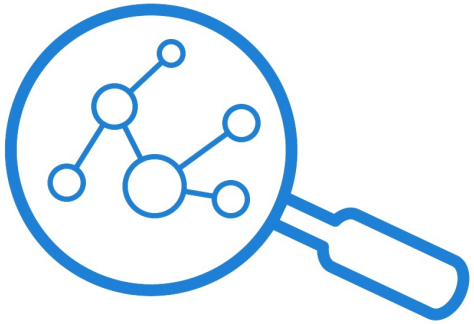
# 数据科学与大数据技术的数学基础

## 第十三讲



计算机学院
余皓然

2024/6/6

# 课程内容

| |
|---|
| **Part1 随机化方法** |
| 一致性哈希 布隆过滤器 CM Sketch方法 最小哈希<br>欧氏距离下的相似搜索 Jaccard相似度下的相似搜索 |
| **Part2 谱分析方法** |
| 主成分分析 奇异值分解 谱图论 |
| **Part3 最优化方法** |
| 压缩感知 |

# 压缩感知

## 信号恢复最优化问题

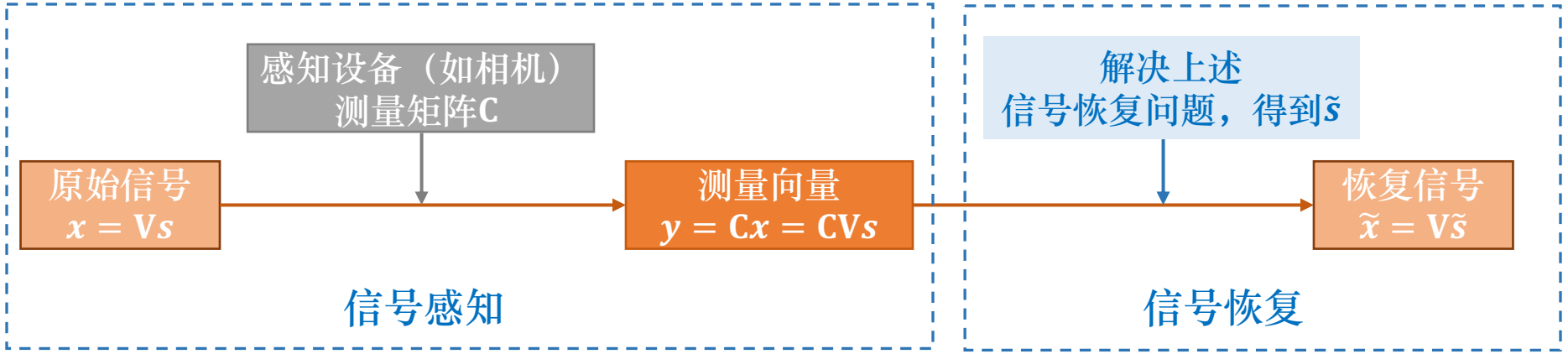
# 优化问题

**压缩感知中的信号恢复问题（Signal Recovery Problem in Compressive Sensing）**

给定$p\times n$测量矩阵$\mathbf{C}$、$n\times n$矩阵$\mathbf{V}$、$p\times 1$测量向量$\boldsymbol{y}$，求解：

$$\min \|\boldsymbol{s}\|_0$$
$$\text{s.t. } \mathbf{C}\mathbf{V}\boldsymbol{s}=\boldsymbol{y}.$$

$\|\boldsymbol{s}\|_0$为$n\times 1$稀疏向量$\boldsymbol{s}$的非零元素个数。




（1）恢复信号$\tilde{x}$能否有效地还原原始信号$x$？ ✓
（2）如何解决信号恢复的最优化问题？

# 优化问题

问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 是非凸优化问题，是NP难问题

什么是凸优化问题（convex optimization）？

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

s.t. $\boldsymbol{x} \in \mathcal{C}$. ← 可行域是凸集

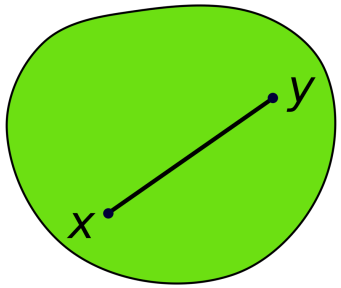
* 最大化问题可以改写成最小化问题
* $\boldsymbol{x} \in \mathcal{C}$可以是关于$\boldsymbol{x}$的等式/不等式，或限定$\boldsymbol{x}$取整数等

# 优化问题

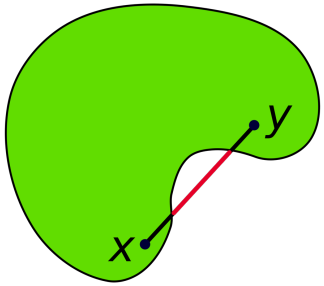
$$\min f(\boldsymbol{x}) \quad \leftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \leftarrow \text{可行域是凸集}$$

凸集（convex set）：

集合中任取两点，两点间线段上的点也属于集合




凸集




非凸集

# 优化问题

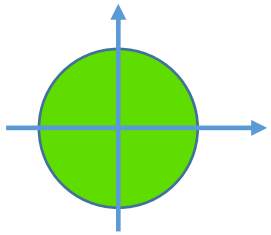
$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

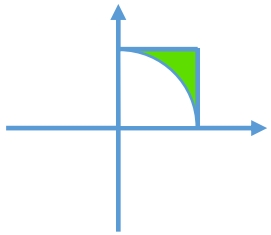
凸集（convex set）：

集合中任取两点，两点间线段上的点也属于集合

例：可行域$\{(x_1, x_2) | x_1^2 + x_2^2 \leq 1\}$是凸集



例：可行域$\{(x_1, x_2) | x_1^2 + x_2^2 \geq 1, 0 \leq x_1, x_2 \leq 1\}$是非凸集



定义：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$，则集合$\mathcal{C}$为凸集。

# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸集（convex set）：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

集合$\{\boldsymbol{s} \in \mathbb{R}^n | \mathbf{C}\mathbf{V}\boldsymbol{s} = \boldsymbol{y}\}$是否是凸集?

# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸集（convex set）：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

集合$\{\boldsymbol{s} \in \mathbb{R}^n | \mathbf{C}\mathbf{V}\boldsymbol{s} = \boldsymbol{y}\}$是否是凸集?

若$\boldsymbol{s}_1, \boldsymbol{s}_2$分别满足$\mathbf{C}\mathbf{V}\boldsymbol{s}_1 = \boldsymbol{y}$与$\mathbf{C}\mathbf{V}\boldsymbol{s}_2 = \boldsymbol{y}$，易证$\lambda\boldsymbol{s}_1 + (1-\lambda)\boldsymbol{s}_2$满足$\mathbf{C}\mathbf{V}(\lambda\boldsymbol{s}_1 + (1-\lambda)\boldsymbol{s}_2) = \boldsymbol{y}$

# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸集（convex set）：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

集合$\{\boldsymbol{s} \in \mathbb{R}^n | \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}\}$是否是凸集?

若$\boldsymbol{s}_1, \boldsymbol{s}_2$分别满足$\mathbf{CV}\boldsymbol{s}_1 = \boldsymbol{y}$与$\mathbf{CV}\boldsymbol{s}_2 = \boldsymbol{y}$，易证$\lambda\boldsymbol{s}_1 + (1-\lambda)\boldsymbol{s}_2$满足$\mathbf{CV}(\lambda\boldsymbol{s}_1 + (1-\lambda)\boldsymbol{s}_2) = \boldsymbol{y}$

例：若$\boldsymbol{s} = (s_1, s_2) \in \mathbb{R}^2$，且等式约束为$s_1 + s_2 = 1$。易见$\boldsymbol{s}$的可行域是一条直线

例：若$\boldsymbol{s} = (s_1, s_2, s_3) \in \mathbb{R}^3$，且等式约束为$s_1 + s_2 + s_3 = 1$。易见$\boldsymbol{s}$的可行域是一个平面

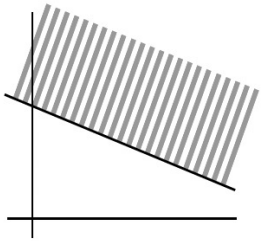
# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

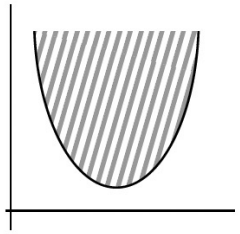
$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸函数：

函数及其以上的区域是凸集

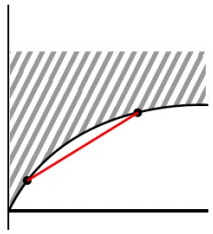


$f(x) = ax + b$



$f(x) = \text{quadratic}$

凸函数



$f(x) = \sqrt{x}$

非凸函数

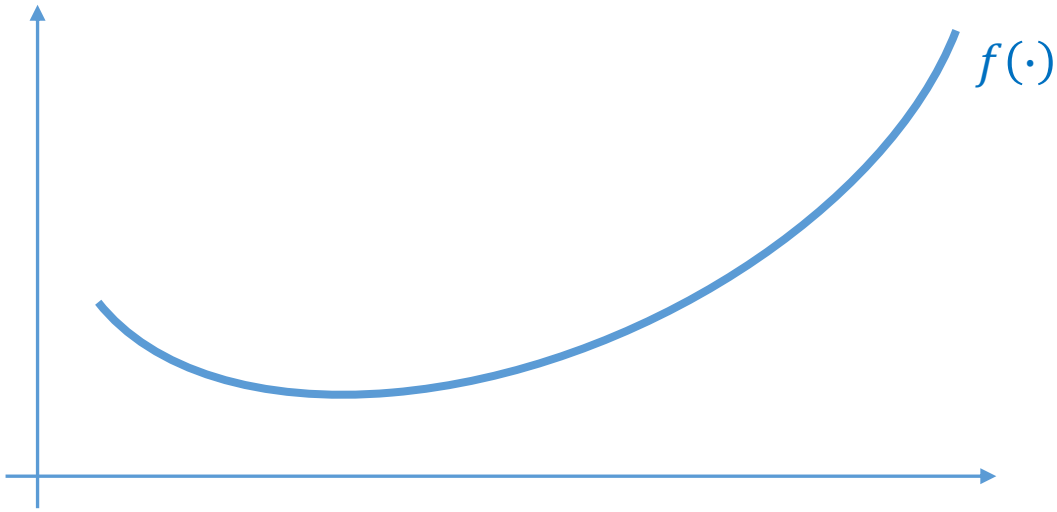
图片取自Tim Roughgarden and Gregory Valiant <CS 168 - The Modern Algorithmic Toolbox>

# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

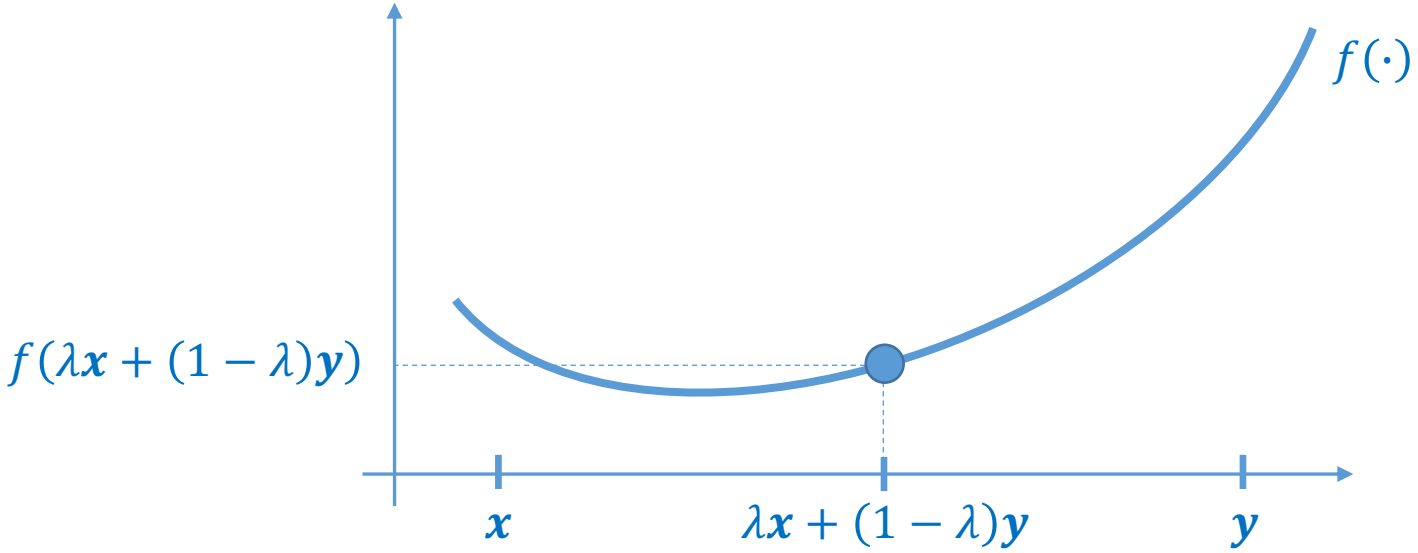


# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

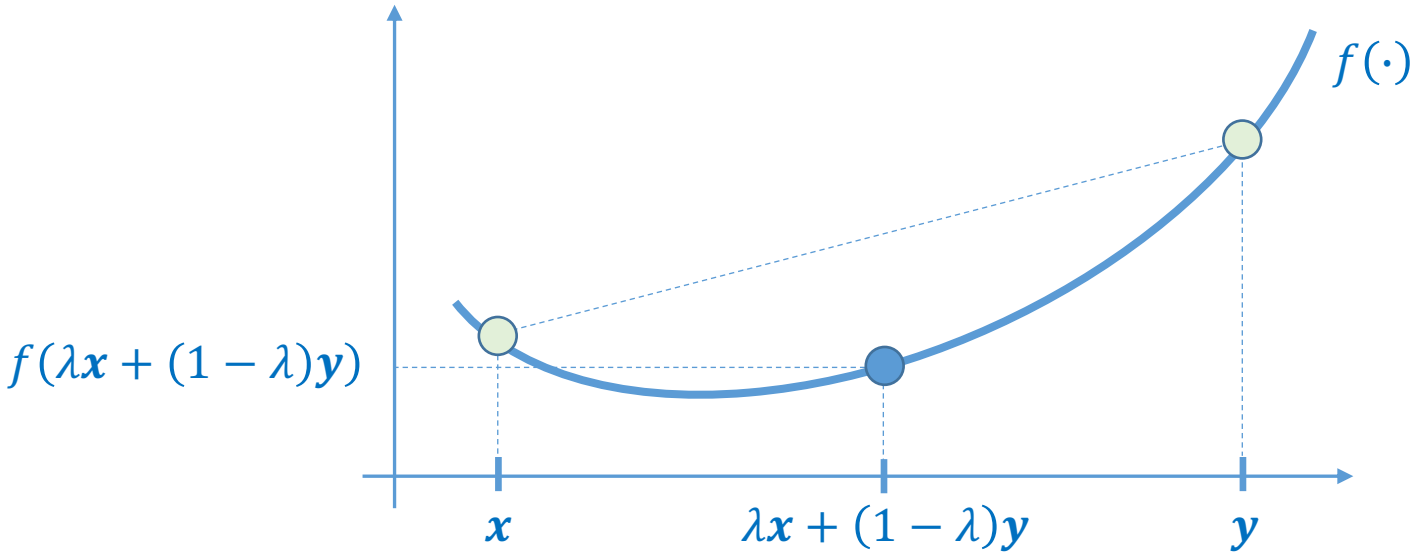


# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

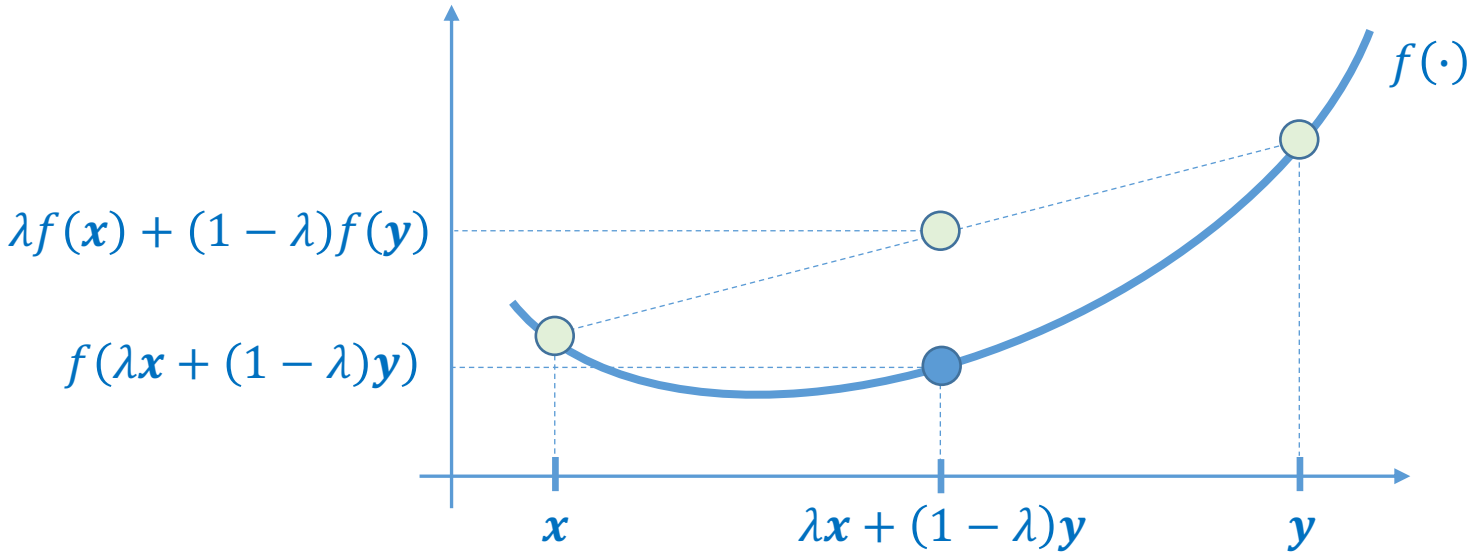


# 优化问题

$\min f(\boldsymbol{x})$ ← 目标函数是凸函数

$\text{s.t. } \boldsymbol{x} \in \mathcal{C}.$ ← 可行域是凸集

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。



# 优化问题

$$\min f(\boldsymbol{x}) \quad \leftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \leftarrow \text{可行域是凸集}$$

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda)f(\boldsymbol{y})$。

# 优化问题

$$\min f(\boldsymbol{x}) \quad \leftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \leftarrow \text{可行域是凸集}$$

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

即$\mathbf{A}\boldsymbol{x} - \boldsymbol{b}$形式

# 优化问题

$$\min f(\boldsymbol{x}) \quad \leftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \leftarrow \text{可行域是凸集}$$

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

即$\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$形式

如何判断函数是不是凸函数：若函数二次可微，检查其海森矩阵是否是半正定矩阵（细节略）

# 优化问题

$$\min f(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{x} \in \mathcal{C}.$$

目标函数是凸函数

可行域是凸集

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \ldots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

即$\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$形式

如何判断函数是不是凸函数：若函数二次可微，检查其海森矩阵是否是半正定矩阵（细节略）

例，对$f(x_1, x_2)$，检查 $\begin{bmatrix} \frac{\partial^2 f(x_1,x_2)}{\partial x_1^2} & \frac{\partial^2 f(x_1,x_2)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(x_1,x_2)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x_1,x_2)}{\partial x_2^2} \end{bmatrix}$

# 优化问题

$$\min f(\boldsymbol{x}) \quad \longleftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \longleftarrow \text{可行域是凸集}$$

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

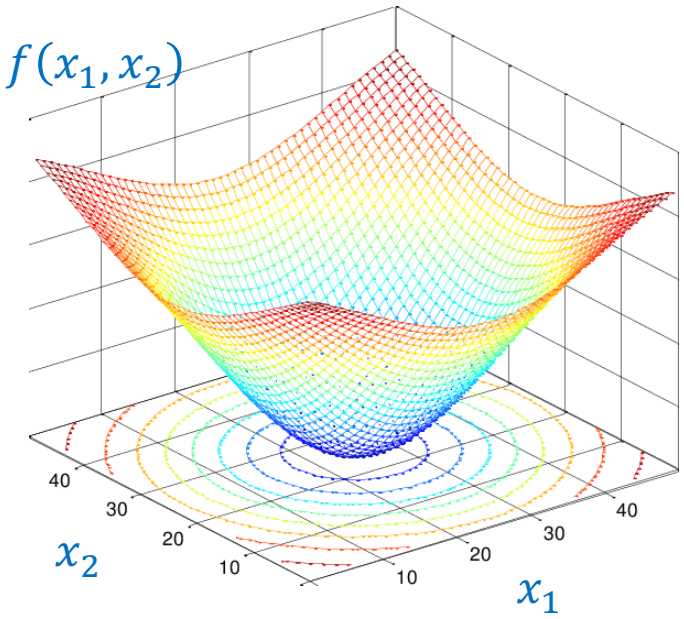
凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \ldots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

凸优化问题的优异性质：局部最优解一定是全局最优解



# 优化问题

$$\min f(\boldsymbol{x}) \quad \longleftarrow \text{目标函数是凸函数}$$
$$\text{s.t. } \boldsymbol{x} \in \mathcal{C}. \quad \longleftarrow \text{可行域是凸集}$$

