Dynamic Trace Estimation

Prathamesh Dharangutte, Christopher Musco
New York University

Conference on Neural Information Processing Systems (NeurIPS 2021)

Implicit trace estimation

- A basic problem in linear algebra
  Given matrix $A \in \mathbb{R}^{n \times n}$ and access to A via matrix vector products $Av \in \mathbb{R}^n$, for $v \in \mathbb{R}^n$ (implicit access), approximate $\text{tr}(A) = \sum_{i=1}^{n} A_{ii}$.

Implicit trace estimation

- A basic problem in linear algebra
  Given matrix $A \in \mathbb{R}^{n\times n}$ and access to A via matrix vector products $Av \in \mathbb{R}^n$, for $v \in \mathbb{R}^n$ (implicit access), approximate $\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$.
- Computing Hessian if often impractical but Hessian-vector product ($\nabla^2 f(x)v$) can be efficiently computed using finite difference method.

Implicit trace estimation

- A basic problem in linear algebra
  Given matrix $A \in \mathbb{R}^{n \times n}$ and access to A via matrix vector products $Av \in \mathbb{R}^n$, for $v \in \mathbb{R}^n$ (implicit access), approximate $\mathrm{tr}(A) = \sum_{i=1}^n A_{ii}$.
- Computing Hessian if often impractical but Hessian-vector product ($\nabla^2 f(x)v$) can be efficiently computed using finite difference method.

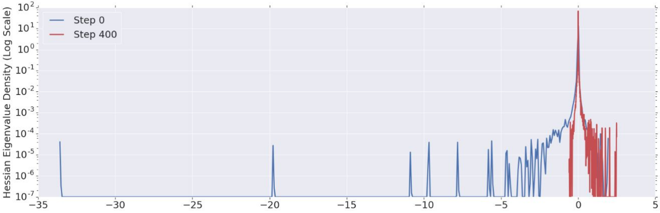


Figure 1: Ghorbani et al. [2019] analyze spectrum of Hessian for Resnet-32.

Implicit trace estimation

- For matrix functions $A = f(B)$, we can leverage iterative methods to approximate $Av = f(B)v$. (e.g. $A = B^{-1} / \exp(B) / \log(B)$). Typically, runtime is $O(n^2)$ compared to $O(n^3)$ for explicitly forming A.

Implicit trace estimation

- For matrix functions $A = f(B)$, we can leverage iterative methods to approximate $Av = f(B)v$. (e.g. $A = B^{-1} / \exp(B) / \log(B)$). Typically, runtime is $O(n^2)$ compared to $O(n^3)$ for explicitly forming A.
- Measure the computational cost in number of matrix-vector products required $Av_1, \ldots, Av_\ell$.

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

- Approximate $\text{tr}(A)$ as $h_\ell(A) = \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T A g_i$ where entries in $g_i \in \mathbb{R}^n$ are random i.i.d. ± 1.

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

- Approximate $\mathrm{tr}(A)$ as $h_\ell(A) = \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T A g_i$ where entries in $g_i \in \mathbb{R}^n$ are random i.i.d. ± 1.

$$\mathbb{E}[g^T A g] = \mathbb{E}\Big(\sum_{i=1}^{n} g_i^2 A_{ii} + \sum_{i=1}^{n}\sum_{\substack{j=1\\ i\neq j}}^{n} g_i g_j A_{ij}\Big) = \sum_{i=1}^{n} A_{ii}\mathbb{E}[g_i^2] = \sum_{i=1}^{n} A_{ii} = \mathrm{tr}(A)$$

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

- Approximate $\text{tr}(A)$ as $h_\ell(A) = \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T A g_i$ where entries in $g_i \in \mathbb{R}^n$ are random i.i.d. ± 1.

$$\mathbb{E}[g^T A g] = \mathbb{E}\Big(\sum_{i=1}^{n} g_i^2 A_{ii} + \sum_{i=1}^{n}\sum_{\substack{j=1\\ i\neq j}}^{n} g_i g_j A_{ij}\Big) = \sum_{i=1}^{n} A_{ii}\mathbb{E}[g_i^2] = \sum_{i=1}^{n} A_{ii} = \text{tr}(A)$$

- $h_\ell(A)$ approximates $\text{tr}(A)$ in expectation.

