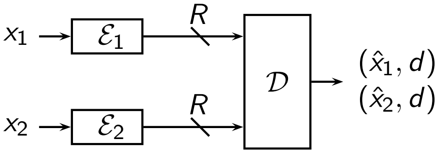
Integer-Forcing Source Coding

Or Ordentlich
Joint work with Uri Erez

June 30th, 2014
ISIT, Honolulu, HI, USA

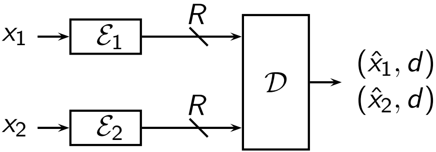
Motivation 1 - Universal Quantization

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$



Motivation 1 - Universal Quantization

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$

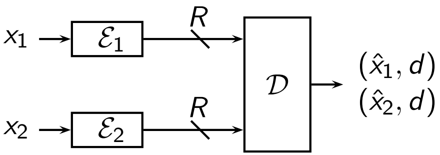


Goal:

- Simple, identical, universal, non-cooperating quantizers $\mathcal{E}_1, \mathcal{E}_2$
- Simple decoder $\mathcal{D}$ that can depend on $\mathbf{K}_{xx}$
- Good performance for all $\mathbf{K}_{xx}$ with the same $\log\det\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}\right)$

Motivation 1 - Universal Quantization

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$



Goal:

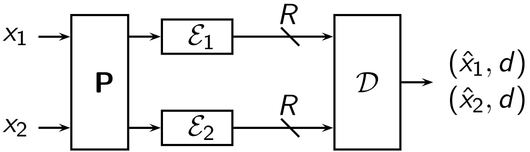
- Simple, identical, universal, non-cooperating quantizers $\mathcal{E}_1, \mathcal{E}_2$
- Simple decoder $\mathcal{D}$ that can depend on $\mathbf{K}_{xx}$
- Good performance for all $\mathbf{K}_{xx}$ with the same $\log\det\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}\right)$

Extreme cases:

$$\mathbf{K}^1_{xx} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{K}^2_{xx} = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbf{K}^3_{xx} = \begin{bmatrix} b & b \\ b & b \end{bmatrix}$$

Motivation 1 - Universal Quantization

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$



Goal:

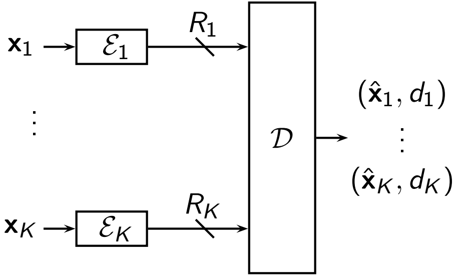
- Simple, identical, universal, non-cooperating quantizers $\mathcal{E}_1, \mathcal{E}_2$
- Simple decoder $\mathcal{D}$ that can depend on $\mathbf{K}_{xx}$
- Good performance for all $\mathbf{K}_{xx}$ with the same $\log\det\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}\right)$

Extreme cases:

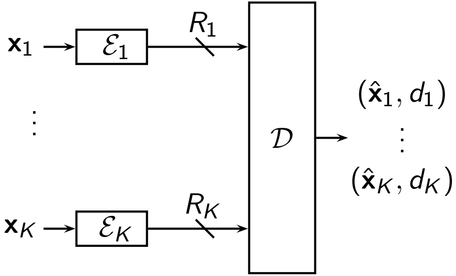
$$\mathbf{K}_{xx}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{K}_{xx}^2 = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbf{K}_{xx}^3 = \begin{bmatrix} b & b \\ b & b \end{bmatrix}$$

