \max_{1\le i\le n}\{z_i\} = t \right\} \tag{11}$$

by introducing the new variable t. For any fixed t, the inner minimization problem on the right of (11) is

$$\min_{z\in\mathbb{R}^n}\left\{t+\frac{1}{2}\|z-y\|^2 : \max_{1\le i\le n}\{z_i\}=t\right\} \tag{12}$$

and the minimizer $\hat{z}(t)$ (as it depends on t) is obviously

$$(\hat{z}(t))_i=\begin{cases} t & \text{if } y_i>t \\ y_i & \text{if } y_i\le t \end{cases}, \quad i=1,\cdots,n, \tag{13}$$

and the minimum value of (12) is

$$f(t):=t+\frac{1}{2}\|\hat{z}(t)-y\|^2=\begin{cases} t+\dfrac{1}{2}\displaystyle\sum_{j=1}^{n}(t-y_j)^2 & \text{if } t\le y_{(1)} \\ t+\dfrac{1}{2}\displaystyle\sum_{j=i+1}^{n}(t-y_{(j)})^2 & \text{if } y_{(i)}\le t\le y_{(i+1)},\ i=1,\cdots,n-1 \\ t+\dfrac{1}{2}(t-y_{(n)})^2 & \text{if } t\ge y_{(n)} \end{cases} \tag{14}$$

Note that f is well-defined at $y_{(1)},\cdots,y_{(n)}$ as shown in Lemma 2.1 below. Therefore, (11) is equivalent to the univariate minimization problem

$$\min_{t\in\mathbb{R}} f(t) \tag{15}$$

where $f(t)$ is defined in (14).

Note that f in (14) is piecewise quadratic, and hence is piecewise convex and smooth. Moreover, the following lemma shows that $f\in C^1(\mathbb{R})$.

Lemma 2.1. *The function f defined in* (14) *is continuous on $\mathbb{R}$, and its derivative f' exists and is also continuous on $\mathbb{R}$.*

Proof: According to (14), for all $i=1,\cdots,n$, there are

$$\lim_{t\to y_{(i)}^-} f(t)=y_{(i)}+\frac{1}{2}\sum_{j=i}^{n}(y_{(i)}-y_{(j)})^2=y_{(i)}+\frac{1}{2}\sum_{j=i+1}^{n}(y_{(i)}-y_{(j)})^2=f(y_{(i)}) \tag{16}$$

and

$$\lim_{t\to y_{(i)}^-} f'(t)=1+\sum_{j=i}^{n}(y_{(i)}-y_{(j)})=1+\sum_{j=i+1}^{n}(y_{(i)}-y_{(j)})=\lim_{t\to y_{(i)}^+} f'(y_{(i)}), \tag{17}$$

which prove the lemma. □

Based on Lemma 2.1, we can obtain the derivative f' as

$$f'(t)=\begin{cases} 1+\displaystyle\sum_{j=1}^{n}(t-y_j) & \text{if } t\le y_{(1)} \\ 1+\displaystyle\sum_{j=i+1}^{n}(t-y_{(j)}) & \text{if } y_{(i)}\le t\le y_{(i+1)},\ i=1,\cdots,n-1 \\ 1+(t-y_{(n)}) & \text{if } t\ge y_{(n)} \end{cases} \tag{18}$$

Note that f' is well-defined at the points $y_{(1)},\cdots,y_{(n)}$ in (18) as it is continuous. Based on the discussion above, we have the theorem that restricts the search in n candidates and obtains the solution to (1) using the only candidate that falls into the correct interval.

Theorem 2.2. *For any vector $y \in \mathbb{R}^n$, the projection of y onto $\triangle^n$ as in (1) is obtained by the positive part of $y - \hat{t}$:*

$$x = (y - \hat{t})_+. \tag{19}$$

where $\hat{t}$ is the only one in $\{t_i : i = 0, \cdots, n-1\}$ that falls into the corresponding interval as follows,

$$t_i := \frac{\sum_{j=i+1}^{n} y_{(j)} - 1}{n-i}, \quad i = 0, \cdots, n-1, \text{ where } t_1 \le y_{(1)} \text{ and } y_{(i)} \le t_i \le y_{(i+1)},\ i = 1, \cdots, n-1. \tag{20}$$

Proof. Based on Lemma 2.1, we know f is piecewise quadratic and $f \in C^1(\mathbb{R})$. As (3) has a unique minimizer, we know that (9) has a unique minimizer as well, and hence the optimal t is unique. Therefore, there is only one single t that has vanish derivative i.e. $f'(t) = 0$, and this t is the minimizer of $\min_{t\in\mathbb{R}} f(t)$.