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

Main takeaways from the Hutchinson's estimator (Avron and Toledo [2011], Roosta-Khorasani and Ascher [2015], Cortinovis and Kressner [2020]):

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

Main takeaways from the Hutchinson's estimator (Avron and Toledo [2011], Roosta-Khorasani and Ascher [2015], Cortinovis and Kressner [2020]):

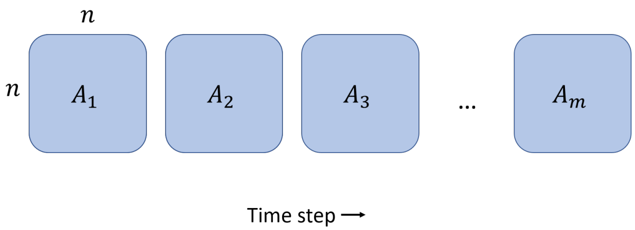
- Variance of $h_\ell(A) \leq \frac{2}{\ell}\|A\|_F^2$

Hutchinson's estimator (Hutchinson [1990], Girard [1987])

Main takeaways from the Hutchinson's estimator (Avron and Toledo [2011], Roosta-Khorasani and Ascher [2015], Cortinovis and Kressner [2020]):

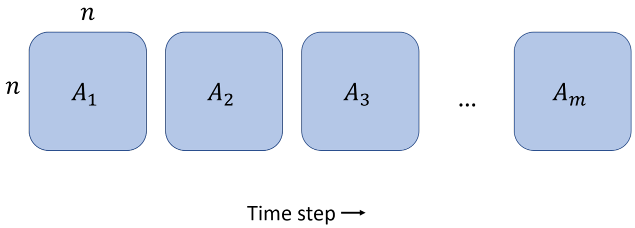
- Variance of $h_\ell(A) \leq \frac{2}{\ell}\|A\|_F^2$
- For $\ell = O(\frac{\log(1/\delta)}{\epsilon^2})$, with high probability, $|h_\ell(A) - \text{tr}(A)| \leq \epsilon\|A\|_F$

Dynamic setting



Want good approximations $t_1, t_2, ..., t_m$ across all time steps.

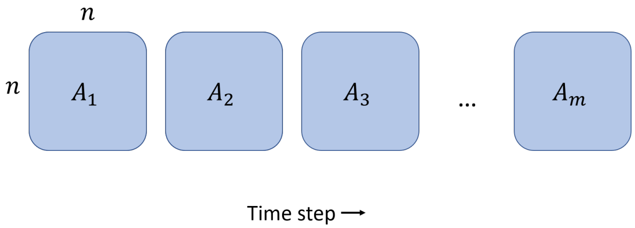
Dynamic setting



Want good approximations $t_1, t_2, \ldots, t_m$ across all time steps.

Naively, would require total $O(\frac{m \log(1/\delta)}{\epsilon^2})$ mat-vec products.

Dynamic setting

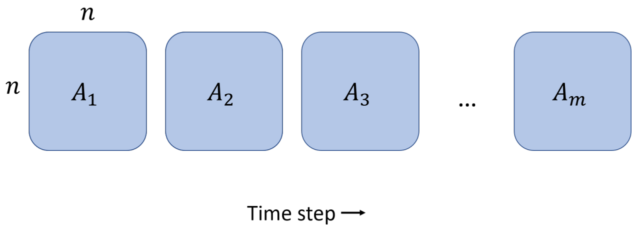


Want good approximations $t_1, t_2, ..., t_m$ across all time steps.

Naively, would require total $O(\frac{m \log(1/\delta)}{\epsilon^2})$ mat-vec products.

A natural question: Can we achieve $\ell_{\text{total}} < O(\frac{m \log(1/\delta)}{\epsilon^2})$?

Dynamic setting



Want good approximations $t_1, t_2, ..., t_m$ across all time steps.

Naively, would require total $O(\frac{m \log(1/\delta)}{\epsilon^2})$ mat-vec products.