Willing to apply a **universal** linear transformation before quantization

Motivation 2 -Distributed Lossy Compression

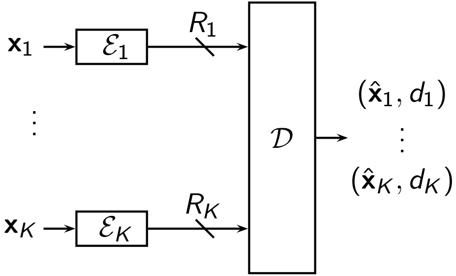


Motivation 2 -Distributed Lossy Compression



- Fundamental limits understood in some cases
- Inner and outer bounds known

Motivation 2 -Distributed Lossy Compression

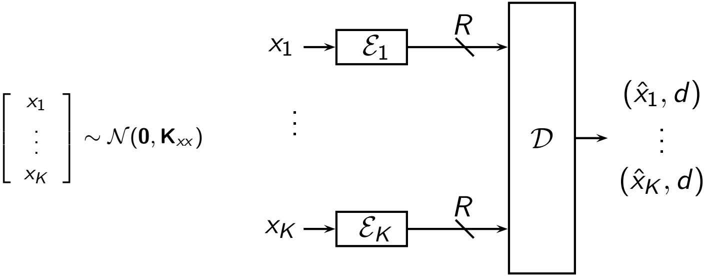


- Fundamental limits understood in some cases
- Inner and outer bounds known

Some applications require

- Extremely simple encoders/decoder
- Extremely short delay

Motivation 2 -Distributed Lossy Compression



We restrict attention to:

- Gaussian sources $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$
- One-shot compression - block length is 1
- Symmetric rates $R_1 = \cdots = R_K = R$
- Symmetric distortions $d_1 = \cdots = d_K = d$
- MSE distortion measure: $E(x_k - \hat{x}_k)^2 \leq d$

Goal and Means

Goal

- Simple encoders: uniform scalar quantizers
- Decoupled decoding
- Performance close to best known inner bounds (Berger-Tung)

Goal and Means

Goal

- Simple encoders: uniform scalar quantizers
- Decoupled decoding
- Performance close to best known inner bounds (Berger-Tung)

Binning:

- Well understood for large blocklengths, less for short blocks
- Requires sophisticated joint decoding techniques

Goal and Means

Goal

- Simple encoders: uniform scalar quantizers
- Decoupled decoding
- Performance close to best known inner bounds (Berger-Tung)

Binning:

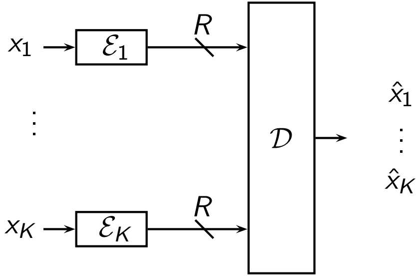
- Well understood for large blocklengths, less for short blocks
- Requires sophisticated joint decoding techniques

Scalar Modulo

- A simple 1-D binning operation
- Allows for efficient decoding using integer-forcing

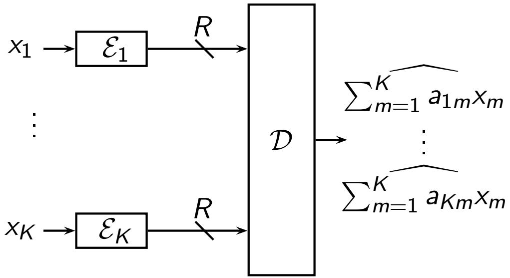
Integer-Forcing Source Coding: Overview

Basic Idea: Rather than solving the problem



Integer-Forcing Source Coding: Overview

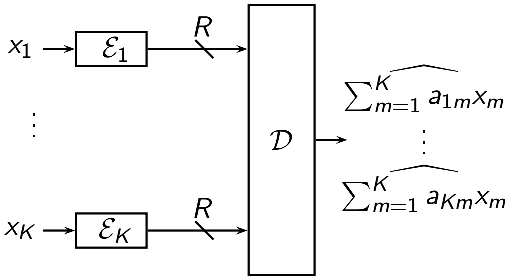
First solve



and then invert equations to get $\hat{x}_1, \ldots, \hat{x}_K$

Integer-Forcing Source Coding: Overview

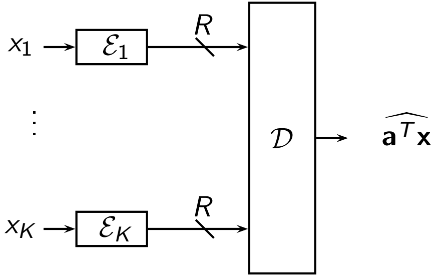
First solve



and then invert equations to get $\hat{x}_1, \ldots, \hat{x}_K$

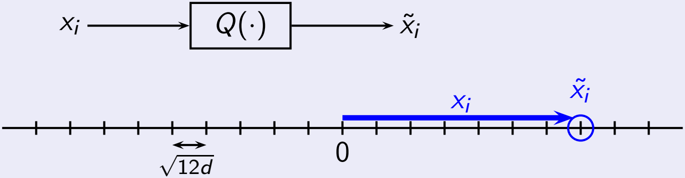
- Problem reduces to simultaneous distributed compression of K linear combinations
- Can be efficiently solved with small rates for certain choices of coefficients
- Equation coefficients can be chosen to optimize performance

