gipsa-lab

Grenoble | images | parole | signal | automatique | laboratoire

Variance Reduction for Inverse Trace Estimation via Random Spanning Forests

Yusuf Yiğit Pilavcı*
Pierre-Olivier Amblard
Simon Barthelmé
Nicolas Tremblay

07/09/2022

cnrs
Grenoble INP
Université Grenoble Alpes
UMR 5216

Introduction

- Trace is an essential algebraic operation.

Introduction

- Trace is an essential algebraic operation.
- In many applications, $\mathrm{tr}(f(\mathsf{L}))$ is the quantity of interest for a given matrix L.

Introduction

- Trace is an essential algebraic operation.
- In many applications, $\mathrm{tr}(f(\mathsf{L}))$ is the quantity of interest for a given matrix L.
- However, it is not always easy to compute...

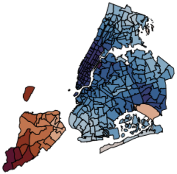
Introduction

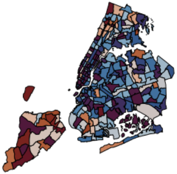
- Trace is an essential algebraic operation.
- In many applications, $\mathrm{tr}(f(\mathsf{L}))$ is the quantity of interest for a given matrix L.
- However, it is not always easy to compute...
- In this work, we focus on

$$f(\mathsf{L}) = q(\mathsf{L} + q\mathsf{I})^{-1},$$

where $q > 0$ and L is symmetric and diagonally dominant *i.e.* $\forall i \in \mathcal{V}$, $\sum_{j=1}^{n} |\mathsf{L}_{i,j}| \leq |\mathsf{L}_{i,i}|$.

Hyperparameter Selection for Graph Signal Smoothing

Original Signal: 

y: 

Hyperparameter Selection for Graph Signal Smoothing

Original Signal: $\mathbf{y}$: $\hat{\mathbf{x}}$:

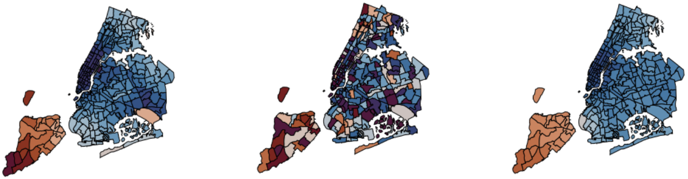


Figure: Median taxi fees paid in drop-off locations in NYC

Hyperparameter Selection for Graph Signal Smoothing

Original Signal: $\mathbf{y}$: $\hat{\mathbf{x}}$:

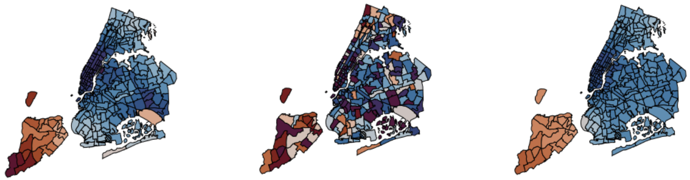


Figure: Median taxi fees paid in drop-off locations in NYC

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} q \underbrace{||\mathbf{y} - \mathbf{x}||^2}_{\text{Fidelity}} + \underbrace{\mathbf{x}^T \mathrm{L}\mathbf{x}}_{\text{Regularization}} \quad , \quad q > 0$$

where L is the graph Laplacian and $\mathbf{x}^T \mathrm{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w(i,j)(x_i - x_j)^2$.

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$$

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$$

- The denoising error $||\mathbf{x} - \hat{\mathbf{x}}||_2^2$ highly depends on q.

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$$

- The denoising error $||\mathbf{x} - \hat{\mathbf{x}}||_2^2$ highly depends on q.
- Many methods of finding good value of q needs to compute $\mathrm{tr}(\mathsf{K})$.

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathsf{K}\mathbf{y} \text{ with } \mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$$

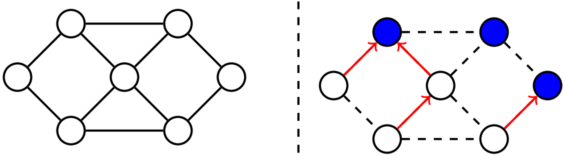
- The denoising error $||\mathbf{x} - \hat{\mathbf{x}}||_2^2$ highly depends on q.
- Many methods of finding good value of q needs to compute $\mathrm{tr}(\mathsf{K})$.
- However, computing the inverse takes $\mathcal{O}(n^3)$ operations.