A natural question: Can we achieve $\ell_{\text{total}} < O(\frac{m \log(1/\delta)}{\epsilon^2})$?

Our result: Yes and can obtain *quadratic* improvements under certain assumptions!

Problem formulation

Problem (Dynamic trace estimation)

Let $A_1, ..., A_m$ be $n \times n$ symmetric matrices satisfying:

1. $\|A_i\|_F \leq 1$, *for all* $i \in [1, m]$.
2. $\|A_{i+1} - A_i\|_F \leq \alpha$, *for all* $i \in [1, m-1]$.

Given implicit matrix-vector multiplication access to each A_i in sequence, the goal is to compute trace approximations $t_1, \ldots, t_m$ for $tr(A_1), ..., tr(A_m)$ such that, for each $i \in 1, \ldots, m$,

$$\mathbb{P}[|t_i - tr(A_i)| \geq \epsilon] \leq \delta.$$

Dynamic setting

$$A_2 = A_1 + \Delta_1$$

Dynamic setting

$$A_2 = A_1 + \Delta_1$$

$$tr(A_2) = tr(A_1) + tr(\Delta_1)$$

Dynamic setting

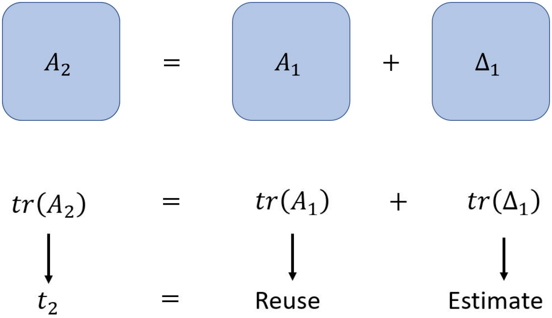
$$A_2 = A_1 + \Delta_1$$

$$tr(A_2) = tr(A_1) + tr(\Delta_1)$$

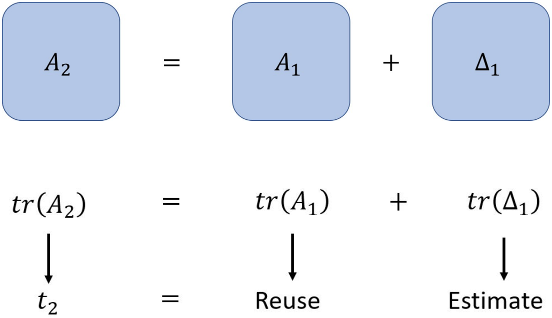
$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

t_2 = Reuse + Estimate

Dynamic setting



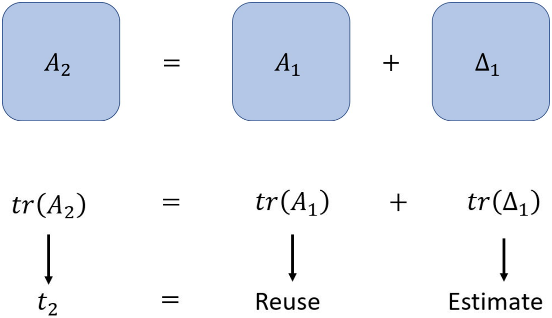
Observation: If $\|\Delta_i\|_F << \|A_i\|_F$, we should be able to accurately estimate $\text{tr}(\Delta_i)$ with a lot less matrix-vector products.



Observation: If $\|\Delta_i\|_F << \|A_i\|_F$, we should be able to accurately estimate $\mathrm{tr}(\Delta_i)$ with a lot less matrix-vector products.