Now we compute the possible choices for the optimizer $\hat{t}$. Note that f is piecewisely defined, and according to (18), there are at most n points that have vanish derivative. It is easy to check that they are those shown in (20).

However, since the optimal $\hat{t}$ exists and is unique, we know that there is one and only one of $\{t_i : i = 0, \cdots, n-1\}$ that can be the optimal $\hat{t}$. In another words, there is only one t_i that falls into the "correct" interval and hence is the optimal choice of $\hat{t}$.

Once $\hat{t}$ is obtained, we have the minimizer of (11) as $\hat{z}(\hat{t})$ where $\hat{z}(\cdot)$ is defined in (13), and hence obtain the solution x to (1) based on the Moreau's identity (10):

$$x_i = (y - \hat{z}(\hat{t}))_i = \begin{cases} y_i - \hat{t} & \text{if } y_i > \hat{t}, \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

which implies that $x = (y - \hat{t})_+$. This completes the proof. □

Based on Theorem 2.2, we only need to find the t_i in (20) that falls in the corresponding interval, and claim it as the optimal $\hat{t}$ for (21). This procedure is described as in Steps 2-4 in the Algorithm 1. An interpretation of such procedure is as follows. Suppose we start with a very large t, namely $t \ge y_{(n)}$, and let t go towards negative. First, we can see $f'(t) = 1 + (t - y_{(n)}) \ge 1 > 0$ if $t \ge y_{(n)}$ and hence the optimal t cannot occur in $[y_{(n)}, \infty)$. As f' is continuous and $f'(y_{(n)}) = 1 > 0$, we know f' is positive near $y_{(n)}^-$. As f is quadratic in $[y_{(n-1)}, y_{(n)})$, this also implies that the optimal t, if exists in this interval, can only be $t_{n-1} := y_{(n)} - 1$ based on (18). Due to the existence and uniqueness of $\hat{t}$, we can surely accept $\hat{t} = t_{n-1}$ if $y_{(n-1)} \le t_{n-1} < y_{(n)}$, or simply $t_{n-1} \ge y_{(n-1)}$ (since obviously $t_{n-1} < y_{(n)}$). If $t_{n-1} < y_{(n-1)}$, we know the optimal t is not achieved yet and hence $f'(y_{(n-1)}) > 0$ due to the fact that f is quadratic in $[y_{(n-1)}, y_{(n)})$. Repeating the similar argument, we can accept $\hat{t} = t_{n-2} := (y_{(n-1)} + y_{(n)} - 1)/2$ if $y_{(n-2)} \le t_{n-2} < y_{(n-1)}$, or simply $t_{n-2} \ge y_{(n-2)}$ as we know $f(t)$ is quadratic and keeps increasing near $y_{(n-1)}$ in this case (hence t_{n-2} cannot be equal to or larger than $y_{(n-1)}$). Based on this analysis, we can repeat at most $n-1$ steps of such "compute t_i – compare to $y_{(i)}$" processes ($i = n-1, n-2, \cdots, 1$) until $\hat{t}$ is found. If $\hat{t}$ is still not found after $n-1$ steps, then it must be $t_0 := (\sum_{j=1}^{n} y_j - 1)/n$.