Inverse Trace Estimation: Random Spanning Forests

- A recent algorithm is based on *random spanning forests* on graphs [Bar+19].

Inverse Trace Estimation: Random Spanning Forests

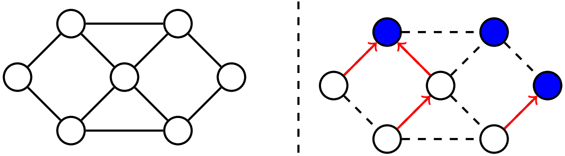
- A recent algorithm is based on *random spanning forests* on graphs [Bar+19].
- A rooted spanning forest on a graph:



A rooted spanning forest

Inverse Trace Estimation: Random Spanning Forests

- A recent algorithm is based on *random spanning forests* on graphs [Bar+19].
- A rooted spanning forest on a graph:

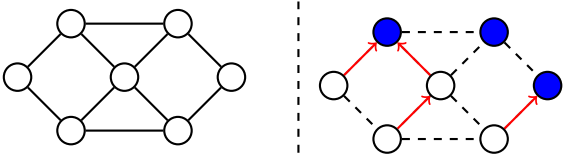


A rooted spanning forest

- Random spanning forests is the process of randomly selecting a spanning forest over all possible forests.

Inverse Trace Estimation: Random Spanning Forests

- A recent algorithm is based on *random spanning forests* on graphs [Bar+19].
- A rooted spanning forest on a graph:



A rooted spanning forest

- Random spanning forests is the process of randomly selecting a spanning forest over all possible forests.
- For a particular distribution [AG13], we have useful links with graph-related algebra.

Inverse Trace Estimation: Random Spanning Forests

- [Bar+19] propose to estimate tr(K) by the number of roots in RSFs, denoted by s.

Inverse Trace Estimation: Random Spanning Forests

- [Bar+19] propose to estimate $\mathrm{tr}(\mathsf{K})$ by the number of roots in RSFs, denoted by s.
- It is an unbiased estimator with a tractable variance:

$$\mathbb{E}[s] = \mathrm{tr}(\mathsf{K}) \text{ with } \mathrm{Var}(s) = \mathrm{tr}(\mathsf{K} - \mathsf{K}^2).$$

Inverse Trace Estimation: Random Spanning Forests

- [Bar+19] propose to estimate $\mathrm{tr}(\mathsf{K})$ by the number of roots in RSFs, denoted by s.
- It is an unbiased estimator with a tractable variance:

$$\mathbb{E}[s] = \mathrm{tr}(\mathsf{K}) \text{ with } \mathrm{Var}(s) = \mathrm{tr}(\mathsf{K} - \mathsf{K}^2).$$

- Empirical results shows that is comparable with Girard's estimator.

Inverse Trace Estimation: Random Spanning Forests

- [Bar+19] propose to estimate $\mathrm{tr}(\mathsf{K})$ by the number of roots in RSFs, denoted by s.
- It is an unbiased estimator with a tractable variance:

$$\mathbb{E}[s] = \mathrm{tr}(\mathsf{K}) \text{ with } \mathrm{Var}(s) = \mathrm{tr}(\mathsf{K} - \mathsf{K}^2).$$

- Empirical results shows that is comparable with Girard's estimator.
- In this work, we give two ways of improving this estimator.

Gradient Descent Update as Control Variate

- Estimation of K can be considered as minimizing the loss following function:

$$L(\mathsf{S}) = \mathrm{tr}\left(\frac{1}{2}\mathsf{S}^\top \mathsf{K}^{-1}\mathsf{S} - \mathsf{S}\right)$$

Gradient Descent Update as Control Variate

- Estimation of K can be considered as minimizing the loss following function:

$$L(\mathsf{S}) = \mathrm{tr}\left(\frac{1}{2}\mathsf{S}^\top \mathsf{K}^{-1}\mathsf{S} - \mathsf{S}\right)$$

- The gradient descent algorithm draws the following iteration:

$$\mathsf{S}_{k+1} = \mathsf{S}_k - \alpha(\mathsf{K}^{-1}\mathsf{S}_k - \mathsf{I}).$$

  where α is the update size.
- In [Pil+21b], we give two unbiased estimators $\tilde{\mathsf{S}}$ and $\bar{\mathsf{S}}$.

