Project Report

Melissa Gaddy, Zheming Gao

December 4, 2017

1 Matrix Multiplicative Weights Algorithm and its application

In the preceding section, we've introduced Multiplicative Weights(MW) algorithm and apply it on solving Linear programming problems of some certain form. We refer to the setting as *basic* or *scalar* case. Now we are considering some problem which has enough structure that we can obtain an analogous algorithm for them.

Moving from vectors to matrices, and from probability vector to density matrix, now we refer to the current setting as *matrix* case.

$$\begin{aligned} a_j^T\mathbf{x} &\geqslant 0 \\ \mathbf{1}^T\mathbf{x} &= 1 \\ \mathbf{x} &\geqslant 0. \end{aligned} \qquad\qquad \begin{aligned} A_j \bullet X &\geqslant 0 \\ Tr(S) &= 1 \\ X &\succeq 0. \end{aligned}$$

1.1 Max Cut problem

The max cut problem can be reformulate as the following form

$$\begin{array}{lll} \max & \frac{1}{4}L \bullet X & \\ \text{s.t.} & (e_j e_j^T) \bullet X & = 1, \quad \forall j = 1, \dots, n \\ & X & \succeq 0. \end{array} \tag{1}$$

where $X \in \mathbb{S}_+^n$.

If we denote the optimal value as b and scale X by $1/n$, then we would like to check the feasibility of the constraints:

$$\begin{array}{ll} \frac{n}{4b}L \bullet X & \geqslant 1 \\ n(e_j e_j^T) \bullet X & = 1, \quad \forall j = 1, \dots, n \\ X & \succeq 0. \end{array}$$

Finally, if we take $A_j = n(e_j e_j^T) - I, (j = 1, \dots, n)$, and $A_{n+1} = \frac{n}{4b}L - I$, then the feasibility problem can be reformed as

$$\begin{array}{lll} A_j \bullet X & \geqslant 0 & \forall j = 1, \ldots, n+1 \\ \mathrm{Tr}(X) & = 1 & \\ X & \succeq 0. & \end{array} \tag{2}$$

on which the Matrix-MW algorithm can be applied. To apply Matrix-MW, we would like to solve the large-margin problem with an error parameter $\delta > 0$. Denote $K = \{X \succeq 0 | \mathrm{Tr}(X) = 1\}$, we either solve the problem to an additive error of δ, i.e., find $X \in K$ such that $A_j \bullet X \geqslant -\delta, \forall j = 1, \ldots, n+1$, or prove the system (2) is infeasible.

Initialize $X^{(1)} = 1/nI$, and set parameters ϵ, $\eta = -\ln(1-\epsilon)$, we apply Matrix-MW on three different examples. For each example, we list the result and errors for different parameters.

1.1.1 Results

ϵ	0.1	0.01	0.001	0.0001
δ	0.1	0.01	0.001	0.0001
# of rounds	20	328	3465	34827
Rank(X)	4	4	4	4
$\lVert X - X^*\rVert_2$	2.7700	2.7557	2.7556	2.7545
$\lVert X - X^*\rVert_2 / \lVert X^*\rVert_2$	0.6925	0.6889	0.6887	0.6886
$\lvert b^* - b\rvert$	0.4595	0.0482	0.0039	0.0002

Table 1: toy example

ϵ	0.1	0.01	0.001	0.0001
δ	0.1	0.01	0.001	0.0001
# of rounds	19	633	7230	73228
Rank(X)	10	10	10	10
$\lVert X - X^*\rVert_2$	8.9298	8.9171	8.9169	8.9168
$\lVert X - X^*\rVert_2 / \lVert X^*\rVert_2$	0.8930	0.8917	0.8917	0.8917
$\lvert b^* - b\rvert$	1.9107	0.1908	0.0204	0.0011

Table 2: 10 nodes example

ϵ	0.1	0.01	0.001	0.0001
δ	0.1	0.01	0.001	0.0001
# of rounds	2777	49971		
$\lvert b^* - b\rvert$	3.2430	0.7318		

Table 3: 100 nodes example

1.1.2 Comments

1. "# or rounds" is the number of iterations for Matrix-MW algorithm. It stops when $A_j \bullet X \geqslant -\delta$ is satisfied for any $j = 1, \ldots, n+1$.

2. Solution X and the value of $b = \frac{1}{4} L \bullet X$ are obtained from iteration. X^* and b^* are optimal solution and optimal value to problem (1). We list the error of solution $\|X - X^*\|_2$ and the error of object value $|b - b^*|$.

3. Notice the solution absolute error in 2-norm is very large, so we also calculate the relative error in 2-norm for the toy example and the 10-node example.

4. The 100-node example has blank units in table since the algorithm converges very slow when ϵ and δ are smaller than 10^{-2}. (# of iteration will exceed 10^5).

5. Rank of the iteration solution is not 1. So the Rank = 1 constraint of original MAX CUT problem cannot be eliminated.

6. Notice that for each result, the ϵ and δ are in the same order of magnitude. The reason is, if we set ϵ and δ in different orders of magnitude, e.g., $\epsilon = 0.01$ and $\delta = 0.001$, or $\epsilon = 0.001$ and $\delta = 0.01$, the algorithm will not converge or converge very slow. From this we conclude that Matrix-MW is very parameter sensitive.