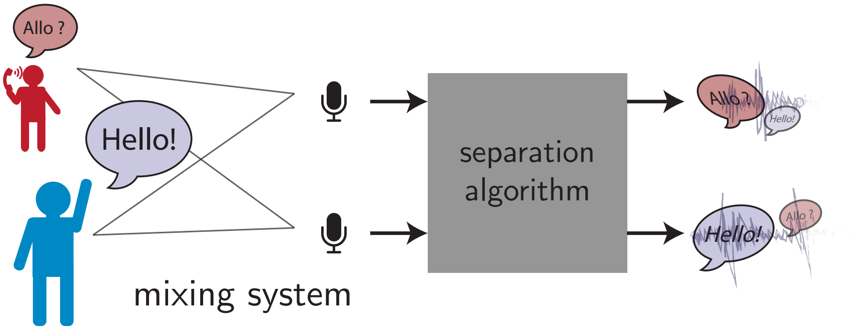
SDR —Medium Rare with Fast Computations

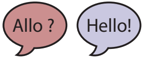
Robin Scheibler

April 21, 2022

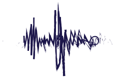
LINE

Evaluation of Source Separation Algorithms





signals/noise with reference available



artifact: unknown noise and algorithmic distortion

Goal Analyze contribution of each component in output.

Background: BSS Eval Metrics

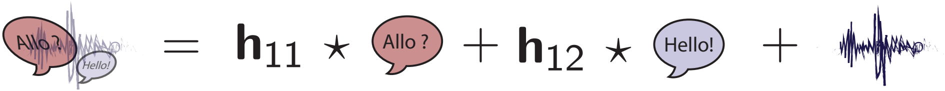
Signal Model

The following signals of length T are available

- Reference signals: $\mathbf{s}_m$, $k = 1, \dots, M$
- Estimated signals: $\hat{\mathbf{s}}_m$, $k = 1, \dots, M$

$$\hat{\mathbf{s}}_m = \sum_k \mathbf{h}_{km} \star \mathbf{s}_k + \mathbf{b}_m, \quad m = 1, \dots, M \tag{1}$$

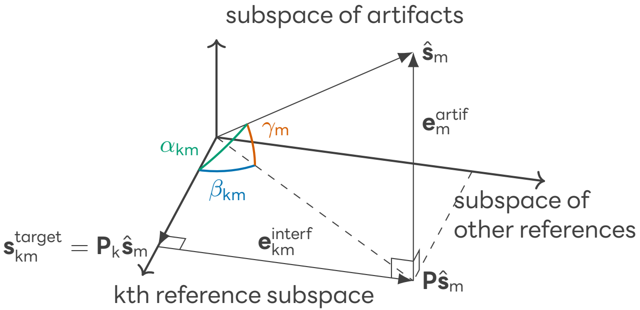
- The **artifact** term is $\mathbf{b}_m$, there is no reference available
- The **unknown** filters $\mathbf{h}_{km}$ of length L model reverb, calibration error, etc.



BSS Eval Metrics —Idea

Ref. Vincent et al., Performance measurement in blind source separation, TASLP, 2006. (**2875 citations**, and counting)

Decomposes the estimated signals in three **orthognal parts**



- $\mathbf{s}_k^{target}$: contribution of reference k
- $\mathbf{e}_{km}^{interf}$: contribution of other sources
- $\mathbf{e}_m^{artif}$: contribution of artifacts

BSS Eval Metrics —Definitions

BSS Eval defines three metrics

- Signal-to-Distortion Ratio (SDR)

$$\mathrm{SDR}_{km} = 10\log_{10}\frac{\|\mathbf{s}_{km}^{\mathrm{target}}\|^2}{\|\mathbf{e}_{km}^{\mathrm{interf}} + \mathbf{e}_{km}^{\mathrm{artif}}\|^2}$$

- Signal-to-Interference Ratio (SIR)

$$\mathrm{SIR}_{km} = 10\log_{10}\frac{\|\mathbf{s}_{km}^{\mathrm{target}}\|^2}{\|\mathbf{e}_{km}^{\mathrm{interf}}\|^2}$$

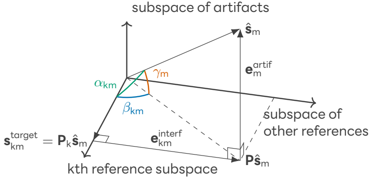
- Signal-to-Artefact Ratio (SDR)

$$\mathrm{SAR}_{km} = 10\log_{10}\frac{\|\mathbf{s}_{km}^{\mathrm{target}} + \mathbf{e}_{km}^{\mathrm{interf}}\|^2}{\|\mathbf{e}_{m}^{\mathrm{artif}}\|^2}$$