$\Delta_1 = A_2 - A_1,$

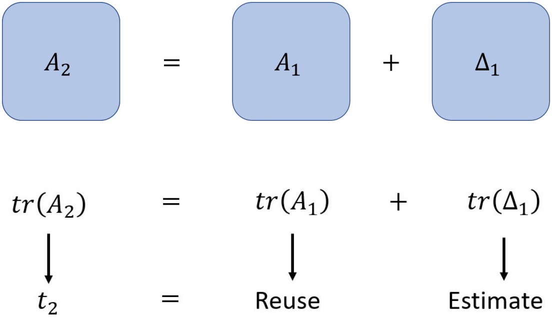
Dynamic setting



Observation: If $\|\Delta_i\|_F << \|A_i\|_F$, we should be able to accurately estimate $\mathrm{tr}(\Delta_i)$ with a lot less matrix-vector products.

$$\Delta_1 = A_2 - A_1, \quad \mathrm{tr}(\Delta_1) = \mathrm{tr}(A_2 - A_1)$$

Dynamic setting



Observation: If $\|\Delta_i\|_F << \|A_i\|_F$, we should be able to accurately estimate $\text{tr}(\Delta_i)$ with a lot less matrix-vector products.

$$\Delta_1 = A_2 - A_1, \quad \text{tr}(\Delta_1) = \text{tr}(A_2 - A_1) = \frac{1}{\ell} \sum_{i=1}^{\ell} g_i^T (A_2 - A_1) g_i$$

Dynamic setting

$$A_2 = A_1 + \Delta_1$$

$$tr(A_2) = tr(A_1) + tr(\Delta_1)$$

$$t_2 = \text{Reuse} + \text{Estimate}$$

Observation: If $\|\Delta_i\|_F << \|A_i\|_F$, we should be able to accurately estimate $\mathrm{tr}(\Delta_i)$ with a lot less matrix-vector products.

$$\Delta_1 = A_2 - A_1, \;\; \mathrm{tr}(\Delta_1) = \mathrm{tr}(A_2 - A_1) = \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T (A_2 - A_1) g_i = \frac{1}{\ell}\sum_{i=1}^{\ell} g_i^T (A_2 g_i - A_1 g_i)$$

DeltaShift

Instead of estimating $t_{i+1} = t_i + h_\ell(A_{i+1} - A_i)$[1]

[1] Note: h_ℓ is the Hutchinson estimator with ℓ mat-vec products.

DeltaShift

Instead of estimating $t_{i+1} = t_i + h_\ell(A_{i+1} - A_i)$[1]

Estimate, for $0 < \gamma < 1$

$$\underline{\text{DeltaShift}}: \quad t_{i+1} = (1-\gamma)t_i + h_\ell(A_{i+1} - (1-\gamma)A_i)$$

[1] Note: h_ℓ is the Hutchinson estimator with ℓ mat-vec products.

DeltaShift

Instead of estimating $t_{i+1} = t_i + h_\ell(A_{i+1} - A_i)$[1]

Estimate, for $0 < \gamma < 1$

$$\underline{\text{DeltaShift}}: \quad t_{i+1} = (1-\gamma)t_i + h_\ell(A_{i+1} - (1-\gamma)A_i)$$

- t_i's are still unbiased estimators of the trace.

[1] Note: h_ℓ is the Hutchinson estimator with ℓ mat-vec products.

DeltaShift

Instead of estimating $t_{i+1} = t_i + h_\ell(A_{i+1} - A_i)$[1]

Estimate, for $0 < \gamma < 1$

$$\underline{\text{DeltaShift}}: \quad t_{i+1} = (1-\gamma)t_i + h_\ell(A_{i+1} - (1-\gamma)A_i)$$

- t_i's are still unbiased estimators of the trace.
- Multiplying by $(1-\gamma)$ reduces the variance of the leading term.

[1] Note: h_ℓ is the Hutchinson estimator with ℓ mat-vec products.

DeltaShift

For any $\epsilon, \delta, \alpha \in (0, 1)$, the DeltaShift algorithm solves Dynamic Trace Estimation problem with

$$O\left(m \cdot \frac{\alpha \log(1/\delta)}{\epsilon^2} + \frac{\log(1/\delta)}{\epsilon^2}\right)$$

total matrix-vector multiplications involving $A_1, \ldots, A_m$.

DeltaShift

For any $\epsilon, \delta, \alpha \in (0, 1)$, the DeltaShift algorithm solves Dynamic Trace Estimation problem with

$$O\left(m \cdot \frac{\alpha \log(1/\delta)}{\epsilon^2} + \frac{\log(1/\delta)}{\epsilon^2}\right)$$

total matrix-vector multiplications involving $A_1, \ldots, A_m$.

