Practical and Fast Momentum-Based Power Methods

T. Rabbani & A. Jain & A. Rajkumar & F. Huang

July 31, 2021

Overview

Classical Power Method

Goal

For a diagonalizable matrix A, compute its dominant eigenvalue λ_1 and associated dominant eigenvector v_1.

The power method is an iterative program which solves this problem assuming there is a gap between the two largest eigenvalues, i.e., $|\lambda_1| > |\lambda_2|$.

Classical Power Method

Assume $A \in \mathbb{R}^{n \times n}$ is diagonalizable and $|\lambda_1| > |\lambda_2|$.

Algorithm 1 Vanilla Power Method

```
Choose a random vector q_0 ∈ R^n.
for k = 1 to T do
    q_k = A q_{k-1}
    q_k ← q_k / ||q_k||
    γ_k = q_k^⊤ A q_k
end for
return q_T, γ_T
```

If q_0 is non-orthogonal to v_1, then $q_T \to v_1, \gamma_T \to \lambda_1$ as $T \to \infty$.

Classical Power Method

Assume $A \in \mathbb{R}^{n \times n}$ is diagonalizable and $|\lambda_1| > |\lambda_2|$.

Algorithm 2 Vanilla Power Method

1: Choose a random vector $q_0 \in \mathbb{R}^n$.
2: **for** $k = 1$ to T **do**
3: $\quad q_k = Aq_{k-1}$
4: $\quad q_k \leftarrow q_k / ||q_k||$
5: $\quad \gamma_k = q_k^\top A q_k$
6: **end for**
7: **return** q_T, γ_T

If q_0 is non-orthogonal to v_1, then $q_T \to v_1, \gamma_T \to \lambda_1$ as $T \to \infty$.
Convergence rate of $T = \mathcal{O}(\frac{1}{\Delta} \log \frac{1}{\epsilon})$, where $\Delta := |\lambda_1 - \lambda_2|$ and $\epsilon := ||q_T - v_1||$ is the desired accuracy.

Accelerated Power Method

De Sa et al., "Accelerated Stochastic Power Iteration." (2018) introduces a scheme for speeding up the power method using a momentum term.

Algorithm 3 Power+M

Choose a random vector $q_0 \in \mathbb{R}^n$.
$q_1 = Aq_0/||Aq_0||$
for $k = 1$ to T **do**
    $q_{k+1} = Aq_k - \beta q_{k-1}$
    $q_{k+1} \leftarrow q_{k+1}/||q_{k+1}||$
    $\gamma_{k+1} = q_{k+1}^\top A q_{k+1}$
end for
return q_T, γ_T

Inspired by Polyak's heavy-ball method, viz., accelerated gradient descent. (B. Polyak, "Some methods of speeding up the convergence of iteration methods." 1964.)

Accelerated Power Method

Algorithm 4 Power+M

```
Choose a random vector q_0 ∈ R^n.
q_1 = Aq_0/||Aq_0||
for k = 1 to T do
    q_{k+1} = Aq_k − βq_{k−1}
    q_{k+1} ← q_{k+1}/||q_{k+1}||
    γ_{k+1} = q_{k+1}^⊤ A q_{k+1}
end for
return q_T, γ_T
```

If $\beta = \lambda_2^2/4$, then this has a convergence rate of $T = \mathcal{O}(\frac{1}{\sqrt{\Delta}} \log \frac{1}{\epsilon})$! Comparable to the convergence rate of the SOTA Lanczos algorithm.

Accelerated Power Method

If $\beta = \lambda_2^2/4$, then this has a convergence rate of $T = \mathcal{O}(\frac{1}{\sqrt{\Delta}} \log \frac{1}{\epsilon})$! Comparable to the convergence rate of the SOTA Lanczos algorithm.

Problem: We have no idea what λ_2 is at runtime. Furthermore, if $\beta \notin [\lambda_2^2/4, \lambda_1^2/4)$ we risk slow or non-convergence.