Distributed Compression of Integer Linear Combination



Distributed Compression of Integer Linear Combination

Scalar Quantization



- High resolution/dithered quantization:

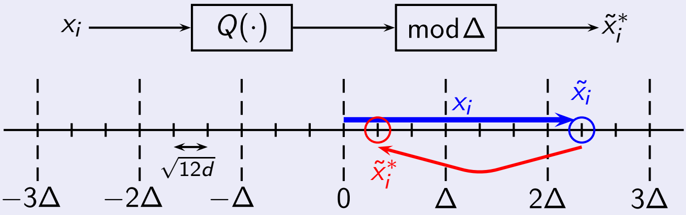
$$\tilde{x}_i = x_i + u_i$$

where $u_i \sim \mathrm{Uniform}\left(\left[-\frac{\sqrt{12d}}{2}, \frac{\sqrt{12d}}{2}\right)\right)$, $u_i \perp\!\!\!\perp x_i$

- $\mathbb{E}(\tilde{x}_i - x_i)^2 = d$

Distributed Compression of Integer Linear Combination

Modulo Scalar Quantization



- $\Delta = 2^R\sqrt{12d} \Longrightarrow$ Compression rate is R
- High resolution/dithered quantization:

$$\tilde{x}_i^* = [x_i + u_i]^*$$

Distributed Compression of Integer Linear Combination

Encoders

Each encoder is a modulo scalar quantizer with rate R : produces $\tilde{x}_k^*$

Distributed Compression of Integer Linear Combination

Encoders

Each encoder is a modulo scalar quantizer with rate R : produces $\tilde{x}_k^*$

Simple modulo property

For any set of integers $a_1, \ldots, a_K$ and real numbers $\tilde{x}_1, \ldots, \tilde{x}_K$

$$\left[\sum_{k=1}^{K} a_k \tilde{x}_k\right]^* = \left[\sum_{k=1}^{K} a_k \tilde{x}_k^*\right]^*$$

Distributed Compression of Integer Linear Combination

Encoders

Each encoder is a modulo scalar quantizer with rate R : produces $\tilde{x}_k^*$

Simple modulo property

For any set of integers $a_1, \ldots, a_K$ and real numbers $\tilde{x}_1, \ldots, \tilde{x}_K$

$$\left[\sum_{k=1}^{K} a_k \tilde{x}_k\right]^* = \left[\sum_{k=1}^{K} a_k \tilde{x}_k^*\right]^*$$

Decoder

- Gets: $\tilde{x}_1^*, \ldots, \tilde{x}_K^*$
- Outputs:

$$\widehat{\mathbf{a}^T\mathbf{x}} = \left[\sum_{k=1}^{K} a_k \tilde{x}_k^*\right]^* = \left[\sum_{k=1}^{K} a_k \tilde{x}_k\right]^* = \left[\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right]^*$$

Compression of Integer Linear Combination - P_e

$$\widehat{\mathbf{a}^T\mathbf{x}} = \left[\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right]^*$$

$$\widehat{\mathbf{a}^T\mathbf{x}} = \begin{cases} \mathbf{a}^T\mathbf{x} + \mathbf{a}^T\mathbf{u} & \text{if } \mathbf{a}^T(\mathbf{x}+\mathbf{u}) \in \left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right) \\ \text{error} & \text{otherwise} \end{cases}$$

- P_e is small if $\frac{\Delta}{\sqrt{\mathrm{Var}\left(\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right)}}$ is large
- Δ grows exponentially with R

Compression of Integer Linear Combination - P_e

$$\widehat{\mathbf{a}^T\mathbf{x}} = \left[\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right]^*$$

$$\widehat{\mathbf{a}^T\mathbf{x}} = \begin{cases} \mathbf{a}^T\mathbf{x} + \mathbf{a}^T\mathbf{u} & \text{if } \mathbf{a}^T(\mathbf{x}+\mathbf{u}) \in \left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right) \\ \text{error} & \text{otherwise} \end{cases}$$