Gradient Descent Update as Control Variate

- Estimation of K can be considered as minimizing the loss following function:

$$L(\mathsf{S}) = \operatorname{tr}\left(\frac{1}{2}\mathsf{S}^\top \mathsf{K}^{-1}\mathsf{S} - \mathsf{S}\right)$$

- The gradient descent algorithm draws the following iteration:

$$\mathsf{S}_{k+1} = \mathsf{S}_k - \alpha(\mathsf{K}^{-1}\mathsf{S}_k - \mathsf{I}).$$

  where α is the update size.
- In [Pil+21b], we give two unbiased estimators $\tilde{\mathsf{S}}$ and $\bar{\mathsf{S}}$.
- Then, we improve them as follows [Pil+21a]:

$$\tilde{\mathsf{Z}} := \tilde{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\tilde{\mathsf{S}} - \mathsf{I})$$
$$\bar{\mathsf{Z}} := \bar{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I})$$

Gradient Descent Update as Control Variate

- Estimation of K can be considered as minimizing the loss following function:

$$L(\mathsf{S}) = \mathrm{tr}\left(\frac{1}{2}\mathsf{S}^\top \mathsf{K}^{-1}\mathsf{S} - \mathsf{S}\right)$$

- The gradient descent algorithm draws the following iteration:

$$\mathsf{S}_{k+1} = \mathsf{S}_k - \alpha(\mathsf{K}^{-1}\mathsf{S}_k - \mathsf{I}).$$

  where α is the update size.
- In [Pil+21b], we give two unbiased estimators $\tilde{\mathsf{S}}$ and $\bar{\mathsf{S}}$.
- Then, we improve them as follows [Pil+21a]:

$$\tilde{\mathsf{Z}} := \tilde{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\tilde{\mathsf{S}} - \mathsf{I})$$
$$\bar{\mathsf{Z}} := \bar{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I})$$

- Focusing on the trace, we define $\tilde{s} := \mathrm{tr}(\tilde{\mathsf{Z}})$ and $\bar{s} := \mathrm{tr}(\bar{\mathsf{Z}})$.

Properties of these estimators

- Both $\tilde{s}$ and $\bar{s}$ are unbiased.

Properties of these estimators

- Both $\tilde{s}$ and $\bar{s}$ are unbiased.
- Additional computations are at time complexity $\mathcal{O}(|\mathcal{E}|)$ per sample.

Properties of these estimators

- Both $\tilde{s}$ and $\bar{s}$ are unbiased.
- Additional computations are at time complexity $\mathcal{O}(|\mathcal{E}|)$ per sample.
- For certain values of α, we have improved theoretical performance *i.e.* $\mathrm{Var}(s) \geq \mathrm{Var}(\tilde{s}) \geq \mathrm{Var}(\bar{s})$.

Properties of these estimators

- Both $\tilde{s}$ and $\bar{s}$ are unbiased.
- Additional computations are at time complexity $\mathcal{O}(|\mathcal{E}|)$ per sample.
- For certain values of α, we have improved theoretical performance *i.e.* $\mathsf{Var}(s) \geq \mathsf{Var}(\tilde{s}) \geq \mathsf{Var}(\bar{s})$.
- For example, the optimal value of α for $\bar{s}$ is:

$$\alpha^\star = \frac{\mathsf{Cov}(s, \mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}{\mathsf{Var}(\mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}.$$

Properties of these estimators

- Both $\tilde{s}$ and $\bar{s}$ are unbiased.
- Additional computations are at time complexity $\mathcal{O}(|\mathcal{E}|)$ per sample.
- For certain values of α, we have improved theoretical performance *i.e.* $\mathsf{Var}(s) \geq \mathsf{Var}(\tilde{s}) \geq \mathsf{Var}(\bar{s})$.
- For example, the optimal value of α for $\bar{s}$ is:

$$\alpha^\star = \frac{\mathsf{Cov}(s, \mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}{\mathsf{Var}(\mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}.$$

- One can either choose a value for α from the safe range ($e.g.$ $\alpha = \frac{2q}{q+2d_{max}}$) or estimate from the samples:

$$\hat{\alpha} = \frac{\widehat{\mathsf{Cov}}(s, \mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}{\widehat{\mathsf{Var}}(\mathsf{tr}(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}))}.$$

Variance Reduction via Stratification

- Consider a random variable Y with an outcome set $\Omega = \cup_{k=1}^{K} C_k$.

