Hutch++

Optimal Stochastic Trace Estimation

Raphael A. Meyer (New York University)

With Christopher Musco (New York University), Cameron Musco (University of Massachusetts Amherst), and David P. Woodruff (Carnegie Mellon University)
Goal: Estimate trace of $d \times d$ matrix $A$:

$$\text{tr}(A) = \sum_{i=1}^{d} A_{ii} = \sum_{i=1}^{d} \lambda_i$$

In Downstream Applications, $A$ is not stored in memory. Instead, $B$ is in memory and $A = f(B)$.

- No. Triangles Estrada Index
- Log-Determinant $\text{tr}(\frac{1}{6}B^3)$
- $\text{tr}(e^B)$
- $\text{tr}(\ln(B))$

Computing $A = \frac{1}{6}B^3$ takes $O(n^3)$ time.

Computing $A x = \frac{1}{6}B(B(Bx))$ takes $O(n^2)$ time.

If $A = f(B)$, then we can often compute $A x$ quickly.
Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $A$:
  
  $$\text{tr}(A) = \sum_{i=1}^{d} A_{ii} = \sum_{i=1}^{d} \lambda_i$$

- In Downstream Applications, $A$ is not stored in memory.

- Instead, $B$ is in memory and $A = f(B)$:
  
<table>
<thead>
<tr>
<th>No. Triangles</th>
<th>Estrada Index</th>
<th>Log-Determinant</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{tr}(\frac{1}{6} B^3)$</td>
<td>$\text{tr}(e^B)$</td>
<td>$\text{tr}(\ln(B))$</td>
</tr>
</tbody>
</table>
Goal: Estimate trace of $d \times d$ matrix $A$:

$$\text{tr}(A) = \sum_{i=1}^{d} A_{ii} = \sum_{i=1}^{d} \lambda_i$$

In Downstream Applications, $A$ is not stored in memory. Instead, $B$ is in memory and $A = f(B)$:

<table>
<thead>
<tr>
<th>No. Triangles</th>
<th>Estrada Index</th>
<th>Log-Determinant</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{tr}(\frac{1}{6} B^3)$</td>
<td>$\text{tr}(e^B)$</td>
<td>$\text{tr}(\ln(B))$</td>
</tr>
</tbody>
</table>

- Computing $A = \frac{1}{6} B^3$ takes $O(n^3)$ time
- Computing $Ax = \frac{1}{6} B(B(Bx))$ takes $O(n^2)$ time
- If $A = f(B)$, then we can often compute $Ax$ quickly
Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $A$:

\[
\text{tr}(A) = \sum_{i=1}^{d} A_{ii} = \sum_{i=1}^{d} \lambda_i
\]

- In Downstream Applications, $A$ is not stored in memory.
- Instead, $B$ is in memory and $A = f(B)$:

<table>
<thead>
<tr>
<th>No. Triangles</th>
<th>Estrada Index</th>
<th>Log-Determinant</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr($\frac{1}{6} B^3$)</td>
<td>tr($e^B$)</td>
<td>tr(ln($B$))</td>
</tr>
</tbody>
</table>

- Computing $A = \frac{1}{6} B^3$ takes $O(n^3)$ time
- Computing $Ax = \frac{1}{6} B(B(Bx))$ takes $O(n^2)$ time
- If $A = f(B)$, then we can often compute $Ax$ quickly
- Goal: Estimate $\text{tr}(A)$ by computing $Ax_1, \ldots, Ax_k$
Implicit Matrix Trace Estimation: Estimate $\text{tr}(A)$ with as few Matrix-Vector products $Ax_1, \ldots, Ax_m$ as possible.

$$(1 - \varepsilon) \text{tr}(A) \leq \tilde{\text{tr}}(A) \leq (1 + \varepsilon) \text{tr}(A) \quad \text{w.p. } 1 - \delta$$
Our Contributions

For PSD matrix trace estimation,

- **Hutch++ algorithm**, which uses $\tilde{O}(\frac{1}{\epsilon})$ matrix-vector products.
  - Improves prior rate of $\tilde{O}(\frac{1}{\epsilon^2})$
  - Empirically works well
  - Matching $\Omega(\frac{1}{\epsilon})$ Lower Bound

Only 5 lines of code:

```matlab
function T = hutchplusplus(A, m)
S = 2*randi(2,size(A,1),m/3);
G = 2*randi(2,size(A,1),m/3);
[Q,~] = qr(A*S,0);
G = G - Q*(Q'*G);
T = trace(Q'*A*Q) + 1/size(G,2)*trace(G'*A*G);
end
```

$\tilde{O}$ notation only hide logarithmic dependence on the failure probability.
Idea: Hutchinson’s Estimator is **very** efficient unless $A$ is almost low-rank.
Hutch++ Intuition

Idea: Hutchinson’s Estimator is very efficient unless $A$ is almost low-rank.

1. Find a good rank-$k$ approximation $\tilde{A}_k$
2. Compute $\tilde{T} \approx \text{tr}(A - \tilde{A}_k)$ with $m$ steps of Hutchinson’s
3. Return $\text{Hutch}^{++}(A) = \text{tr}(\tilde{A}_k) + \tilde{T}$
THANK YOU

Code available at
github.com/RaphaelArkadyMeyerNYU/hutchplusplus