凸集：对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，有$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C}$。

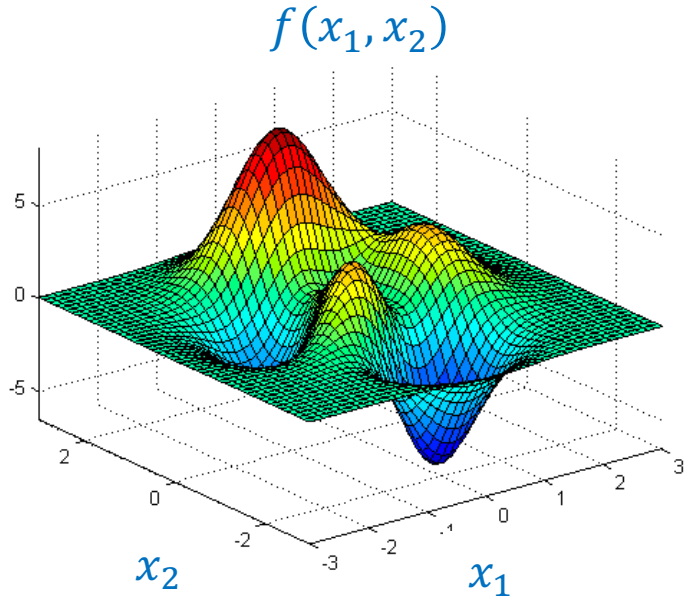
凸函数：若对任意$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$及$\lambda \in [0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$。

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \ldots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

非凸优化问题：局部最优解不一定是全局最优解



# 优化问题

问题 $\begin{aligned} &\min \|\boldsymbol{s}\|_0 \\ &\text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}. \end{aligned}$ 是非凸优化问题，是NP难问题

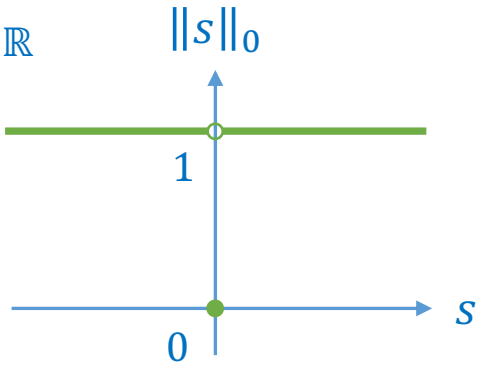
- 可行域是凸集
- 目标函数是非凸函数

# 优化问题

问题 $\begin{aligned}&\min \|\boldsymbol{s}\|_0\\ &\text{s.t. } \boldsymbol{CV}\boldsymbol{s}=\boldsymbol{y}.\end{aligned}$ 是非凸优化问题，是NP难问题

- 可行域是凸集
- 目标函数是非凸函数

例：若$\boldsymbol{s}\in\mathbb{R}$

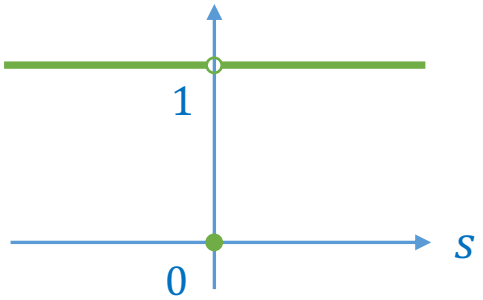


# 优化问题

问题 $\begin{aligned}&\min \|\boldsymbol{s}\|_0\\ &\text{s.t. } \boldsymbol{CVs}=\boldsymbol{y}.\end{aligned}$ 是非凸优化问题，是NP难问题

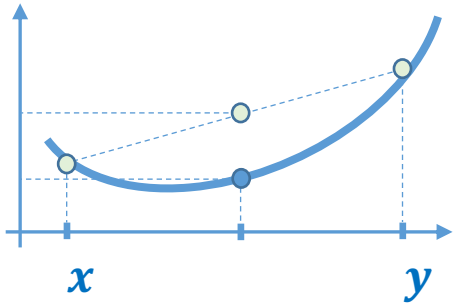
- 可行域是凸集
- 目标函数是非凸函数

例：若$\boldsymbol{s}\in\mathbb{R}$




例：若$\boldsymbol{s}\in\mathbb{R}^2$，分析目标函数在$(1,0),(0,1)$两点的取值以及在$(\lambda,1-\lambda)$的取值，也可见非凸性

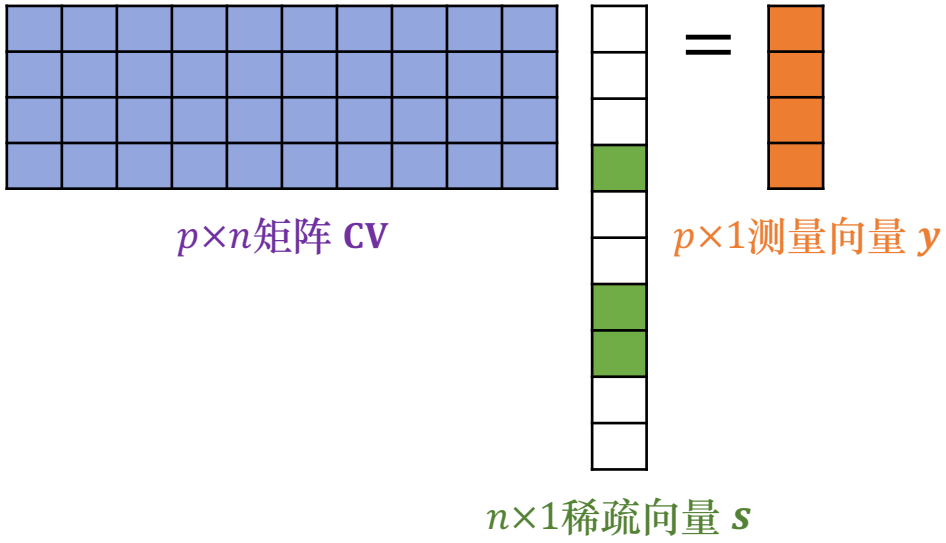
凸函数：若对任意$\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}$及$\lambda\in[0,1]$，函数$f(\cdot)$满足$f(\lambda\boldsymbol{x}+(1-\lambda)\boldsymbol{y})\leq\lambda f(\boldsymbol{x})+(1-\lambda)f(\boldsymbol{y})$。



# 优化问题

问题 $\begin{aligned} &\min \|\boldsymbol{s}\|_0 \\ &\text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}. \end{aligned}$ 是非凸优化问题，是NP难问题

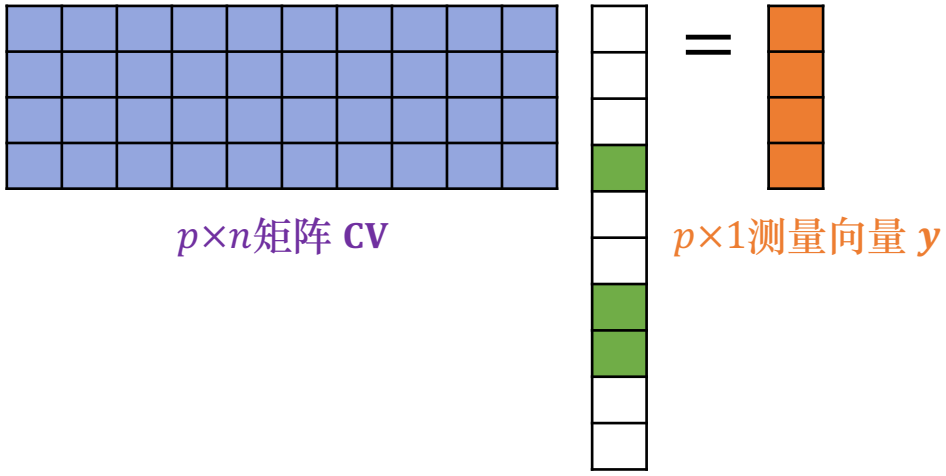
属于组合优化的范畴



# 优化问题

问题 $\begin{aligned}&\min \|\boldsymbol{s}\|_0\\ &\text{s.t. } \boldsymbol{CVs}=\boldsymbol{y}.\end{aligned}$ 是非凸优化问题，是NP难问题

属于组合优化的范畴




$p\times n$矩阵 $\boldsymbol{CV}$

$p\times 1$测量向量 $\boldsymbol{y}$

$n\times 1$稀疏向量 $\boldsymbol{s}$

如果知道向量$\boldsymbol{s}$非零元素的个数及位置，容易直接求解$\boldsymbol{CVs}=\boldsymbol{y}$

但实际不知道$\boldsymbol{s}$非零元素的个数及位置，如果穷举各种可能性：

$$\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{n}\sim 2^n$$

指数时间复杂度

分析取自清华大学张颢 <现代数字信号处理2>

# 优化问题

问题 $\begin{aligned}&\min \|\boldsymbol{s}\|_0\\ &\text{s.t. } \boldsymbol{CVs}=\boldsymbol{y}.\end{aligned}$ 是非凸优化问题，是NP难问题

解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$　候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$　　L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

# 优化问题

问题 $\begin{aligned}&\min \|\boldsymbol{s}\|_0\\ &\text{s.t. } \boldsymbol{CVs}=\boldsymbol{y}.\end{aligned}$ 是非凸优化问题，是NP难问题

**解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$** 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

易证$\|\boldsymbol{s}\|_1$和$\|\boldsymbol{s}\|_2$都是凸函数

$\|\cdot\|_1$满足$\|\lambda\boldsymbol{x}+(1-\lambda)\boldsymbol{y}\|_1 \le \lambda\|\boldsymbol{x}\|_1+(1-\lambda)\|\boldsymbol{y}\|_1$

$\|\cdot\|_2$满足$\|\lambda\boldsymbol{x}+(1-\lambda)\boldsymbol{y}\|_2 \le \lambda\|\boldsymbol{x}\|_2+(1-\lambda)\|\boldsymbol{y}\|_2$（两边之和大于第三边）

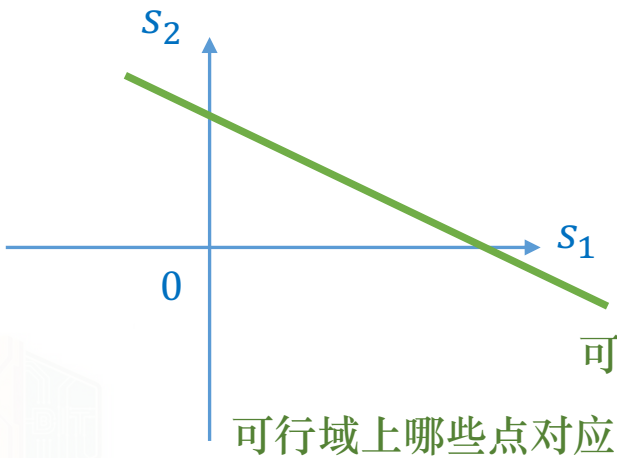
# 优化问题

问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$ 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n} |s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n} s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$




可行域上哪些点对应的$\|\boldsymbol{s}\|_0$最小?