- P_e is small if $\frac{\Delta}{\sqrt{\mathrm{Var}\left(\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right)}}$ is large
- Δ grows exponentially with R

$$P_e \leq 2\exp\left\{-\frac{3}{2}2^{2\left(R-\frac{1}{2}\log\left(\frac{\mathbf{a}^T(\mathbf{K}_{\mathbf{xx}}+d\mathbf{I})\mathbf{a}}{d}\right)\right)}\right\}$$

Compression of Integer Linear Combination - P_e

$$\widehat{\mathbf{a}^T\mathbf{x}} = \left[\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right]^*$$

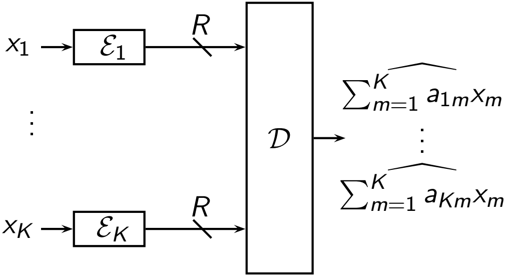
$$\widehat{\mathbf{a}^T\mathbf{x}} = \begin{cases} \mathbf{a}^T\mathbf{x} + \mathbf{a}^T\mathbf{u} & \text{if } \mathbf{a}^T(\mathbf{x}+\mathbf{u}) \in \left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right) \\ \text{error} & \text{otherwise} \end{cases}$$

- P_e is small if $\frac{\Delta}{\sqrt{\mathrm{Var}\left(\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right)}}$ is large
- Δ grows exponentially with R

$$P_e \leq 2\exp\left\{-\frac{3}{2}2^{2\left(R-\frac{1}{2}\log\left(\frac{\mathbf{a}^T(\mathbf{K_{xx}}+d\mathbf{I})\mathbf{a}}{d}\right)\right)}\right\}$$

For $\mathbf{a}$ with small $\mathrm{Var}\left(\mathbf{a}^T(\mathbf{x}+\mathbf{u})\right)$ we can take small R

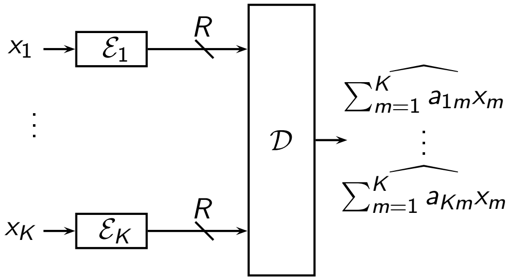
Integer-Forcing Source Coding



- Need to estimate K linearly independent integer linear combinations
- If all combinations estimated without error, can compute

$$\hat{\mathbf{x}} = \mathbf{A}^{-1}\widehat{\mathbf{Ax}} = \mathbf{A}^{-1}(\mathbf{Ax} + \mathbf{Au}) = \mathbf{x} + \mathbf{u}$$

Integer-Forcing Source Coding



- Need to estimate K linearly independent integer linear combinations
- If all combinations estimated without error, can compute

$$\hat{\mathbf{x}} = \mathbf{A}^{-1}\widehat{\mathbf{A}\mathbf{x}} = \mathbf{A}^{-1}(\mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{u}) = \mathbf{x} + \mathbf{u}$$

$$P_e \leq 2K \exp\left\{-\frac{3}{2}2^{2\left(R - \frac{1}{2}\log\left(\frac{\max_{m=1,\ldots,K} \mathbf{a}_m^T(\mathbf{K}_{\mathbf{xx}} + d\mathbf{I})\mathbf{a}_m}{d}\right)\right)}\right\}$$

Integer-Forcing Source Coding - Performance

Let

$$R_{\mathrm{IF}}(\mathbf{A}, d) \triangleq \frac{1}{2} \log \left(\max_{m=1,\ldots,K} \mathbf{a}_m^T \left(\mathbf{I} + \frac{1}{d} \mathbf{K}_{\mathbf{xx}} \right) \mathbf{a}_m \right)$$

Integer-Forcing Source Coding - Performance

Let

$$R_{\mathrm{IF}}(\mathbf{A}, d) \triangleq \frac{1}{2}\log\left(\max_{m=1,\ldots,K} \mathbf{a}_m^T \left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{\mathbf{xx}}\right)\mathbf{a}_m\right)$$