BSS Eval Metrics —Conventional Computations

Most toolboxes follow definition:

1. Compute $\mathbf{s}_{km}^{\text{target}}$, $\mathbf{e}_{km}^{\text{interf}}$, $\mathbf{e}_{m}^{\text{artif}}$
2. Apply definition of SDR, SIR, SAR



Required Computations

M: number of signals, T: signal length, L filter size

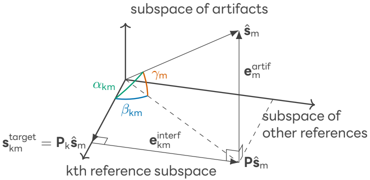
1. Compute statistics of ref./est. $O(M^2 T \log T)$
2. Solve large linear systems $O(ML^3)/O((ML)^3)$ (Gaussian elimination)
3. Filter signals $O(M^2 T \log T)$

We propose an efficient implementation of BSS Eval!

BSS Eval Metrics —Conventional Computations

Most toolboxes follow definition:

1. Compute $\mathbf{s}_{km}^{target}$, $\mathbf{e}_{km}^{interf}$, $\mathbf{e}_{m}^{artif}$
2. Apply definition of SDR, SIR, SAR



Required Computations

M: number of signals, T: signal length, L filter size

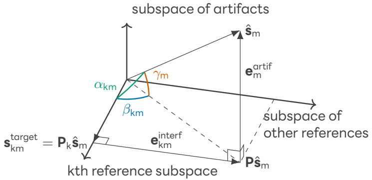
1. Compute statistics of ref./est. $O(M^2 T \log T)$
2. Solve large linear systems $O(ML^3)/O((ML)^3)$ (Gaussian elimination)
3. Filter signals $O(M^2 T \log T)$

We propose an efficient implementation of BSS Eval!

BSS Eval Metrics —Conventional Computations

Most toolboxes follow definition:

1. Compute $\mathbf{s}_{km}^{target}$, $\mathbf{e}_{km}^{interf}$, $\mathbf{e}_{m}^{artif}$
2. Apply definition of SDR, SIR, SAR



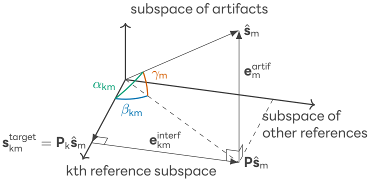
Required Computations

M: number of signals, T: signal length, L filter size

1. Compute statistics of ref./est. $O(M^2 T \log T)$
2. Solve large linear systems $O(ML^3)/O((ML)^3)$ (Gaussian elimination)
3. Filter signals $O(M^2 T \log T)$

We propose an efficient implementation of BSS Eval!

BSS Eval Metrics —Conventional Computations



Most toolboxes follow definition:

1. Compute $\mathbf{s}_{km}^{target}$, $\mathbf{e}_{km}^{interf}$, $\mathbf{e}_m^{artif}$
2. Apply definition of SDR, SIR, SAR

Required Computations

M: number of signals, T: signal length, L filter size

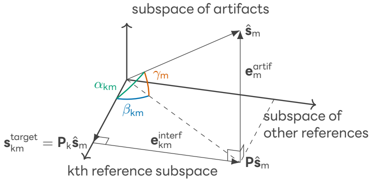
1. Compute statistics of ref./est. $O(M^2 T \log T)$
2. Solve large linear systems $O(ML \log L)/O(M^3 L \log L)$ **(Conjugate Gradient)**
3. Filter signals $O(M^2 T \log T)$

We propose an efficient implementation of BSS Eval!

BSS Eval Metrics —Conventional Computations

Most toolboxes follow definition:

1. Compute $\mathbf{s}_{km}^{\text{target}}$, $\mathbf{e}_{km}^{\text{interf}}$, $\mathbf{e}_{m}^{\text{artif}}$
2. Apply definition of SDR, SIR, SAR



Required Computations

M: number of signals, T: signal length, L filter size

1. Compute statistics of ref./est. $O(M^2 T \log T)$
2. Solve large linear systems $O(ML \log L)$/$O(M^3 L \log L)$ **(Conjugate Gradient)**
3. ~~Filter signals $O(M^2 T \log T)$~~ $\rightarrow O(ML)$

We propose an efficient implementation of BSS Eval!