For $\alpha \approx \epsilon$, DeltaShift requires $O(\frac{\log(1/\delta)}{\epsilon})$ total matrix-vector products.

Selecting γ

- How do you choose the parameter γ?

Selecting γ

- How do you choose the parameter γ?
- Can estimate near-optimal γ at each time-step with very little overhead. Let v_i be the variance of estimator at time-step i.

Selecting γ

- How do you choose the parameter γ?
- Can estimate near-optimal γ at each time-step with very little overhead. Let v_i be the variance of estimator at time-step i.
- Use: For any matrix A, $\|A\|_F^2 = \text{tr}(A^T A)$

Selecting γ

- How do you choose the parameter γ?
- Can estimate near-optimal γ at each time-step with very little overhead. Let v_i be the variance of estimator at time-step i.
- Use: For any matrix A, $\|A\|_F^2 = \mathrm{tr}(A^T A)$

$$\gamma_j^* = \min_{\gamma} \left[(1-\gamma)^2 v_{j-1} + \frac{2}{\ell} \|\Delta_j\|_F^2 \right]$$

Selecting γ

- How do you choose the parameter γ?
- Can estimate near-optimal γ at each time-step with very little overhead. Let v_i be the variance of estimator at time-step i.
- Use: For any matrix A, $\|A\|_F^2 = \mathrm{tr}(A^T A)$

$$\gamma_j^* = \min_{\gamma} \left[(1-\gamma)^2 v_{j-1} + \frac{2}{\ell}\|\Delta_j\|_F^2 \right]$$
$$= 1 - \frac{2h_\ell(A_{j-1}^T A_j)}{\ell v_{j-1} + 2h_\ell(A_{j-1}^T A_{j-1})}$$

Selecting γ

- How do you choose the parameter γ?
- Can estimate near-optimal γ at each time-step with very little overhead. Let v_i be the variance of estimator at time-step i.
- Use: For any matrix A, $\|A\|_F^2 = \mathrm{tr}(A^T A)$

$$\gamma_j^* = \min_{\gamma} \left[(1-\gamma)^2 v_{j-1} + \frac{2}{\ell} \|\Delta_j\|_F^2 \right]$$
$$= 1 - \frac{2 h_\ell(A_{j-1}^T A_j)}{\ell v_{j-1} + 2 h_\ell(A_{j-1}^T A_{j-1})}$$

- Note: We can reuse the same matrix-vector products used by trace estimation.

DeltaShift++

For a PSD matrix, recent algorithm by Meyer et al. [2021] obtains the (ϵ, δ) bounds with $\frac{\log(1/\delta)}{\epsilon}$ matrix-vector products.

DeltaShift++

For a PSD matrix, recent algorithm by Meyer et al. [2021] obtains the (ϵ, δ) bounds with $\frac{\log(1/\delta)}{\epsilon}$ matrix-vector products.

For stronger assumptions (in form of nuclear norm) on sequence of matrices:

$$\underline{\text{DeltaShift++}} : t_{i+1} = \gamma \cdot h_\ell^{++}(A_{i+1}) + (1 - \gamma) \cdot \left(t_i + h_\ell^{++}(A_{i+1} - A_i)\right)$$

DeltaShift++

For a PSD matrix, recent algorithm by Meyer et al. [2021] obtains the (ϵ, δ) bounds with $\frac{\log(1/\delta)}{\epsilon}$ matrix-vector products.

For stronger assumptions (in form of nuclear norm) on sequence of matrices:

$$\underline{\text{DeltaShift++}} : t_{i+1} = \gamma \cdot h_\ell^{++}(A_{i+1}) + (1-\gamma) \cdot \left(t_i + h_\ell^{++}(A_{i+1} - A_i)\right)$$

For $\|A_i\|_* \leq 1$ and $\|A_{i+1} - A_i\|_* \leq \alpha$ for all i, DeltaShift++ solves dynamic trace estimation problem with

$$O\left(m \cdot \frac{\sqrt{\alpha/\delta}}{\epsilon} + \frac{\sqrt{1/\delta}}{\epsilon}\right)$$

total matrix-vector products with $A_1, A_2, ..., A_m$.