Theorem

Let $R = R_{\mathrm{IF}}(\mathbf{A}, d) + \delta$. IF source coding produces estimates with average MSE distortion d for all $x_1, \ldots, x_K$ with probability $> 1 - 2K\exp\left\{-\frac{3}{2}2^{2\delta}\right\}$

Integer-Forcing Source Coding - Performance

Let

$$R_{\text{IF}}(\mathbf{A}, d) \triangleq \frac{1}{2}\log\left(\max_{m=1,\ldots,K} \mathbf{a}_m^T\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{\mathbf{xx}}\right)\mathbf{a}_m\right)$$

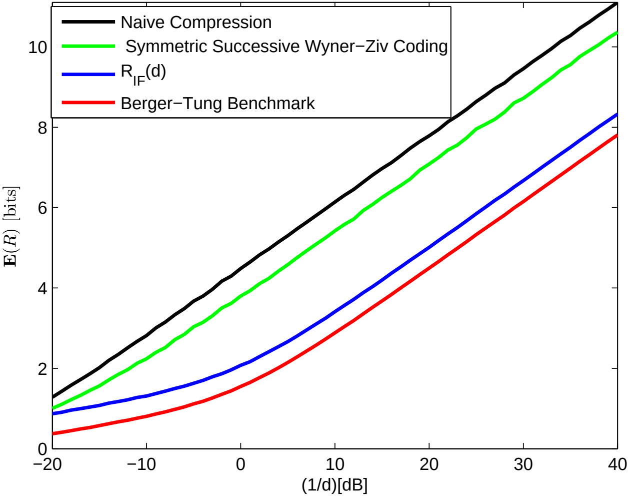
Theorem

Let $R = R_{\text{IF}}(\mathbf{A}, d) + \delta$. IF source coding produces estimates with average MSE distortion d for all $x_1, \ldots, x_K$ with probability $> 1 - 2K\exp\left\{-\frac{3}{2}2^{2\delta}\right\}$

Can minimize compression rate by minimizing $R_{\text{IF}}(\mathbf{A}, d)$ w.r.t. $\mathbf{A}$

Integer-Forcing Source Coding: Example

$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{xx}})$, $\mathbf{K_{xx}} = \mathbf{I} + \text{SNR}\mathbf{H}\mathbf{H}^T$, SNR $= 20$dB and $\mathbf{H} \in \mathbb{R}^{8\times 2}$



Back to Motivation 1

How close is $R_{\mathrm{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

How close is $R_{\mathrm{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

However, if we change the setting...

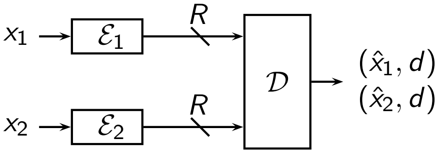
this obstacle can be overcome.

Back to Motivation 1

How close is $R_{\mathrm{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$

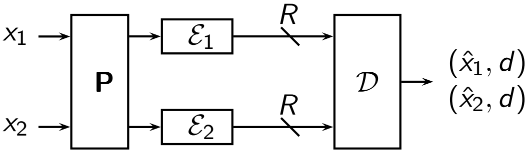


Back to Motivation 1

How close is $R_{\text{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$

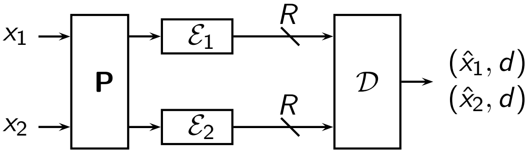


Back to Motivation 1

How close is $R_{\text{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$



Requirements

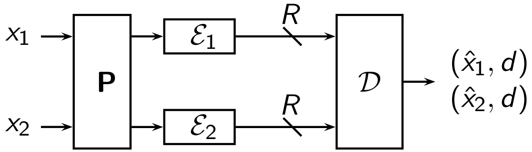
- Universal precoding matrix $\mathbf{P}$ (does not depend on $\mathbf{K}_{xx}$)
- $R_{\text{IF}}(d) \leq \text{const} + \frac{1}{2K}\log(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx})$ for all $\mathbf{K}_{xx}$

Back to Motivation 1

How close is $R_{\mathrm{IF}}(d)$ to the optimal performance?