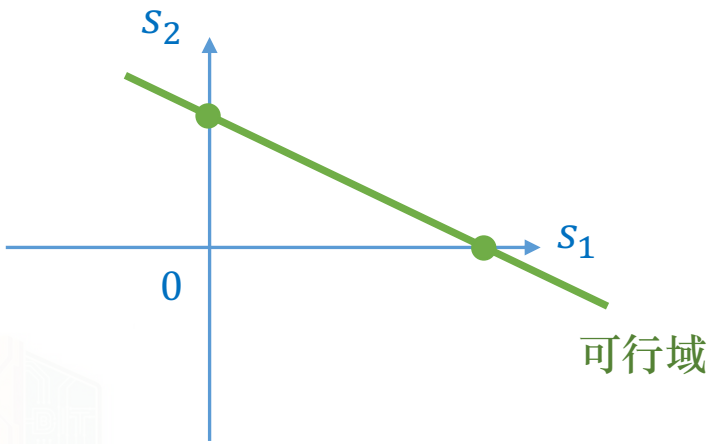
# 优化问题

问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$ 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$



# 优化问题

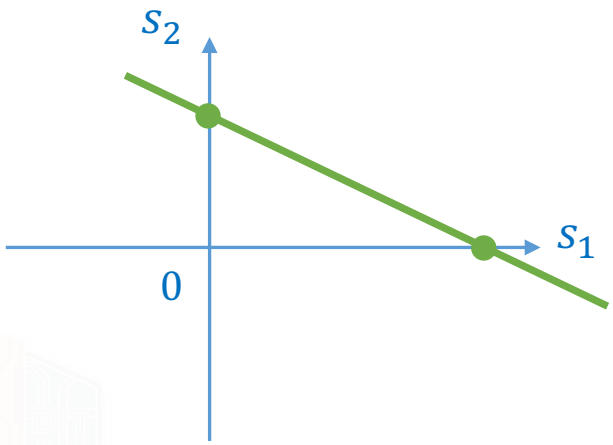
问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \boldsymbol{C}\boldsymbol{V}\boldsymbol{s} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

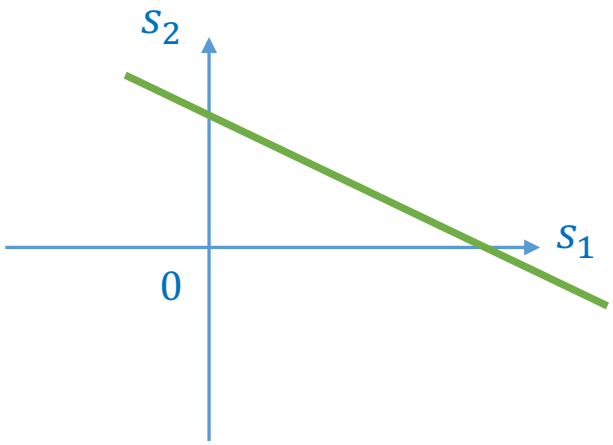
解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$　候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$　　L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ $\text{s.t. } s_1 + 2s_2 = 2.$



# 优化问题

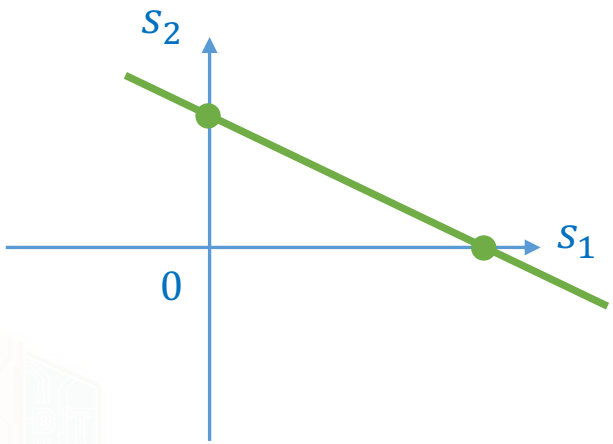
问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

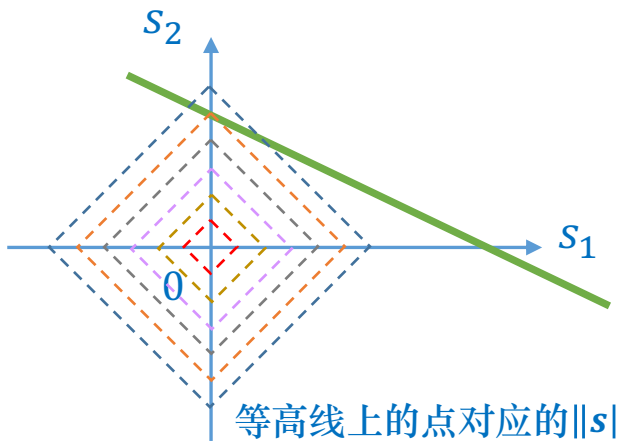
**解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$** 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ $\text{s.t. } s_1 + 2s_2 = 2.$




等高线上的点对应的$\|\boldsymbol{s}\|_1$值相等

# 优化问题

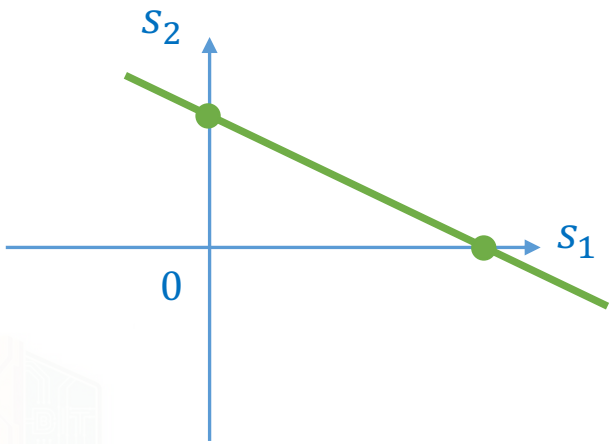
问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

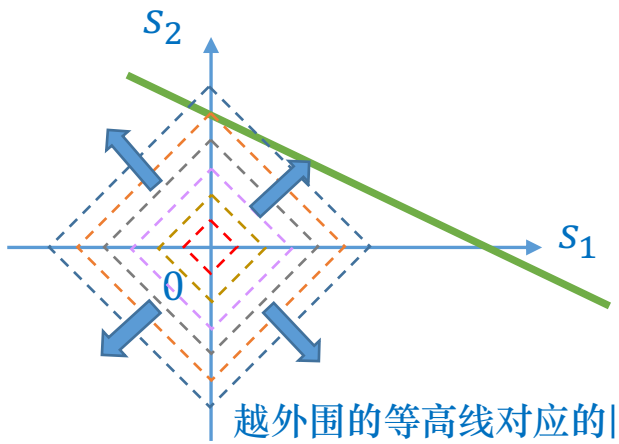
**解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$** 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ $\text{s.t. } s_1 + 2s_2 = 2.$




越外围的等高线对应的$\|\boldsymbol{s}\|_1$值越大

# 优化问题

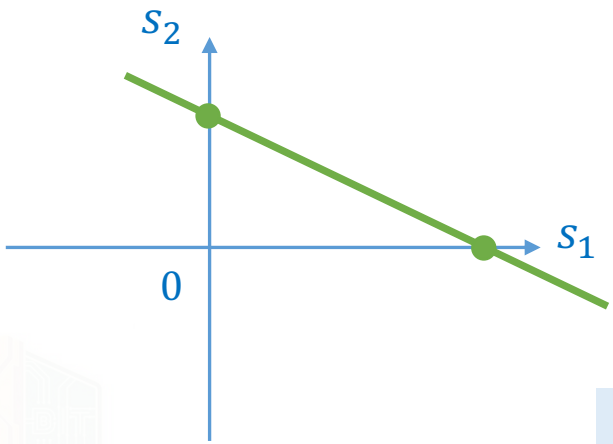
问题 $\min \|\boldsymbol{s}\|_0$ $\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$ 是非凸优化问题，是NP难问题

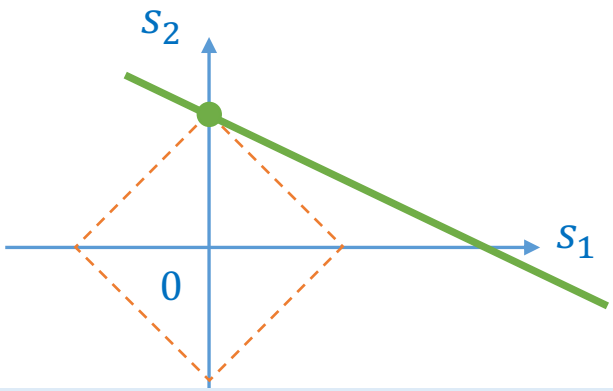
解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$　候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$　　L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ $\text{s.t. } s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ $\text{s.t. } s_1 + 2s_2 = 2.$




最优解的位置与L0-范数问题的一个最优解重合

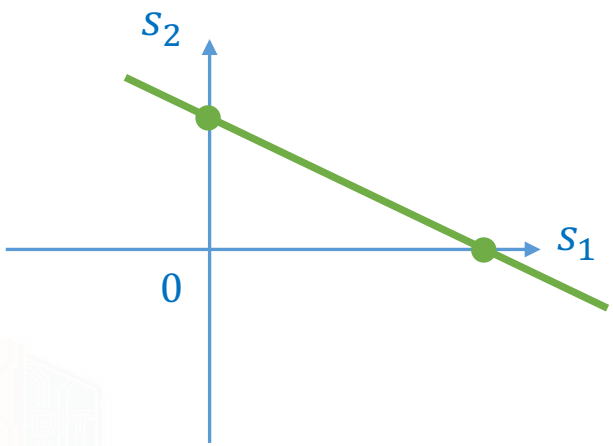
# 优化问题

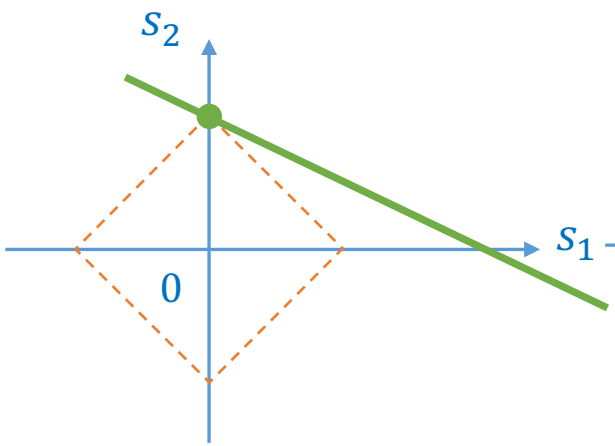
问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 是非凸优化问题，是NP难问题