DeltaShift++

We can estimate near-optimal γ for DeltaShift++ as well!

Let $K_A = \|A - A_k\|_F^2$

$$\gamma_j^* = \min_{\gamma} \left[\frac{\gamma^2 8 K_{A_j}}{\ell} + (1-\gamma)^2 (v_{j-1} + \frac{8 K_{\Delta_j}}{\ell}) \right]$$

We can estimate near-optimal γ for DeltaShift++ as well!

Let $K_A = \|A - A_k\|_F^2$

$$\gamma_j^* = \min_{\gamma} \left[\frac{\gamma^2 8 K_{A_j}}{\ell} + (1-\gamma)^2 (v_{j-1} + \frac{8 K_{\mathbf{\Delta}_j}}{\ell}) \right]$$
$$= \frac{8 K_{\mathbf{\Delta}_j} + \ell v_{j-1}}{8 K_{A_j} + \ell v_{j-1} + 8 K_{\mathbf{\Delta}_j}}$$

DeltaShift++

We can estimate near-optimal γ for DeltaShift++ as well!

Let $K_A = \|A - A_k\|_F^2$

$$\gamma_j^* = \min_{\gamma} \left[\frac{\gamma^2 8K_{A_j}}{\ell} + (1-\gamma)^2 (v_{j-1} + \frac{8K_{\mathbf{\Delta}_j}}{\ell}) \right]$$
$$= \frac{8K_{\mathbf{\Delta}_j} + \ell v_{j-1}}{8K_{A_j} + \ell v_{j-1} + 8K_{\mathbf{\Delta}_j}}$$

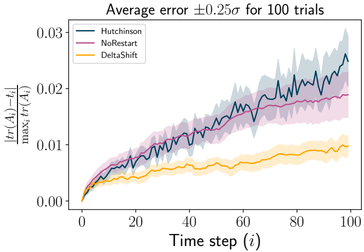
Similar to DeltaShift, we can reuse matrix-vector products from trace estimation!

Empirical results

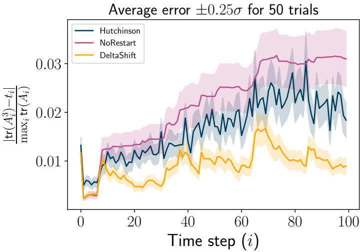
For the dynamic trace problem, we compare using the **same number of total matrix-products** for

- Hutchinson's estimator at each time step
- Estimate $\text{tr}(\Delta_i)$ at each time step and add to $\text{tr}(A_i)$ (NoRestart)
- DeltaShift

Empirical results



(a) Synthetic data with total matrix-vector products= $8 * 10^3$



(b) Graph data with total matrix-vector products= 10^4

- For estimating spectral density, trace of polynomials of the Hessian is used.

Empirical results

- For estimating spectral density, trace of polynomials of the Hessian is used.

The three term recurrence relation for Chebyshev polynomials is:

$$T_0(H) = I, \qquad T_1(H) = H, \qquad T_{n+1}(H) = 2HT_n(H) - T_{n-1}(H).$$

Empirical results

Table 1: Average error for trace of polynomials of Hessian with learning rate 0.001 and total matrix-vector products = 2000

	Hutchinson	NoRestart	DeltaShift
$T_1(H)$	2.5E-02	3.7E-02	**1.7E-02**
$T_2(H)$	1.2E-06	1.7E-06	**8.0E-07**
$T_3(H)$	4.0E-02	4.1E-02	**3.1E-02**
$T_4(H)$	1.5E-06	1.7E-06	**1.0E-06**
$T_5(H)$	2.1E-02	4.3E-02	**1.9E-02**

Future work

- Current choice of γ is a greedy heuristic, but works well empirically. Can we do better?

Future work

- Current choice of γ is a greedy heuristic, but works well empirically. Can we do better?
- Can we do better when Δ matrices have additional structure? Partial progress in form of DeltaShift++.

Thank you!