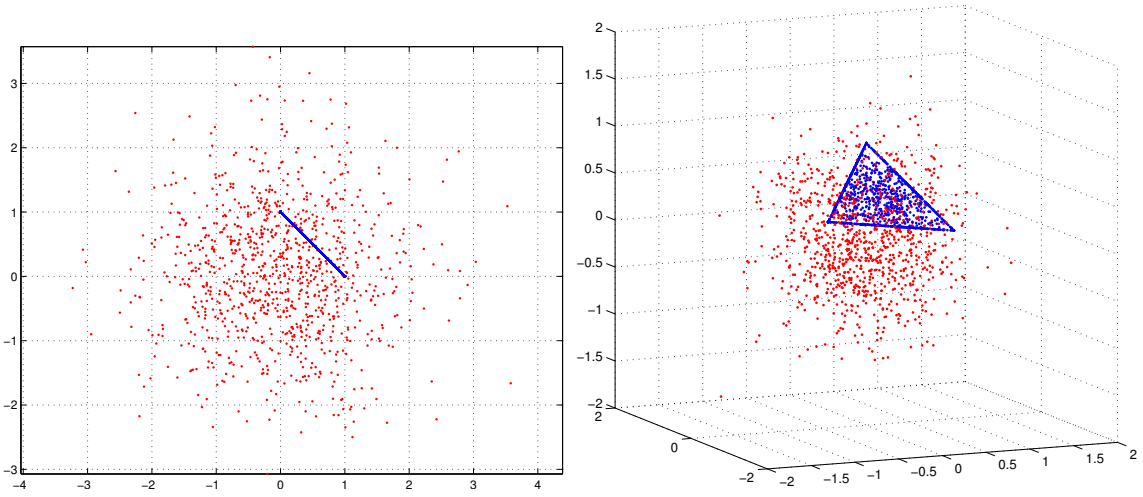
Completed with the last step $x = (y - \hat{t})_+$, the algorithm is summarized in Algorithm 1.

3 Numerical Examples

We manually computed the projections of several examples of y in low-dimensional (2D and 3D) cases using Algorithm 1, and the results turned out to be exact as expected. For more general cases, it is usually nontrivial to demonstrate the correctness of Algorithm 1 numerically.

Algorithm 1 (`projsplx`). Projection of $y \in \mathbb{R}^n$ onto the simplex $\triangle^n$.

1. Input $y = (y_1, \cdots, y_n)^T \in \mathbb{R}^n$;
2. Sort y in the ascending order as $y_{(1)} \leq \cdots \leq y_{(n)}$, and set $i = n - 1$;
3. Compute $t_i = \frac{\sum_{j=i+1}^{n} y_{(j)} - 1}{n-i}$. If $t_i \geq y_{(i)}$ then set $\hat{t} = t_i$ and go to Step 5, otherwise set $i \leftarrow i - 1$ and redo Step 3 if $i \geq 1$ or go to Step 4 if $i = 0$;
4. Set $\hat{t} = \frac{\sum_{j=1}^{n} y_j - 1}{n}$;
5. Return $x = (y - \hat{t})_+$ as the projection of y onto $\triangle^n$.



(a) 2D case (b) 3D case

Figure 1: Projections (blue) of 1024 random points (red) onto simplexes $\triangle^2$ (left) and $\triangle^3$ (right).

For completeness of this report, we show several numerical examples of Algorithm 1. The algorithm is implemented in C and compiled in MATLAB (Version R2010b) using `mex` function. The computations are performed on a Lenovo ThinkPad laptop with Intel Core 2 CPU at 2.53GHz, 3GB of memory, and GNU/Linux (Kernel version 2.6.35) operating system.

We first generate 1024 2D points from $N(0, I_2)$ using the MATLAB function `randn(2,1024)` and project them onto $\triangle^2$ using Algorithm 1. The results are plotted in Figure 1(a). A similar test projects 1024 samples from $N(0, 0.5I_3)$ to $\triangle^3$ and plots the results in Figure 1(b). Note that both tests are not designed to show that Algorithm computed the correct projections, since the correspondences are not shown. On the other hand, these two tests demonstrate that the outputs x are indeed located at the simplex (note that we did not check anywhere in Algorithm 1 that x is on the simplex).

The next test shows the CPU time of projections of $2^{16} = 65,536$ n-dimensional points draw from $N(0, I_n)$ for $n = 2, 3, \cdots, 50$. The results are plotted in Figure 3. The CPU time has the trend of increase as n becomes larger. Interestingly, the increasing rate is rather low. For example, for $n = 5$ to $n = 50$, the CPU time used only changes from 1.88s to 2.52s, for which the difference seems rather minor compared to the significant changes in computational complexity of the problem.