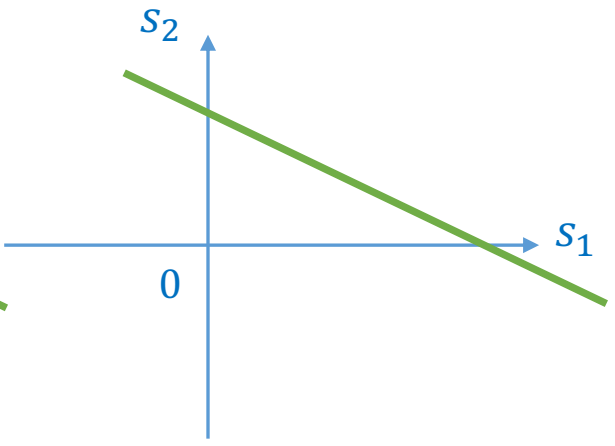
解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$　候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

L1-范数 $\sum_{i=1}^{n}|s_i|$　L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：$\min \|\boldsymbol{s}\|_0$ s.t. $s_1 + 2s_2 = 2.$　$\min \|\boldsymbol{s}\|_1$ s.t. $s_1 + 2s_2 = 2.$　$\min \|\boldsymbol{s}\|_2$ s.t. $s_1 + 2s_2 = 2.$



# 优化问题

问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 是非凸优化问题，是NP难问题

**解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$** 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

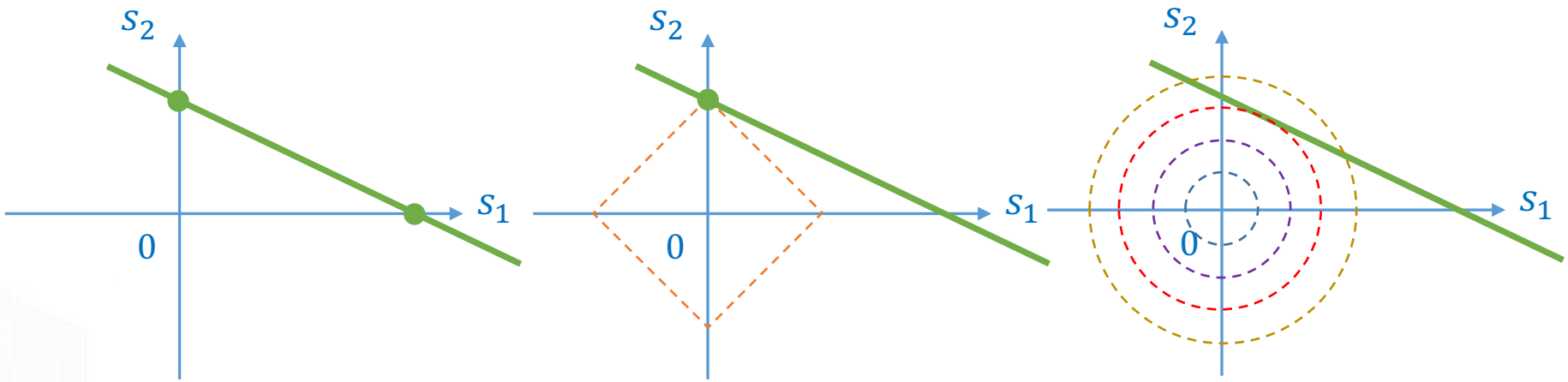
L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

例：

$\min \|\boldsymbol{s}\|_0$ s.t. $s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ s.t. $s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_2$ s.t. $s_1 + 2s_2 = 2.$



# 优化问题

问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 是非凸优化问题，是NP难问题

**解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$** 候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$

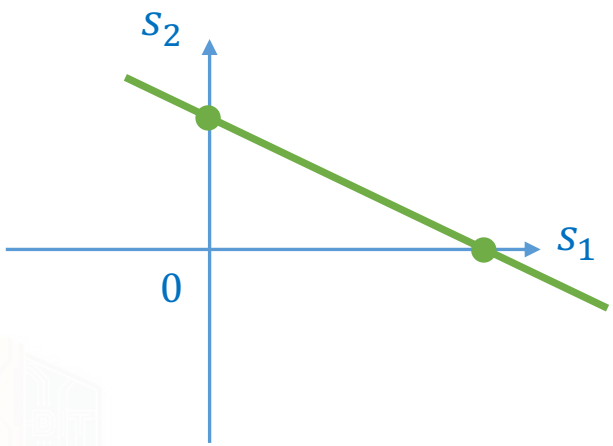
L1-范数 $\sum_{i=1}^{n}|s_i|$ L2-范数 $\sqrt{\sum_{i=1}^{n}s_i^2}$

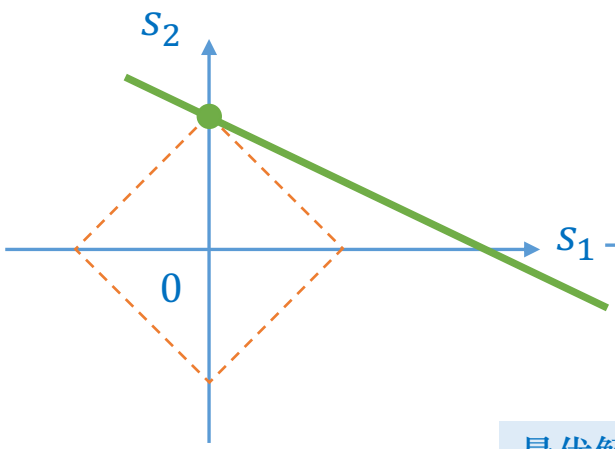
例：

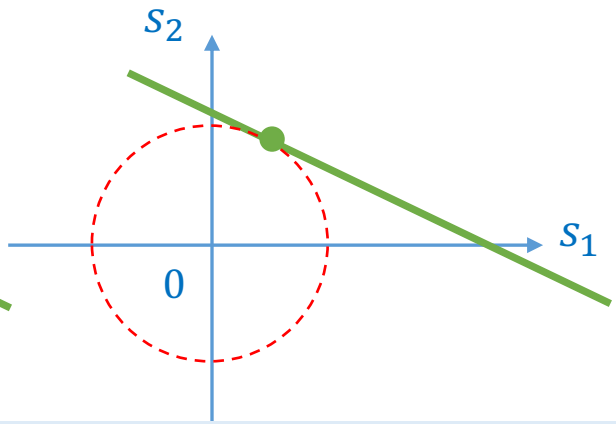
$\min \|\boldsymbol{s}\|_0$ s.t. $s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_1$ s.t. $s_1 + 2s_2 = 2.$

$\min \|\boldsymbol{s}\|_2$ s.t. $s_1 + 2s_2 = 2.$



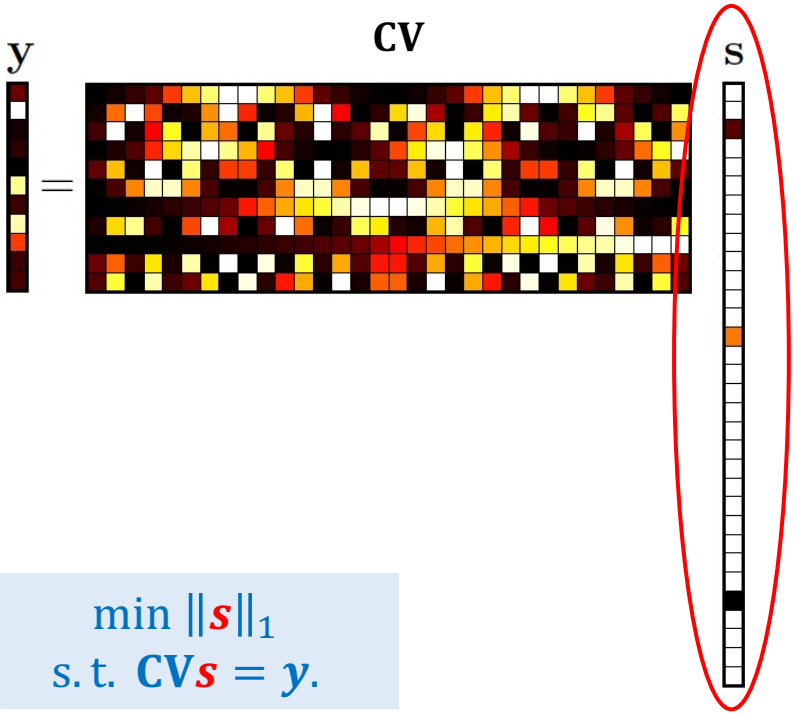




最优解的位置与L0-范数问题的最优解不同

# 优化问题

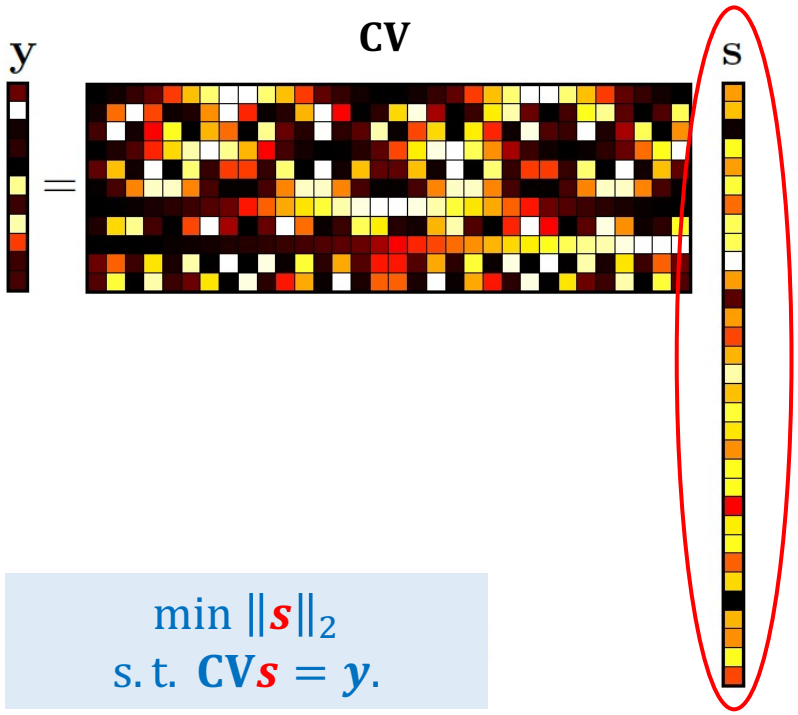
问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 是非凸优化问题，是NP难问题

解决方案：用新目标函数近似原目标函数$\|\boldsymbol{s}\|_0$　候选：$\|\boldsymbol{s}\|_1$以及$\|\boldsymbol{s}\|_2$




$\min \|\boldsymbol{s}\|_1$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$.