Accelerated Power Method

If $\beta = \lambda_2^2/4$, then this has a convergence rate of $T = \mathcal{O}(\frac{1}{\sqrt{\Delta}} \log \frac{1}{\epsilon})$! Comparable to the convergence rate of the SOTA Lanczos algorithm.

Problem: We have no idea what λ_2 is at runtime. Furthermore, if $\beta \notin [\lambda_2^2/4, \lambda_1^2/4)$ we risk slow or non-convergence.

Solution: Intelligently approximate λ_2 and update β every iteration accordingly.

DMPower: Delayed-Momentum Power Method

How to approximate λ_2? By using Hotelling deflation and approximates of λ_1.

Algorithm 5 Hotelling Deflation

Require: λ_1, v_1.

1: Choose a random vector $w_0 \in \mathbb{R}^n$.
2: **for** $k = 1$ to T **do**
3: $\quad w_k = (A - \lambda_1 v_1 {v_1}^\top) w_{k-1}$
4: $\quad w_{k+1} \leftarrow w_{k+1} / ||w_{k+1}||$
5: $\quad \pi_{k+1} = w_{k+1}^\top A w_{k+1}$
6: **end for**
7: **return** w_T, π_T

If $|\lambda_2| > |\lambda_3|$, then $w_k \to v_2$, $\pi_k \to \lambda_2$.

DMPower: Delayed-Momentum Power Method

DMPower is broken up into two phases, a Pre-M phase which approximates λ_2 and then an M phase which experiences acceleration.

Algorithm 6 DMPower: Pre-M Phase

Require: $A \in \mathbb{R}^{d \times d}$ symmetric, unit $q_0 \in \mathbb{R}^d$, pre-momentum phase iterations J, momentum phase iterations K, unit $w_0 \in \mathbb{R}^d$

for $j = 1, 2, \ldots, J$ **do**
- $q_j \leftarrow Aq_{j-1}$
- $q_j \leftarrow q_j / q_j$
- $\nu_j \leftarrow q_j^\top A q_j$ {Rayleigh Quotient estimate of λ_1}
- $P \leftarrow \nu_j q_j q_j^\top$ {Approximation of $\lambda_1 v_1 v_1^\top$}
- $w_j \leftarrow (A - P) w_{j-1}$ {Inexact deflation}
- $w_j \leftarrow w_j / w_j$
- $\mu_j \leftarrow w_j^\top A w_j$ {Rayleigh Quotient estimate of λ_2}

end for

return w_J, μ_J

DMPower: Delayed-Momentum Power Method

Approximation of λ_1 and v_1.

Algorithm 7 DMPower: Pre-M Phase

Require: $A \in \mathbb{R}^{d\times d}$ symmetric, unit $q_0 \in \mathbb{R}^d$, pre-momentum phase iterations J, momentum phase iterations K, unit $w_0 \in \mathbb{R}^d$

for $j = 1, 2, \ldots, J$ **do**
  $q_j \leftarrow Aq_{j-1}$
  $q_j \leftarrow q_j / q_j$
  $\nu_j \leftarrow q_j^\top A q_j$ {Rayleigh Quotient estimate of λ_1}
  $P \leftarrow \nu_j q_j q_j^\top$ {Approximation of $\lambda_1 v_1 v_1^\top$}
  $w_j \leftarrow (A - P)w_{j-1}$ {Inexact deflation}
  $w_j \leftarrow w_j / w_j$
  $\mu_j \leftarrow w_j^\top A w_j$ {Rayleigh Quotient estimate of λ_2}
end for
return w_J, μ_J

DMPower: Delayed-Momentum Power Method

Approximation of λ_2 and v_2.

