Convergence of Power Methods

Qi Lei
(Dated: January 26, 2015)

Two versions of Power Method. One is the classical one, the other is with some noise.

Algorithm 1:
Input: Symmetric matrix $A \in \mathbb{R}^n$, number of iteration L.
1. Choose $x_0 \in \mathbb{R}^n$.
2. For $l = 1$ to L:
   (a) $y_l \longleftarrow Ax_l$
   (b) $x_l = y_l / \|y_l\|$

Output: vector x_L
Lemma: $\sigma_1 \leq \sigma_2, \leq \cdots \leq \sigma_{n-1} < \sigma_n$ are the singular values of symmetric square matrix A. And $z_1, z_2, \cdots, z_n$ are the corresponding right eigenvectors. Denote $tan\theta_l = tan\theta(z_n, x_l)$. Then we have $tan\theta_{l+1} \leq tan\theta_l \times \sigma_{n-1}/\sigma_n$.
Proof. Suppose $x_l = cos\theta_l z_n + sin\theta_l u_l$. $u_l \in z_n^{\perp}$.
Then

$$\begin{aligned} Ax_l &= cos\theta_l Az_n + sin\theta_l Au_l \\ &= cos\theta_l \sigma_n z_n + sin\theta_l \|Au_l\| \frac{Au_l}{\|Au_l\|} \end{aligned}$$

Suppose $u_l = \sum_{p=1}^{n-1} \alpha_p z_p$, then $Au_l = \sum_{p=0}^{n-1} \sigma_p \alpha_p z_p \in z_n^{\perp}$, so

$$tan\theta_{l+1} = \frac{sin\theta_l \|Au_l\|}{cos\theta_l \sigma_n}$$

Now $\|Au_l\|^2 = \sum_{p=1}^{n-1} \sigma_p^2 \alpha_p^2 \leq \max_{p=1}^{n-1}\{\sigma_p^2\} \sum_{p=1}^{n-1} \alpha_p^2 \leq \sigma_{n-1}^2$. So $tan\theta_l + 1 \leq tan\theta_l \frac{\sigma_{n-1}}{\sigma_n}$.

Algorithm 2:
Input: Symmetric matrix $A \in \mathbb{R}^n$, noise added in each step g_l, number of iteration L.
1. Choose $x_0 \in \mathbb{R}^n$.
2. For $l = 1$ to L:
   (a) $y_l \longleftarrow Ax_l + g_l$
   (b) $x_l = y_l / \|y_l\|$

Output: vector x_L
Lemma: $\sigma_1 \leq \sigma_2, \leq \cdots \leq \sigma_{n-1} < \sigma_n$ are the singular values of symmetric square matrix A. And $z_1, z_2, \cdots, z_n$ are the corresponding right eigenvectors. Denote $tan\theta_l = tan\theta(z_n, x_l)$. g_l is the noise added in each iteration step. Then $tan\theta_{l+1} \leq max\{tan\theta_l \times \sigma_{n-1}/\sigma_n, tan\langle z_n, g_l \rangle\}$.
Proof. Suppose $x_l = cos\theta_l z_n + sin\theta_l u_l$. $u_l \in z_n^{\perp}$.
Then

$$\begin{aligned} y_{l+1} = Ax_l + g_l &= cos\theta_l Az_n + sin\theta_l Au_l + g_l \\ &= cos\theta_l \sigma_n z_n + sin\theta_l Au_l + g_l. \end{aligned}$$

Now suppose $x_{l+1} = cos\theta_{l+1} z_n + sin\theta_{l+1} u_{l+1}$, for some $u_{l+1} \in z_n^{\perp}$. Then

$$\begin{aligned}
cos\theta_{l+1} &= z_n^T x_{l+1} = (cos\theta_l \sigma_n + z_n^T g_l)/\|y_{l+1}\| \\
sin\theta_{l+1} &= u_{l+1}^T x_{l+1} = (sin\theta_l u_{l+1}^T A u_l + u_{l+1}^T g_l)/\|y_{l+1}\|. \\
tan\theta_{l+1} &= \frac{sin\theta_{l+1}}{cos\theta_{l+1}} \\
&= \frac{sin\theta_l u_{l+1}^T A u_l + u_{l+1}^T g_l}{cos\theta_l \sigma_n + z_n^T g_l} \\
&\leq \frac{sin\theta_l u_{l+1}^T A u_l + \|g_l\| sin\langle z_n, g_l\rangle}{cos\theta_l \sigma_n + \|g_l\| cos\langle z_n, g_l\rangle} \\
&\leq \frac{sin\theta_l u_{l+1}^T A u_l + \|g_l\| sin\langle z_n, g_l\rangle}{cos\theta_l \sigma_n - \|g_l\| |cos\langle z_n, g_l\rangle|}
\end{aligned}$$

The above part is what appears in the paper and also from the webpage. So what we need to do here is to bound both $sin\langle g_l, z_n\rangle$ and $cos\langle g_l, z_n\rangle$ from above, which means we need just to bound $\|g_l\|$. But this is not possible in our case. So I think about change a little bit about the lemma to the lower part.

$$\begin{aligned}
tan\theta_{l+1} &\leq \frac{sin\theta_l u_{l+1}^T A u_l + \|g_l\| sin\langle z_n, g_l\rangle}{cos\theta_l \sigma_n + \|g_l\| cos\langle z_n, g_l\rangle} \quad \texttt{(Suppose } sin\langle z_n, g_l\rangle, cos\langle z_n, g_l\rangle \texttt{ are positive.)} \\
&\leq max\{\frac{sin\theta_l \sigma_{n-1}}{cos\theta_l \sigma_n}, \frac{sin\langle z_n, g_l\rangle}{cos\langle z_n, g_l\rangle}\} \\
&= max\{tan\theta_l \frac{\sigma_{n-1}}{\sigma_n}, tan\langle z_n, g_l\rangle\}
\end{aligned}$$