- Usually very close to the performance of the Berger-Tung inner bound.
- But... the gap can be arbitrarily large.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{xx})$$

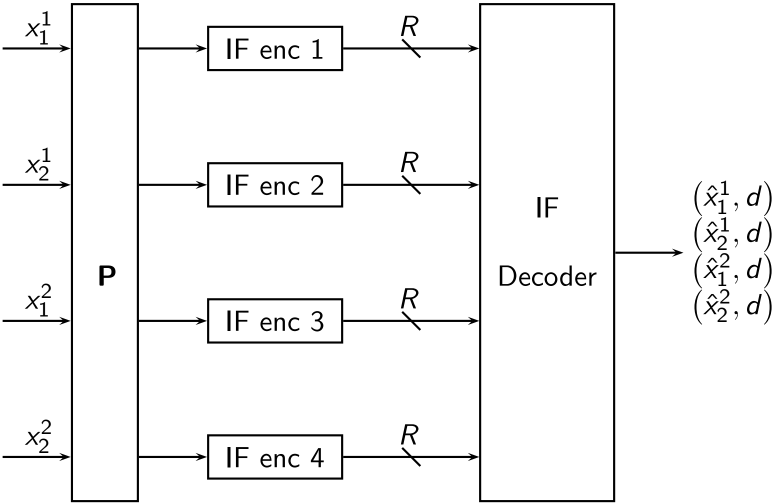


Requirements

- Universal precoding matrix $\mathbf{P}$ (does not depend on $\mathbf{K}_{xx}$)
- $R_{\mathrm{IF}}(d) \leq \text{const} + \frac{1}{2K}\log(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx})$ for all $\mathbf{K}_{xx}$

Price of universality - need to jointly encode K realizations

Space-Time Source Coding



Space-Time Source Coding - Performance Guarantees

Let **P** be a generating matrix of a "perfect" linear dispersion space-time code, with minimum det $\delta_{\min}(\mathcal{C}_\infty^{\text{ST}})$

Theorem

For any source with covariance matrix $\mathbf{K}_{\mathbf{xx}}$, the rate-distortion function of space-time integer-forcing source coding with precoding matrix **P** is bounded by

$$R_{\text{IF}}(d) < \frac{1}{2K} \log\det\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{\mathbf{xx}}\right) + \Gamma\left(K, \delta_{\min}(\mathcal{C}_\infty^{\text{ST}})\right)$$

where $\Gamma\left(K, \delta_{\min}(\mathcal{C}_\infty^{\text{ST}})\right) \triangleq 2K^2\log(2K^2) + K\log\frac{1}{\delta_{\min}(\mathcal{C}_\infty^{\text{ST}})}$

Remark: For $K = 2$ the golden-code precoding matrix has $\delta_{\min}(\mathcal{C}_\infty^{\text{ST}}) = 1/5$

Example

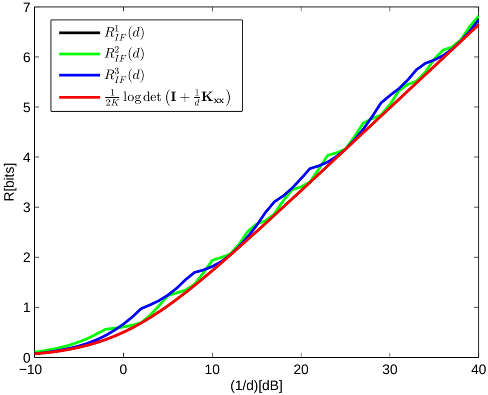
$$\mathbf{K}_{xx}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{K}_{xx}^2 = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbf{K}_{xx}^3 = \begin{bmatrix} b & b \\ b & b \end{bmatrix}$$

$$\frac{1}{2K}\log\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}^1\right) = \frac{1}{2K}\log\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}^2\right) = \frac{1}{2K}\log\left(\mathbf{I} + \frac{1}{d}\mathbf{K}_{xx}^3\right)$$

Example

$$\mathbf{K}_{xx}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{K}_{xx}^2 = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } \mathbf{K}_{xx}^3 = \begin{bmatrix} b & b \\ b & b \end{bmatrix}$$

$$\frac{1}{2K}\log\left(\mathbf{I}+\frac{1}{d}\mathbf{K}_{xx}^1\right) = \frac{1}{2K}\log\left(\mathbf{I}+\frac{1}{d}\mathbf{K}_{xx}^2\right) = \frac{1}{2K}\log\left(\mathbf{I}+\frac{1}{d}\mathbf{K}_{xx}^3\right)$$



Thanks for your attention!