Algorithm 8 DMPower: Pre-M Phase

Require: $A \in \mathbb{R}^{d \times d}$ symmetric, unit $q_0 \in \mathbb{R}^d$, pre-momentum phase iterations J, momentum phase iterations K, unit $w_0 \in \mathbb{R}^d$

for $j = 1, 2, \ldots, J$ **do**

  $q_j \leftarrow A q_{j-1}$

  $q_j \leftarrow q_j / q_j$

  $\nu_j \leftarrow q_j^\top A q_j$ {Rayleigh Quotient estimate of λ_1}

  $P \leftarrow \nu_j q_j q_j^\top$ {Approximation of $\lambda_1 v_1 v_1^\top$}

  $w_j \leftarrow (A - P) w_{j-1}$ {Inexact deflation}

  $w_j \leftarrow w_j / w_j$

  $\mu_j \leftarrow w_j^\top A w_j$ {Rayleigh Quotient estimate of λ_2}

end for

return w_J, μ_J

DMPower: Delayed-Momentum Power Method

Momentum phase. Closely resembles Power+M.

Algorithm 9 DMPower: M Phase

$\widehat{\lambda}_2 = \mu_J$
$\beta \leftarrow \widehat{\lambda}_2^2/4$ {Approximated optimal momentum coefficient}
$q_1 \leftarrow q_J$ {Current estimate of v_1}
$q_0 \leftarrow \vec{0}$
for $k = 1, 2, \ldots, K$ **do**
    $q_{k+1} \leftarrow Aq_k - \beta q_{k-1}$ {Momentum update}
    $q_{k+1} \leftarrow q_{k+1}/q_{k+1}$
    $\nu_k \leftarrow q_{k+1}^\top A q_{k+1}$
end for
return q_K, ν_K

DMPower: Fine-tuning Pre-M

In our meta-algorithm, pre-M terminates after J iterations, but we need to terminate once we believe our approximation $\mu_j \approx \lambda_2$. In implementation, we therefore set a hyperparameter ρ, which determines when to exit pre-M.

Specifically, pre-M exits once $|\mu_j - \mu_{j-1}| < \rho$.

DMPower: Fine-tuning Pre-M

In our meta-algorithm, pre-M terminates after J iterations, but we need to terminate once we believe our approximation $\mu_j \approx \lambda_2$. In implementation, we therefore set a hyperparameter ρ, which determines when to exit pre-M.

Specifically, pre-M exits once $|\mu_j - \mu_{j-1}| < \rho$.

An error-bound such as ρ is far less restrictive at runtime than a guess of λ_2. Furthermore, in our paper, we provide a guarantee of convergence if one has a priori lower bound on $\Delta = |\lambda_1 - \lambda_2|$.

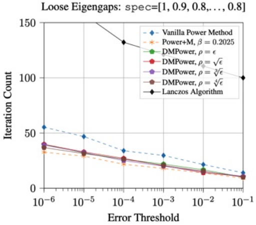
Convergence & Results

Theorem

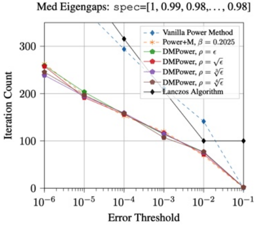
Let $\Delta = |\lambda_1 - \lambda_2|$ *denote the absolute difference between the largest and second-largest eigenvalues.*

With high probability, our proposed practical DMPower, after an efficient pre-momentum warm-up stage, outputs an ϵ*-close estimate of the leading eigenvector within the state-of-the-art* $\mathcal{O}\left(\frac{1}{\sqrt{\Delta}} \log\left(\frac{1}{\epsilon}\right)\right)$ *iteration complexity using a momentum acceleration, without requiring knowledge of* λ_2 *or hyperparameter selection for* λ_2*.*

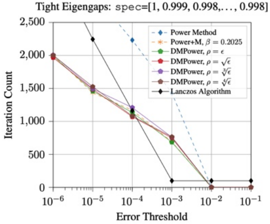
Convergence & Results



(a) $\Delta_{1,2} = \Delta_{2,3} = 0.1$



(b) $\Delta_{1,2} = \Delta_{2,3} = 0.01$



(c) $\Delta_{1,2} = \Delta_{2,3} = 0.001$

What else is in the paper?

- Streaming algorithm with convergence guarantee.
- Application: Spectral Clustering
- Comparisons between pre-M & M phase iterations.
- Extensive wall-time & iteration complexity comparisons.

Fin

Thanks for watching!