Algorithm 2+:
Input: Symmetric matrix $A \in \mathbb{R}^n$, selected row number r, number of iteration L.
1. Choose $x_0 \in \mathbb{R}^n, y_0 = x_0$.
2. For $l = 1$ to L:
   (a) $\mathcal{K}_l$ is a random subset of $\{1, 2, \cdots, n\}, |\mathcal{K}_l| = r, y_l \longleftarrow y_{l-1}, y_{l,\mathcal{K}_l} \longleftarrow A_{\mathcal{K}_l} x_l$
   (b) $x_l = y_l/\|y_l\|$
Output: vector x_L
Remark: For some matrix of vector X, and set $\mathcal{K} \subset \{1, 2, \cdots, n\}$,

$$X_{\mathcal{K}} = X_{k_1,k_2,\cdots,k_r} = \begin{bmatrix} x_{k_1} \\ x_{k_2} \\ \cdots \\ x_{k_r} \end{bmatrix} \sim \begin{bmatrix} 0 \\ \cdots \\ 0 \\ x_{k_1} \\ 0 \\ \cdots \\ 0 \\ x_{k_2} \\ \cdots \\ x_{k_r} \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

Some analysis: As in Algorithm 2, the difference between y_{l+1} and Ax_l could be considered as noise. The noise g_l produced by Algorithm 2+ could be denoted as

$$\begin{aligned}
g_l &= y_{l+1} - Ax_l \\
&= y_l - y_{l,\mathcal{K}_l} + A_{\mathcal{K}_l} x_l - Ax_l \\
&= (I - I_{\mathcal{K}_l}) y_l + (A_{\mathcal{K}_l} - A) x_l \\
&= (A - \|y_l\| I)_{\{n\}-\mathcal{K}_l} x_l
\end{aligned}$$

So

$$
\begin{aligned}
tan\langle g_l, z_n\rangle &= \frac{\|V^T(A-\|y_l\|I)_{\{n\}-\mathcal{K}_l}x_l\|}{z_n^T(A-\|y_l\|I)_{\{n\}-\mathcal{K}_l}x_l} \\
&= \frac{\|V^T_{\{n\}-\mathcal{K}_l}(A-\|y_l\|I)x_l\|}{z^T_{n,\{n\}-\mathcal{K}_l}(A-\|y_l\|I)x_l} \quad \texttt{(here } V=[z_1|z_2|\cdots|z_{n-1}])
\end{aligned}
$$

Some observations between different optimized ways and original power method:

1. uniformly sampled rows

Eventually it will converge. Intuitively, the expected performance of each iteration is just similar to power method in the long run.

However, it may cost a little more time.

2.weighted sampled rows

The larger n is, the lesser λ_1/λ_2 is, the better weighted sampling performs.

Weight on dominant eigenvector is better than weight on the norm of A.

MATRIX COMPLETION INTUITION

$$
\begin{aligned}
f(x,y) &= \|A-\vec{x}\vec{y}^T\|_F \\
&= \sum_i\sum_j(a_{ij}-x_iy_j)^2 \\
&= \sum_i\|\vec{a}_i-x_i\vec{y}\|_2^2
\end{aligned}
$$

For individual i,

$$
\begin{aligned}
&\|\vec{a}_i-x_i\vec{y}\|_2^2 \\
=\ &\|x_i\vec{y}\|_2^2-2x_i\vec{a}_i^T\vec{y}+\|\vec{a}_i\|_2^2 \\
=\ &\|\vec{y}\|_2^2(x_i-\frac{a_i^Ty}{\|y\|_2^2})^2+\|a_i\|_2^2-\frac{(a_i^Ty)^2}{\|y\|_2^2}
\end{aligned}
$$

Take $x_i=\frac{a_i^Ty}{\|y\|_2^2}$, then $f(x,y)$ reaches its minimum for individual $x_i, i=1,2,\cdots,n$, which is $\|a_i\|_2^2-\frac{(a_i^Ty)^2}{\|y\|_2^2}$. And $f(x,y)$ correspondingly decreases $\|\vec{y}\|_2^2(x_i-\frac{a_i^Ty}{\|y\|_2^2})^2$, written as Δf_{x_i}.

Likewise, for individual $y_j,\ j=1,2,\cdots,n,\ f(x,y)$ reaches its minimum when we take new $y_j\doteq\frac{a_j^Tx}{\|x\|_2^2}$, and $f(x,y)$ correspondingly decreases $\|\vec{x}\|_2^2(y_j-\frac{a_j^Tx}{\|x\|_2^2})^2$, written as Δf_{y_j}.

- Greedy Coordinate Descent:
  By comparing the potential decrease of $f(x,y)$, we could apply Greedy Coordinate Descent to this approach. For each step t, we update k entries of $x^{(t)}$ or $y^{(t)}$. Take $x^{(t)}$ as an example. $x_\Omega^{(t+1)}\leftarrow A_\Omega y^{(t)}/|y^{(t)}|^2$. Then $\Delta f_{x_\Omega}^{(t+1)}$ vanishes to 0. And also $\Delta f_y^{(t+1)}=\|x^{(t+1)}\|_2^2(y_j^{(t)}-\frac{a_j^Tx^{(t+1)}}{\|x^{(t+1)}\|_2^2})^2$. The whole process takes up to $4k+kn+2n$ flops.