更能保证得到稀疏的$\boldsymbol{s}$




$\min \|\boldsymbol{s}\|_2$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$.

图片取自Steven L. Brunton, J. Nathan Kutz
<Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control>

# 优化问题

**Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information**

EJ Candès, J Romberg, T Tao - IEEE Transactions on …, 2006 - ieeexplore.ieee.org

This paper considers the model problem of reconstructing an object from incomplete frequency samples. Consider a discrete-time signal f/spl isin/C/sup N/and a randomly chosen set of frequencies/spl Omega/. Is it possible to reconstruct f from the partial knowledge of its Fourier coefficients on the set/spl Omega/? A typical result of this paper is as follows. Suppose that f is a superposition of| T| spikes f (t)=/spl sigma//sub/spl tau//spl isin/T/f (/spl tau/)/spl delta/(t-/spl tau/) obeying| T|/spl les/C/sub M//spl middot/(log N)/sup …

☆ Save 99 Cite Cited by 17967 Related articles All 32 versions

A typical result of this paper is as follows: for each $M > 0$, suppose that $f$ obeys

$$\#\{t,\ f(t) \neq 0\} \leq \alpha(M) \cdot (\log N)^{-1} \cdot \#\Omega,$$

then with probability at least $1 - O(N^{-M})$, $f$ can be reconstructed exactly as the solution to the $\ell_1$ minimization problem

$$\min_g \sum_{t=0}^{N-1} |g(t)|, \quad \text{s.t.} \ \ \hat{g}(\omega) = \hat{f}(\omega) \ \text{for all} \ \ \omega \in \Omega.$$

In short, exact recovery may be obtained by solving a convex optimization problem.

可以严格证明，在一些条件下，L1最小化问题的最优解是原信号（图像）的***s***

# 压缩感知

## 基于线性规划的求解法

# 线性规划

求解近似问题

$$\min \|\boldsymbol{s}\|_1$$
$$\text{s.t. } \mathbf{C}\mathbf{V}\boldsymbol{s} = \boldsymbol{y}.$$

# 线性规划

求解近似问题

$$\min \|\boldsymbol{s}\|_1$$
$$\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$$

凸优化问题中最容易求解的问题之一是线性规划（linear programming）

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \le 0, i = 1, \dots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

**线性规划问题**

$$\begin{aligned} \min \quad & \boldsymbol{a}_0\boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i\boldsymbol{x} + b_i \le 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j\boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

当$f(\cdot), g_i(\cdot)$也都是仿射函数时

# 线性规划

求解近似问题

$$\min \|\boldsymbol{s}\|_1$$
$$\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$$

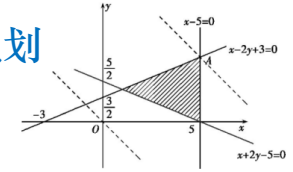
高中学过图解法求解二元线性规划



考向二　线性目标函数的最值问题

例2　(2018课标Ⅱ,14,5分)若$x,y$满足约束条件$\begin{cases} x+2y-5\geq 0, \\ x-2y+3\geq 0, \\ x-5\leq 0, \end{cases}$则$z=x+y$的最大值为_____.

解析　由线性约束条件画出可行域(如图中阴影部分所示).




当直线$x+y-z=0$经过点$A(5,4)$时,$z=x+y$取得最大值,最大值为9.

答案　9

凸优化问题中最容易求解的问题之一是线性规划（linear programming）

**凸优化问题的标准形式**

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_i(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

其中，$f(\cdot), g_i(\cdot)$为凸函数、$h_i(\boldsymbol{x})$为仿射函数。

**线性规划问题**

$$\begin{aligned} \min \quad & \boldsymbol{a}_0\boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i\boldsymbol{x} + b_i \leq 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j\boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

当$f(\cdot), g_i(\cdot)$也都是仿射函数时

得益于线性规划在工业界的广泛应用，已有多种多项式时间算法可以高效求得大规模（如包含上千万个决策变量）线性规划问题的全局最优解

在所有类型的优化问题中，线性规划或许是最兼具易解性与适用性的问题

# 线性规划

求解近似问题

$$\min \sum_{i=1}^{n} |s_i|$$
$$\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$$

**线性规划问题**

$$\begin{aligned} \min \quad & \boldsymbol{a}_0\boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i\boldsymbol{x} + b_i \le 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j\boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

是否是线性规划？可否改写成线性规划？

# 线性规划

求解近似问题

$$\min \sum_{i=1}^{n} |s_i|$$
$$\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$$

**线性规划问题**

$$\begin{aligned} \min \quad & \boldsymbol{a}_0\boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i\boldsymbol{x} + b_i \le 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j\boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

是否是线性规划？可否改写成线性规划？

引入新变量$\boldsymbol{t}$，将问题改写为线性规划问题：

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} t_i \\ \text{s.t.} \quad & \boldsymbol{CVs} = \boldsymbol{y}, \\ & t_i \ge s_i, \forall i, \\ & t_i \ge -s_i, \forall i. \end{aligned}$$

$\boldsymbol{t}, \boldsymbol{s} \in \mathbb{R}^n$都是决策变量

# 线性规划

求解近似问题

$$\min \sum_{i=1}^{n} |s_i|$$
$$\text{s.t. } \boldsymbol{CVs} = \boldsymbol{y}.$$

**线性规划问题**

$$\begin{aligned} \min \quad & \boldsymbol{a}_0\boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i\boldsymbol{x} + b_i \le 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j\boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

是否是线性规划？可否改写成线性规划？

引入新变量$\boldsymbol{t}$，将问题改写为线性规划问题：

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} t_i \\ \text{s.t.} \quad & \boldsymbol{CVs} = \boldsymbol{y}, \\ & t_i \ge s_i, \forall i, \\ & t_i \ge -s_i, \forall i. \end{aligned}$$

$\boldsymbol{t}, \boldsymbol{s} \in \mathbb{R}^n$都是决策变量

* $t_i \ge s_i$与$t_i \ge -s_i$对应$t_i \ge \max\{s_i, -s_i\} = |s_i|$

* 对最优解$t_i^*, s_i^*$，一定有$t_i^* = |s_i^*|$（可反证得）

# 线性规划

可将问题 $\min \|\boldsymbol{s}\|_1$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$. 改写成线性规划再求解

实际中可考虑的其它问题建模方式：

$\min \|\boldsymbol{s}\|_1$ s.t. $\|\boldsymbol{CVs} - \boldsymbol{y}\|_2 \leq \varepsilon$.

$\min \lambda\|\boldsymbol{s}\|_1 + \|\boldsymbol{CVs} - \boldsymbol{y}\|_2$

# 实验一：对比L1与L2范数

给定任意矩阵CV及向量$\boldsymbol{y}$，对比以下两个优化问题的最优解的稀疏度

$$\min \|\boldsymbol{s}\|_1 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

$$\min \|\boldsymbol{s}\|_2 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

```
n = 10^4;
p = 10^3;
matrixCV = randn(p,n);
y = randn(p,1);
```



代码修改自Steven L. Brunton, J. Nathan Kutz
<Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control>

# 实验一：对比L1与L2范数

给定任意矩阵CV及向量$y$，对比以下两个优化问题的最优解的稀疏度

$$\min \|s\|_1 \quad \text{s.t. } CVs = y.$$

$$\min \|s\|_2 \quad \text{s.t. } CVs = y.$$

```
n = 10^4;
p = 10^3;
matrixCV = randn(p,n);
y = randn(p,1);

cvx_begin;
  variable s(n);
  minimize(norm(s,1));
  subject to
    matrixCV*s==y;
cvx_end;
```

Matlab用于求解凸优化问题的包

执行时间 381.13秒

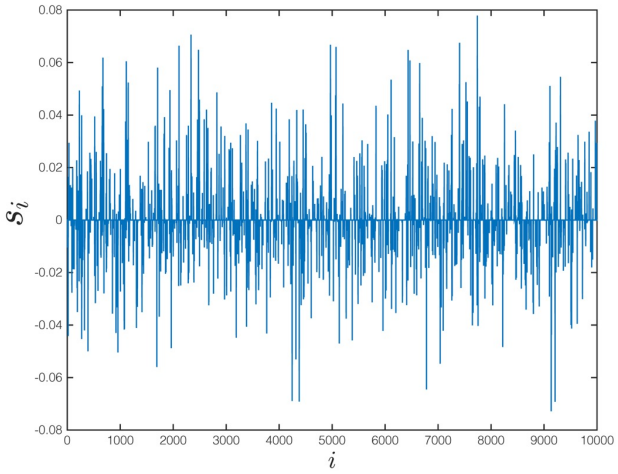
# 实验一：对比L1与L2范数

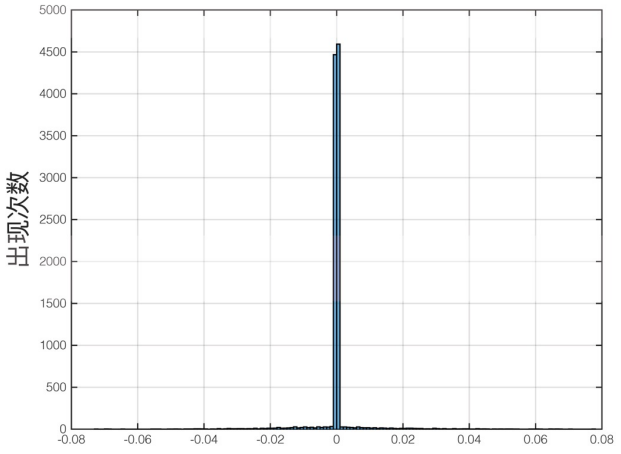
给定任意矩阵CV及向量$y$，对比以下两个优化问题的最优解的稀疏度

$$\min \|s\|_1 \quad \text{s.t. } CVs = y.$$

$$\min \|s\|_2 \quad \text{s.t. } CVs = y.$$

```
n = 10^4;
p = 10^3;
matrixCV = randn(p,n);
y = randn(p,1);

cvx_begin;
  variable s(n);
  minimize(norm(s,1));
  subject to
    matrixCV*s==y;
cvx_end;

figure(1)
plot(s);
figure(2)
histogram(s);
```




直方图

代码修改自Steven L. Brunton, J. Nathan Kutz
<Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control>

# 实验一：对比L1与L2范数

给定任意矩阵**CV**及向量$\boldsymbol{y}$，对比以下两个优化问题的最优解的稀疏度

$$\min \|\boldsymbol{s}\|_1 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

$$\min \|\boldsymbol{s}\|_2 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