Variance Reduction via Stratification

- Consider a random variable Y with an outcome set $\Omega = \cup_{k=1}^{K} C_k$.
- We assume:
  - $\mathbb{P}(Y \in C_k)$ is accessible,
  - $s | Y \in C_k$ is easy to sample.

Variance Reduction via Stratification

- Consider a random variable Y with an outcome set $\Omega = \cup_{k=1}^{K} C_k$.
- We assume:
  - $\mathbb{P}(Y \in C_k)$ is accessible,
  - $s|Y \in C_k$ is easy to sample.
- Then the stratified sampling takes the following form:

$$s_{st} := \sum_{k=1}^{K} \left(\frac{1}{N_k} \sum_{j=1}^{N_k} s^{(j)} | Y \in C_k \right) \mathbb{P}(Y \in C_k).$$

Variance Reduction via Stratification

- Consider a random variable Y with an outcome set $\Omega = \cup_{k=1}^{K} C_k$.
- We assume:
  - $\mathbb{P}(Y \in C_k)$ is accessible,
  - $s|Y \in C_k$ is easy to sample.
- Then the stratified sampling takes the following form:

$$s_{st} := \sum_{k=1}^{K} \left(\frac{1}{N_k} \sum_{j=1}^{N_k} s^{(j)} | Y \in C_k \right) \mathbb{P}(Y \in C_k).$$

- For certain allocations N_k's, s_{st} has a reduced variance.

Variance Reduction via Stratification

- Consider a random variable Y with an outcome set $\Omega = \cup_{k=1}^{K} C_k$.
- We assume:
  - $\mathbb{P}(Y \in C_k)$ is accessible,
  - $s|Y \in C_k$ is easy to sample.
- Then the stratified sampling takes the following form:

$$s_{st} := \sum_{k=1}^{K} \left(\frac{1}{N_k} \sum_{j=1}^{N_k} s^{(j)} | Y \in C_k \right) \mathbb{P}(Y \in C_k).$$

- For certain allocations N_k's, s_{st} has a reduced variance.
- We find a such random variable Y in RSFs!

Comparison with SOTA

- We compare the proposed algorithms with Hutchinson's estimator combined with several linear solvers.

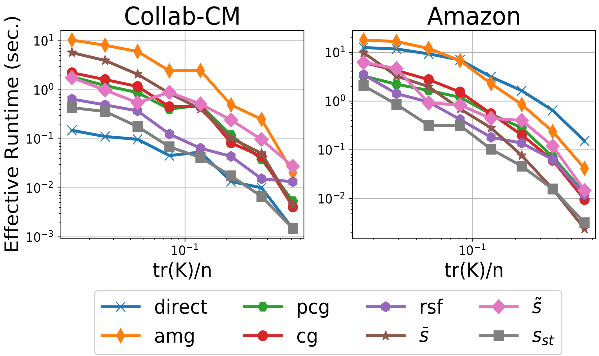


Figure: Effective Runtime vs $\mathrm{tr}(K)/n$.

Conclusion

- Random spanning forests are useful for randomized linear algebra involving SDD matrices.

Conclusion

- Random spanning forests are useful for randomized linear algebra involving SDD matrices.
- We propose two ways of improving the forest-based trace estimator.

Conclusion

- Random spanning forests are useful for randomized linear algebra involving SDD matrices.
- We propose two ways of improving the forest-based trace estimator.
- We validate these methods over real-life datasets.

Conclusion

- Random spanning forests are useful for randomized linear algebra involving SDD matrices.
- We propose two ways of improving the forest-based trace estimator.
- We validate these methods over real-life datasets.
- We hope to extend these results for estimating other Laplacian-based quantities, such as effective resistances.

References

Luca Avena and Alexandre Gaudillière. “Random spanning forests, Markov matrix spectra and well distributed points”. In: *arXiv preprint arXiv:1310.1723* (2013).

Simon Barthelmé et al. “Estimating the inverse trace using random forests on graphs”. In: *arXiv preprint arXiv:1905.02086* (2019).

Yusuf Pilavcı et al. “Variance reduction in stochastic methods for large-scale regularised least-squares problems”. In: *arXiv preprint arXiv:2110.07894* (2021).

Yusuf Yiğit Pilavcı et al. “Graph tikhonov regularization and interpolation via random spanning forests”. In: *IEEE transactions on Signal and Information Processing over Networks* 7 (2021), pp. 359–374.