Fast BSS Eval

Main New Insight

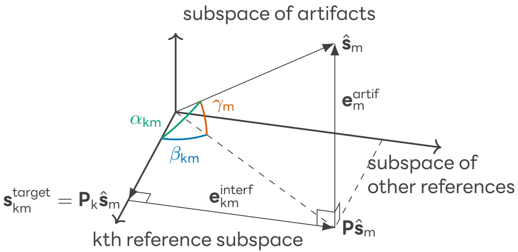
The metrics are functions of the **subspace angles**!

Theorem

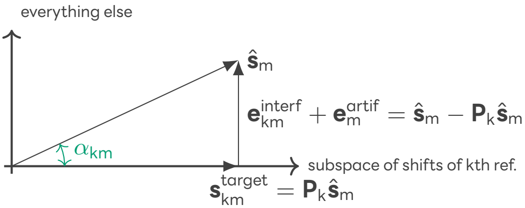
$$\mathrm{SDR}_{\mathrm{km}} = -10 \log_{10} \tan^2 \alpha_{\mathrm{km}}$$

$$\mathrm{SIR}_{\mathrm{km}} = -10 \log_{10} \tan^2 \beta_{\mathrm{km}}$$

$$\mathrm{SAR}_{\mathrm{m}} = -10 \log_{10} \tan^2 \gamma_{\mathrm{km}}$$



Proof (SDR only)

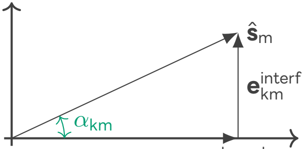


Recall definition of SDR, wlog $\|\hat{\mathbf{s}}_m\| = 1$

$$\text{SDR}_{km} = 10 \log_{10} \frac{\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2}{\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2} = 10 \log_{10} \frac{\cos^2 \alpha_{km}}{1 - \cos^2 \alpha_{km}}$$

1. Definition of cosine: $\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \cos^2 \alpha_{km}$
2. Pythagor: $\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \|\hat{\mathbf{s}}_m\|^2 - \|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = 1 - \cos^2 \alpha_{km}$

Proof (SDR only)

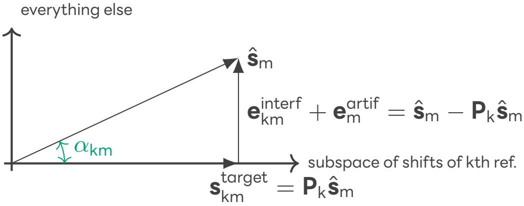


Recall definition of SDR, wlog $\|\hat{\mathbf{s}}_m\| = 1$

$$\mathrm{SDR}_{km} = 10\log_{10}\frac{\|\mathbf{P}_k\hat{\mathbf{s}}_m\|^2}{\|\hat{\mathbf{s}}_m - \mathbf{P}_k\hat{\mathbf{s}}_m\|^2} = 10\log_{10}\frac{\cos^2\alpha_{km}}{1-\cos^2\alpha_{km}}$$

1. Definition of cosine: $\|\mathbf{P}_k\hat{\mathbf{s}}_m\|^2 = \cos^2\alpha_{km}$
2. Pythagor: $\|\hat{\mathbf{s}}_m - \mathbf{P}_k\hat{\mathbf{s}}_m\|^2 = \|\hat{\mathbf{s}}_m\|^2 - \|\mathbf{P}_k\hat{\mathbf{s}}_m\|^2 = 1 - \cos^2\alpha_{km}$

Proof (SDR only)

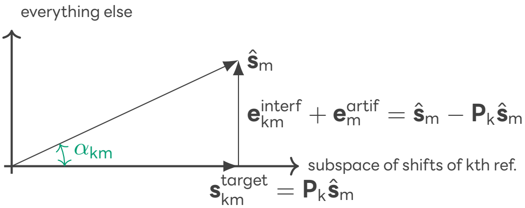


Recall definition of SDR, wlog $\|\hat{\mathbf{s}}_m\| = 1$

$$\text{SDR}_{km} = 10 \log_{10} \frac{\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2}{\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2} = 10 \log_{10} \frac{\cos^2 \alpha_{km}}{1 - \cos^2 \alpha_{km}}$$

1. Definition of cosine: $\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \cos^2 \alpha_{km}$
2. Pythagor: $\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \|\hat{\mathbf{s}}_m\|^2 - \|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = 1 - \cos^2 \alpha_{km}$

Proof (SDR only)



Recall definition of SDR, wlog $\|\hat{\mathbf{s}}_m\| = 1$

$$\text{SDR}_{km} = 10 \log_{10} \frac{\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2}{\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2} = 10 \log_{10} \frac{\cos^2 \alpha_{km}}{1 - \cos^2 \alpha_{km}}$$