等价于约束条件均为等式的二次规划问题
可以推导出全局最优解的解析形式

```
n = 10^4;
p = 10^3;
matrixCV = randn(p,n);
y = randn(p,1);

s = pinv(matrixCV)*y;
```

pinv(matrixCV)计算**CV**的伪逆矩阵$(\mathbf{CV})^+$

求解迅速

代码修改自Steven L. Brunton, J. Nathan Kutz
<Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control>

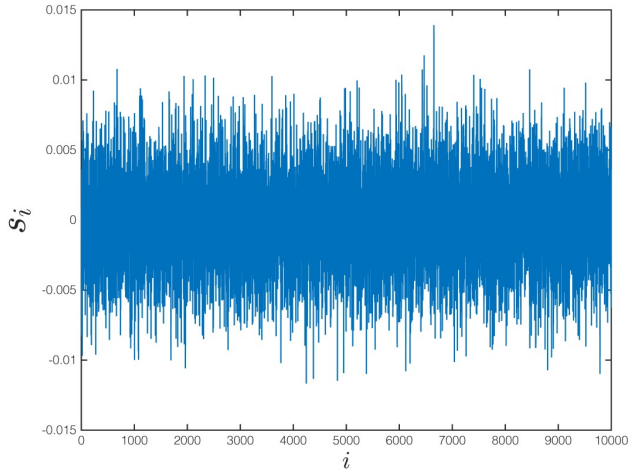
# 实验一：对比L1与L2范数

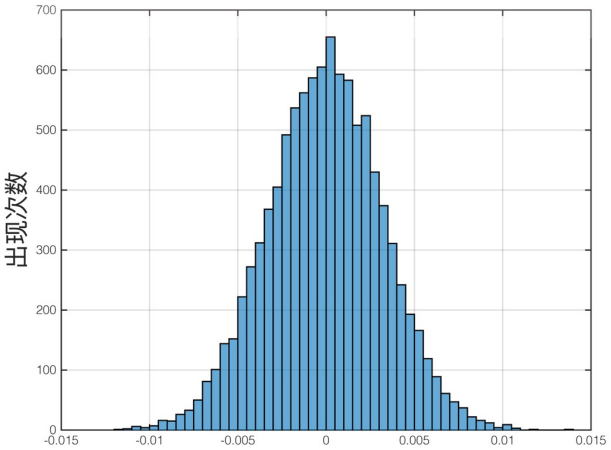
给定任意矩阵CV及向量$y$，对比以下两个优化问题的最优解的稀疏度

$$\min \|s\|_1 \quad \text{s.t. } CVs = y.$$

$$\min \|s\|_2 \quad \text{s.t. } CVs = y.$$

```
n = 10^4;
p = 10^3;
matrixCV = randn(p,n);
y = randn(p,1);

s = pinv(matrixCV)*y;

figure(1)
plot(s);
figure(2)
histogram(s);
```



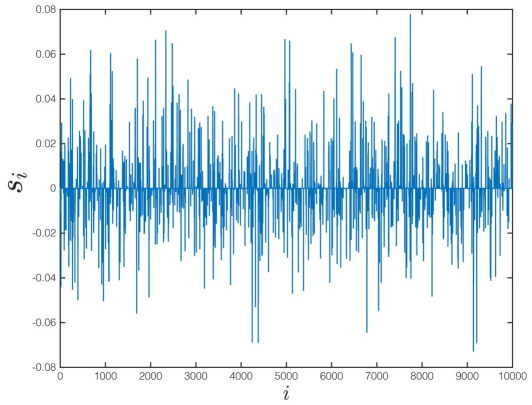

代码修改自Steven L. Brunton, J. Nathan Kutz
<Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control>

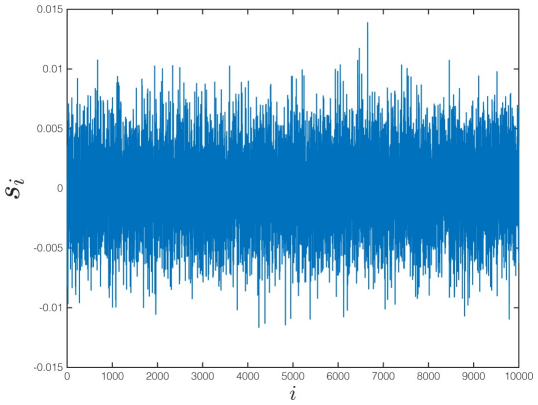
# 实验一：对比L1与L2范数

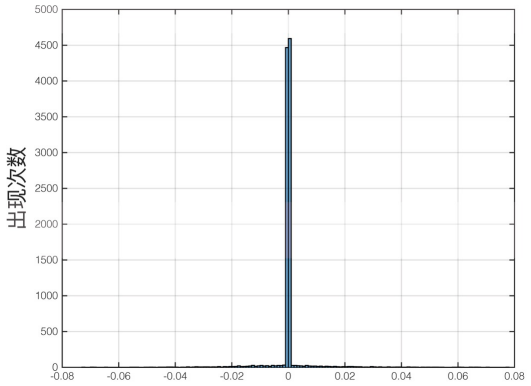
给定任意矩阵CV及向量$\boldsymbol{y}$，对比以下两个优化问题的最优解的稀疏度

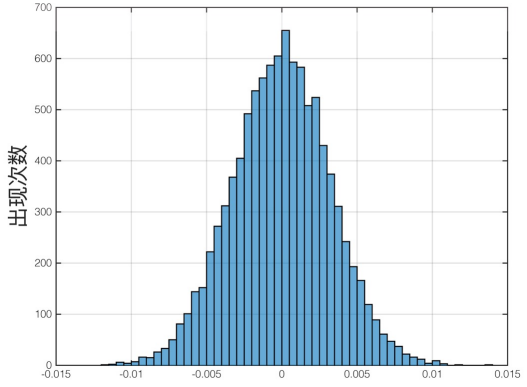
$$\min \|\boldsymbol{s}\|_1 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

$$\min \|\boldsymbol{s}\|_2 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

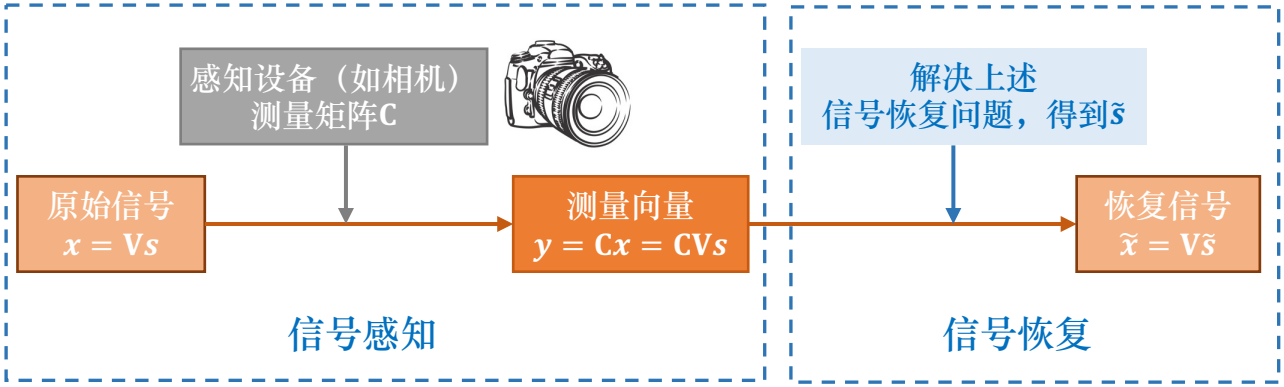


# 实验二：压缩感知

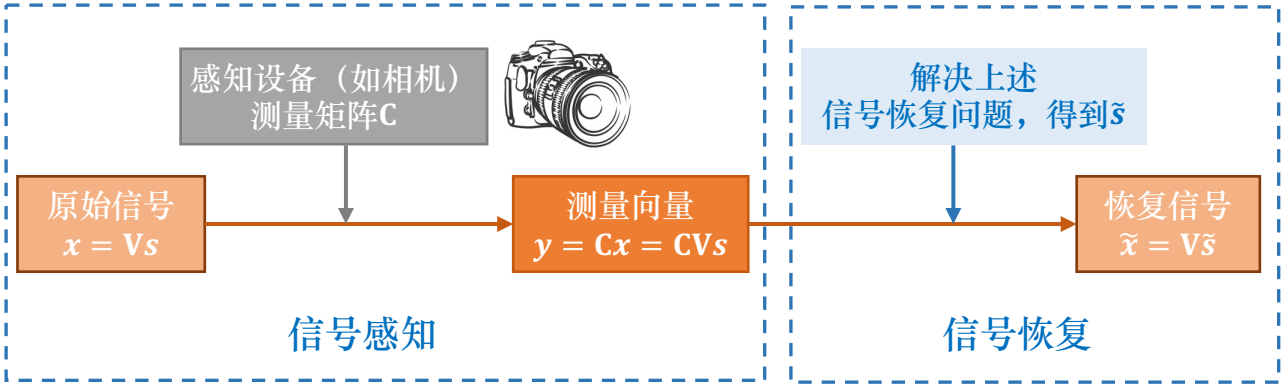



模拟信号感知的过程：

```
A=imread('animal','png');
Abw=rgb2gray(A);
imwrite(uint8(Abw),'csoriginal.jpg');
```

限于算力，选择100×100像素图片

# 实验二：压缩感知



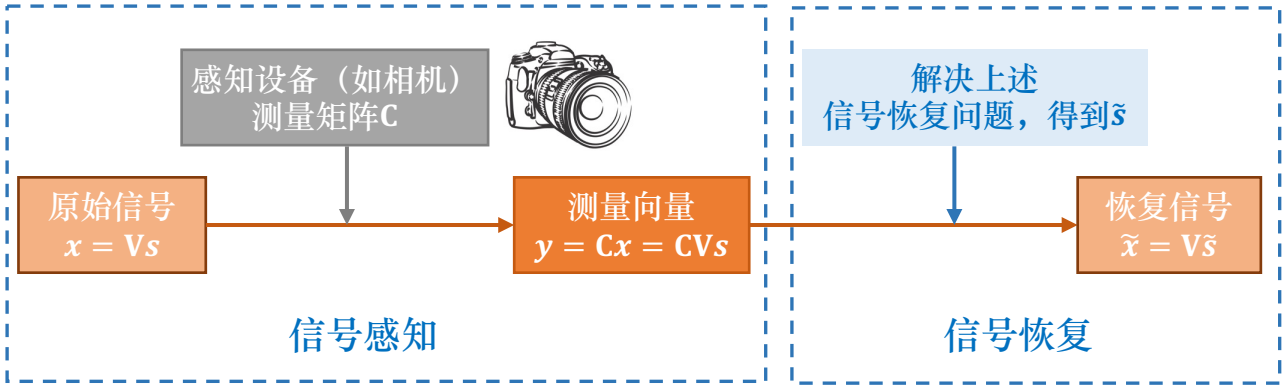

模拟信号感知的过程：

```
A=imread('animal','png');
Abw=rgb2gray(A);
imwrite(uint8(Abw),'csoriginal.jpg');

[rsize,csize]=size(Abw);
n=rsize*csize;
p=round(0.2*n);
```