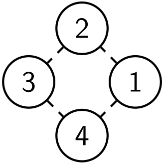
Questions

If you are hiring post-docs, scan me!

Thanks! Questions?

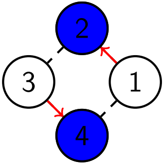
Estimating K with Forests

- We previously proposed two estimators for K [Pil+21b]:



Estimating K with Forests

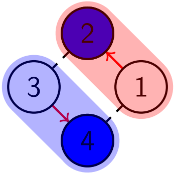
- We previously proposed two estimators for K [Pil+21b]:



$$\tilde{S} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Estimating K with Forests

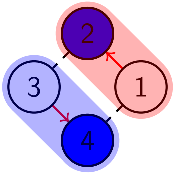
- We previously proposed two estimators for K [Pil+21b]:



$$\tilde{S} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad \bar{S} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix} \end{array}$$

Estimating K with Forests

- We previously proposed two estimators for K [Pil+21b]:



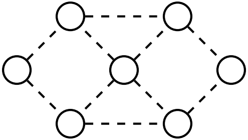
$$\tilde{S} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad \bar{S} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix} \end{array}$$

- Both are unbiased with tractable variances.

Variance Reduction via Stratification

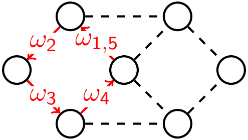
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



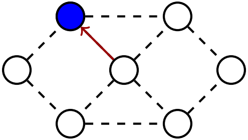
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



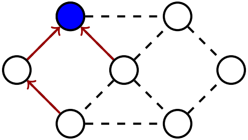
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



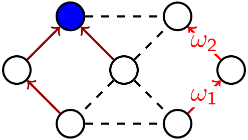
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



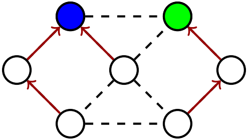
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



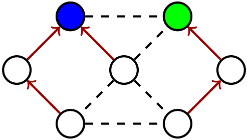
Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



Variance Reduction via Stratification

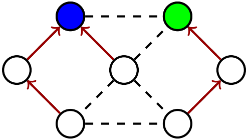
- The algorithm for sampling RSFs works in the following way:



- We choose our stratification variable $Y = s'$ as the number of roots sampled at *the first sight*:

Variance Reduction via Stratification

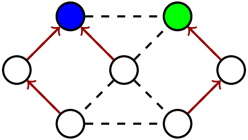
- The algorithm for sampling RSFs works in the following way:



- We choose our stratification variable $Y = s'$ as the number of roots sampled at *the first sight*:
  - $s' \sim \sum_{i=1}^{n} \text{Ber}\left(\frac{q}{q+d_i}\right)$

Variance Reduction via Stratification

- The algorithm for sampling RSFs works in the following way:



- We choose our stratification variable $Y = s'$ as the number of roots sampled at *the first sight*:
  - $s' \sim \sum_{i=1}^{n} \text{Ber}\left(\frac{q}{q+d_i}\right)$
  - Sampling RSFs given $s' \in C_k$ is easy.

Inverse Trace Estimation: Hutchinson's Estimator

- SOTA for estimating tr(K) is Hutchinson's estimator []:

$$h := \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}^{(i)\top} \mathsf{K} \mathbf{a}^{(i)}$$

where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.

Inverse Trace Estimation: Hutchinson's Estimator

- SOTA for estimating tr(K) is Hutchinson's estimator []:

$$h := \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}^{(i)\top} \mathsf{K} \mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.

Inverse Trace Estimation: Hutchinson's Estimator

- SOTA for estimating tr(K) is Hutchinson's estimator []:

$$h := \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}^{(i)\top} \mathrm{K} \mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- The cumbersome computation here is $\mathrm{K}\mathbf{a}^{(i)}$ for N vectors.

Inverse Trace Estimation: Hutchinson's Estimator

- SOTA for estimating tr(K) is Hutchinson's estimator []:

$$h := \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}^{(i)\top} \mathrm{K}\mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- The cumbersome computation here is $\mathrm{K}\mathbf{a}^{(i)}$ for N vectors.
- It can be done via:
  - Direct computation via Cholesky decomposition
  - (Preconditioned) Iterative solvers
  - Algebraic Multigrid solvers
  - ...