1. Definition of cosine: $\|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \cos^2 \alpha_{km}$
2. Pythagor: $\|\hat{\mathbf{s}}_m - \mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = \|\hat{\mathbf{s}}_m\|^2 - \|\mathbf{P}_k \hat{\mathbf{s}}_m\|^2 = 1 - \cos^2 \alpha_{km}$

Fast Computation

The projection matrix $\mathbf{P}_k$ onto the shifts of reference $\mathbf{s}_k$ is

$$\mathbf{P}_k = \mathbf{A}_k(\mathbf{A}_k^\top \mathbf{A}_k)^{-1}\mathbf{A}_k^\top$$

where $\mathbf{A}_k$ contains shifted versions of $\mathbf{s}_k$ in its columns
For the SDR, we only need to compute

$$\|\mathbf{P}_k\hat{\mathbf{s}}_m\|^2 = (\mathbf{A}_k^\top\hat{\mathbf{s}}_m)^\top(\mathbf{A}_k^\top\mathbf{A}_k)^{-1}(\mathbf{A}_k^\top\hat{\mathbf{s}}_m)$$

Proposed Algorithm (SDR)

1. Compute $\mathbf{R}_k = \mathbf{A}_k^\top\mathbf{A}_k$ and $\mathbf{x}_{km} = \mathbf{A}_k^\top\hat{\mathbf{s}}_m$
2. Solve $\mathbf{R}_k\mathbf{h} = \mathbf{x}_{km}$, this is a **Toeplitz** system
3. Compute $\cos^2\alpha_{km} = \mathbf{x}_{km}^\top\mathbf{h}$
4. $\text{SDR}_{km} = 10\log_{10}\frac{\cos^2\alpha_{km}}{1-\cos^2\alpha_{km}}$

Solve $\mathbf{R}_k \mathbf{h} = \mathbf{x}_{km}$

The $L \times L$ matrix $\mathbf{R}_k$ is **Toeplitz**, there are fast solvers!

- Conjugate Gradient Algorithm
- Multiplication by $\mathbf{R}_k$ in $O(L \log L)$ via FFT
- Circulant pre-conditioner, also $O(L \log L)$ via FFT
- Eigenvalues cluster around 1, and converges in few iterations [Chan1996]

Reference

R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems," SIAM Rev., vol. 38, no. 3, pp. 427–482, Sep. 1996.

Solve $R_k h = x_{km}$

The $L \times L$ matrix $\mathbf{R}_k$ is **Toeplitz**, there are fast solvers!

- Conjugate Gradient Algorithm
- Multiplication by $\mathbf{R}_k$ in $O(L \log L)$ via FFT
- Circulant pre-conditioner, also $O(L \log L)$ via FFT
- Eigenvalues cluster around 1, and converges in few iterations [Chan1996]

Reference

R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems,"SIAM Rev., vol. 38, no. 3, pp. 427–482, Sep. 1996.

Solve $\mathbf{R}_k \mathbf{h} = \mathbf{x}_{km}$

The $L \times L$ matrix $\mathbf{R}_k$ is **Toeplitz**, there are fast solvers!

- Conjugate Gradient Algorithm
- Multiplication by $\mathbf{R}_k$ in $O(L \log L)$ via FFT
- Circulant pre-conditioner, also $O(L \log L)$ via FFT
- Eigenvalues cluster around 1, and converges in few iterations [Chan1996]

Reference

R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems,"SIAM Rev., vol. 38, no. 3, pp. 427–482, Sep. 1996.

Solve $\mathbf{R}_k \mathbf{h} = \mathbf{x}_{km}$

The $L \times L$ matrix $\mathbf{R}_k$ is **Toeplitz**, there are fast solvers!

- Conjugate Gradient Algorithm
- Multiplication by $\mathbf{R}_k$ in $O(L \log L)$ via FFT
- Circulant pre-conditioner, also $O(L \log L)$ via FFT
- Eigenvalues cluster around 1, and converges in few iterations [Chan1996]

Reference

R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems,"SIAM Rev., vol. 38, no. 3, pp. 427–482, Sep. 1996.

Solve $\mathbf{R}_k \mathbf{h} = \mathbf{x}_{km}$

The $L \times L$ matrix $\mathbf{R}_k$ is **Toeplitz**, there are fast solvers!

- Conjugate Gradient Algorithm
- Multiplication by $\mathbf{R}_k$ in $O(L \log L)$ via FFT
- Circulant pre-conditioner, also $O(L \log L)$ via FFT
- Eigenvalues cluster around 1, and converges in few iterations [Chan1996]

Reference

R. H. Chan and M. K. Ng, "Conjugate Gradient Methods for Toeplitz Systems," SIAM Rev., vol. 38, no. 3, pp. 427–482, Sep. 1996.