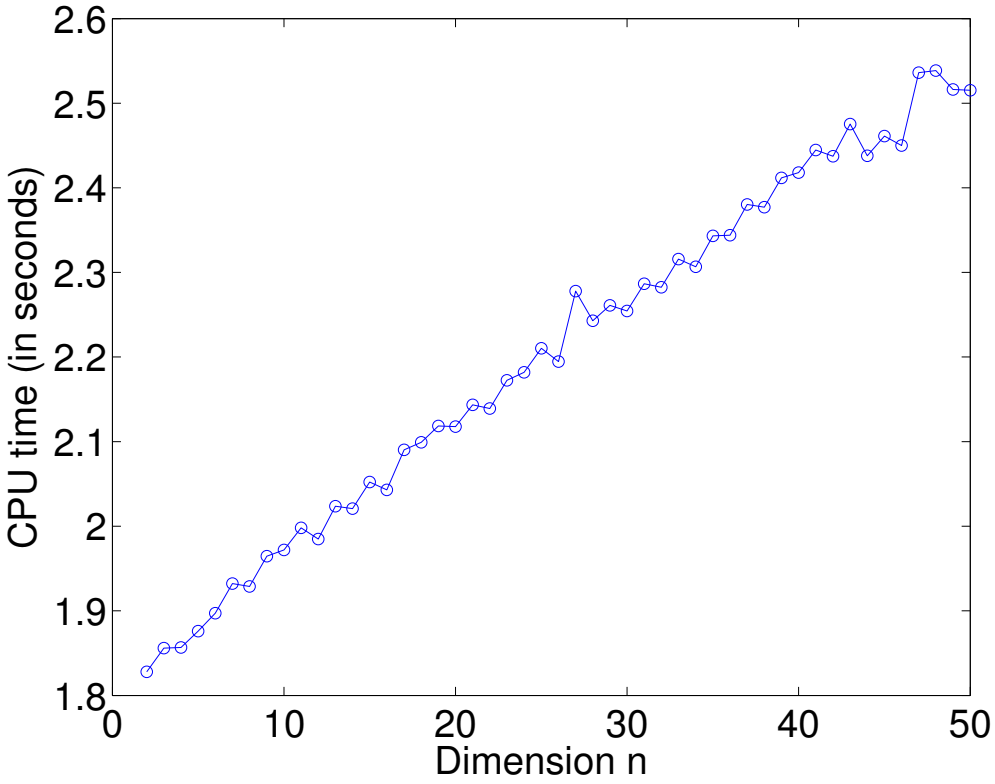


Figure 2: CPU time (in seconds) of projections of $65,536$ n-dimensional points drawn from $N(0, I_n)$. The tests are carried on for $n = 2, \cdots, 50$.

4 Concluding Remarks

In this paper we propose a fast and simple algorithm `projsplx` as shown in Algorithm 1 that projects an n-dimensional vector y to the canonical simplex $\triangle^n$. The computation comprises of the sort of components of the input y and at most n simple "compute-compare" processes. The solution is exact and the algorithm is extremely easy to implement. The MATLAB/C code is provided on the following websites for public use.

- Author's website: `http://www.math.ufl.edu/~xye/codes/projsplx.zip`
- Matlab Central: `http://www.mathworks.com/matlabcentral/fileexchange/30332`

5 Acknowledgement

Xiaojing Ye would like to thank Stephen Becker (CalTech) for his helpful comments in complementing the references.

References

[1] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Model. Simul.*, 4(4):1168–1200, 2005.

[2] John Duchi, Shai S. Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[3] C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simiplex of $\mathbb{R}^n$. *J. Optim. Theory Appl.*, 50(1):195–200, 1986.

[4] J. Moreau. Proximité et dualité dans un espace hilbertien. *Bull. Soc. Math. France*, 93:273–299, 1965.

[5] Ewout van den Berg and Michael P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31:890–912, November 2008.