Experiments

Implementation

Python implementation in `fast-bss-eval` package

```
pip install fast-bss-eval
```

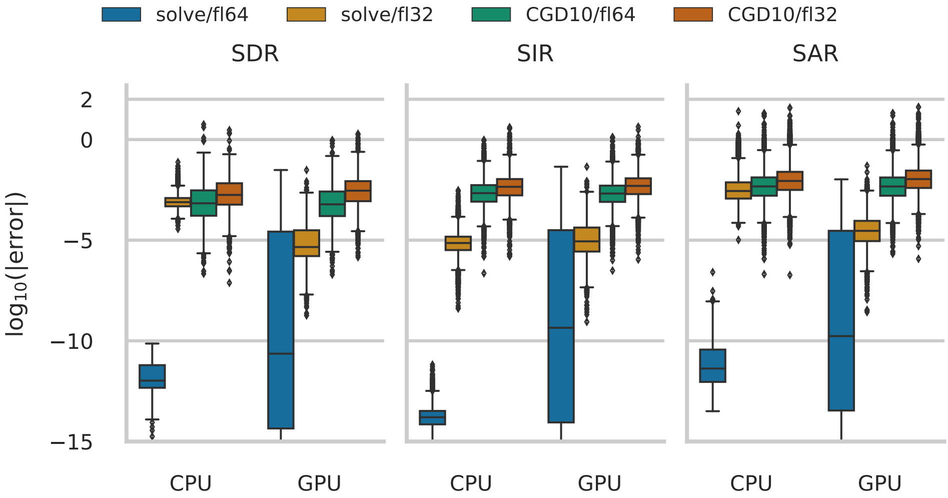
- Supports numpy/torch transparently
- Differentiable via torch
- Options for Gaussian eliminiation / conjugate gradient
- Improved numerical stability

Baselines

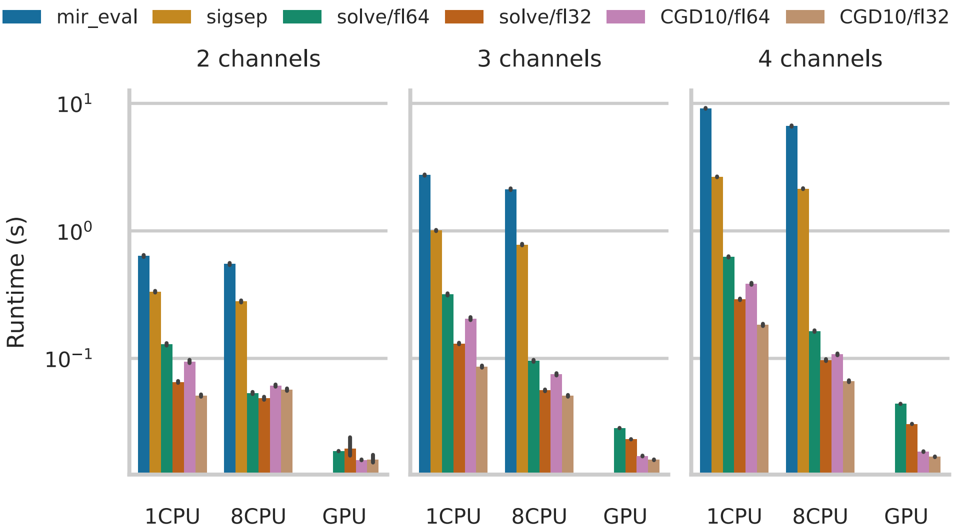
package	mir_eval	sigsep	ci_sdr
metrics	SDR/SIR/SAR	SDR/SIR/SAR	SDR only
backend	numpy	numpy	torch
reference	[Raffel2014]	github/sigsep	[Boeddeker2021]

Testing Accuracy

error wrt `mir_eval`
dataset: WSJ1 (test), 2/3/4 speakers, noise from CHiME3



Speed Contest



Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```

Conclusion

Main Contributions

A **fast** algorithm for BSS Eval, to evaluate source separation algorithms

- New insights into BSS Eval metrics as subspace angles
- Reduced operation count
- Fast Toeplitz solver
- Python package compatible with numpy/torch
- Orders of magnitude speed-up compared to existing packages

Use it: pip install fast-bss-eval

```
from fast_bss_eval import bss_eval_sources
sdr, sir, sar, perm = bss_eval_sources(ref, est)
```