读取维数

压缩至20%

# 实验二：压缩感知

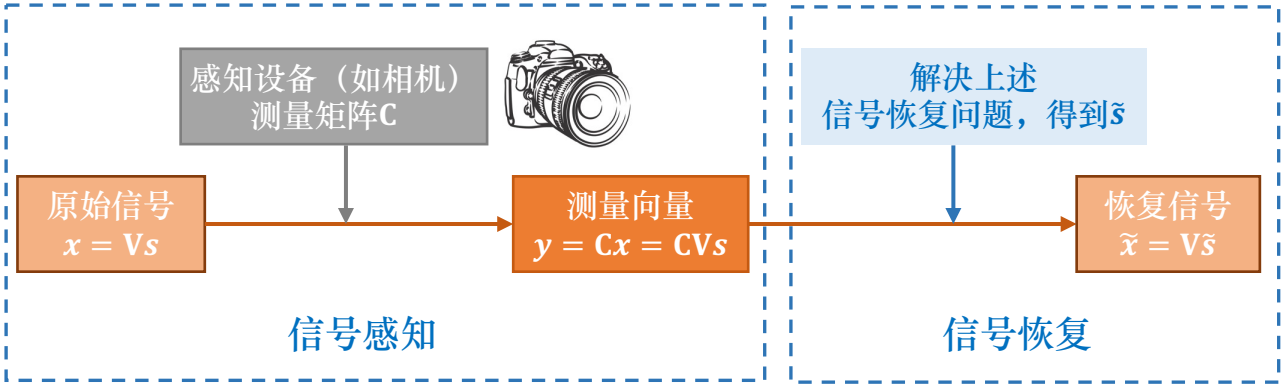



模拟信号感知的过程：

```
A=imread('animal','png');
Abw=rgb2gray(A);
imwrite(uint8(Abw),'csoriginal.jpg');

[rsize,csize]=size(Abw);
n=rsize*csize;
p=round(0.2*n);

x=reshape(Abw,[],1);
C=normrnd(0,1,p,n);
y=C*double(x);
```

通过感知得到2000×1向量$y$

# 实验二：压缩感知




模拟信号恢复的过程：（接上）

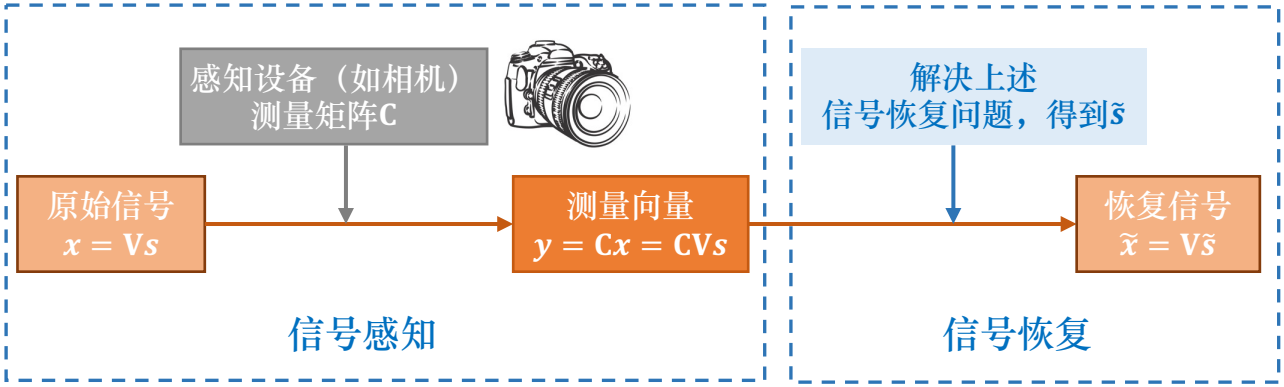
```
V=dct(eye(n,n));
matrixCV=C*V;

cvx_begin;
  variable s(n);
  minimize(norm(s,1));
  subject to
    matrixCV*s==y;
cvx_end;
```

/ ▸ Users ▸ haoranyu ▸ Documents ▸ MATLAB ▸

Command Window

```
Calling SDPT3 4.0: 20000 variables, 2000 equality constraints
------------------------------------------------------------

 num. of constraints = 2000
 dim. of socp   var  = 20000,   num. of socp blk  = 10000
*****************************************************************
   SDPT3: Infeasible path-following algorithms
*****************************************************************
 version  predcorr  gam  expon  scale_data
    NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|1.0e+00|1.0e+02|7.7e+08| 7.431660e+06  0.000000e+00| 0:0:30| chol  1  1
 1|1.000|0.841|1.3e-09|1.7e+01|1.8e+08| 1.022502e+07  1.906350e+05| 0:1:00| chol  1  1
 2|1.000|0.965|6.6e-11|8.2e-01|1.6e+07| 8.853295e+06  5.046843e+04| 0:1:58| chol  1  1
 3|1.000|0.822|7.1e-11|2.5e-01|2.8e+06| 2.289119e+06  5.478894e+04| 0:2:57| chol  1  1
 4|0.914|0.529|4.4e-10|1.5e-01|7.1e+05| 6.835456e+05  8.139438e+04| 0:3:55| chol  1  1
 5|0.843|0.518|6.3e-11|8.8e-02|2.6e+05| 3.399570e+05  1.108943e+05| 0:4:53| chol  1  1
 6|0.823|0.566|1.1e-10|4.7e-02|1.0e+05| 2.214542e+05  1.322548e+05| 0:5:52| chol  1  1
 7|0.726|0.608|1.4e-10|2.3e-02|4.5e+04| 1.827298e+05  1.422268e+05| 0:6:50| chol  1  1
 8|0.734|0.605|4.4e-11|1.1e-02|2.0e+04| 1.640164e+05  1.457914e+05| 0:7:49| chol  1  1
 9|0.709|0.636|1.1e-11|5.4e-03|9.6e+03| 1.560941e+05  1.473191e+05| 0:8:47| chol  1  1
10|0.762|0.607|2.9e-12|2.7e-03|4.2e+03| 1.517348e+05  1.479353e+05| 0:9:46| chol  1  1
11|0.722|0.584|2.1e-12|1.4e-03|1.9e+03| 1.499273e+05  1.482083e+05| 0:10:44| chol  1  1
12|0.713|0.661|2.0e-11|6.4e-04|9.7e+02| 1.492084e+05  1.483353e+05| 0:11:43| chol  1  1
13|0.798|0.773|2.5e-11|2.4e-04|3.9e+02| 1.487549e+05  1.483969e+05| 0:12:42| chol  2  2
14|0.797|0.759|1.4e-11|1.0e-04|1.6e+02| 1.485584e+05  1.484157e+05| 0:13:41| chol  1  2
15|0.784|0.620|1.6e-11|5.8e-05|6.8e+01| 1.484791e+05  1.484202e+05| 0:14:39| chol  1  2
16|0.886|0.814|3.8e-11|2.3e-05|2.3e+01| 1.484423e+05  1.484224e+05| 0:15:38| chol  2  2
17|0.805|0.725|1.0e-10|6.4e-06|8.5e+00| 1.484296e+05  1.484220e+05| 0:16:37| chol  2  2
18|0.744|0.851|1.2e-10|9.5e-07|3.6e+00| 1.484256e+05  1.484221e+05| 0:17:35| chol  2  2
19|0.859|0.847|2.5e-10|1.4e-07|1.3e+00| 1.484235e+05  1.484222e+05| 0:18:34| chol  2  2
20|1.000|0.741|1.2e-09|3.7e-08|4.0e-01| 1.484227e+05  1.484223e+05| 0:19:32| chol  2  2
21|0.704|0.942|4.7e-10|2.2e-09|1.7e-01| 1.484225e+05  1.484223e+05| 0:20:31| chol  3  2
22|0.860|0.801|1.5e-09|4.7e-10|2.6e-02| 1.484224e+05  1.484223e+05| 0:21:30| chol  2  2
23|0.928|0.966|4.6e-10|7.2e-11|2.3e-03| 1.484223e+05  1.484223e+05| 0:22:28|
```

# 实验二：压缩感知

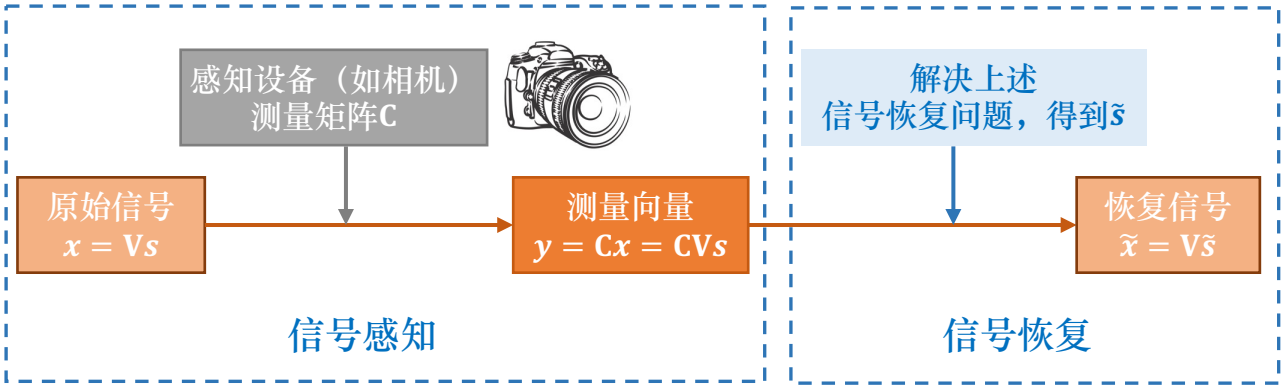



模拟信号恢复的过程：（接上）

```
V=dct(eye(n,n));
matrixCV=C*V;

cvx_begin;
  variable s(n);
  minimize(norm(s,1));
  subject to
    matrixCV*s==y;
cvx_end;

xrecovery=V*s;

rematrix=reshape(xrecovery,rsize,csize);
imwrite(uint8(rematrix),'recovery.jpg');
```

压缩至20%后恢复图像
$(p = 2000, n = 10000)$

# 实验二：压缩感知




模拟信号恢复的过程：（接上）

```
V=dct(eye(n,n));
matrixCV=C*V;
```

```
cvx_begin;
  variable s(n);
  minimize(norm(s,1));
  subject to
    matrixCV*s==y;
cvx_end;
```

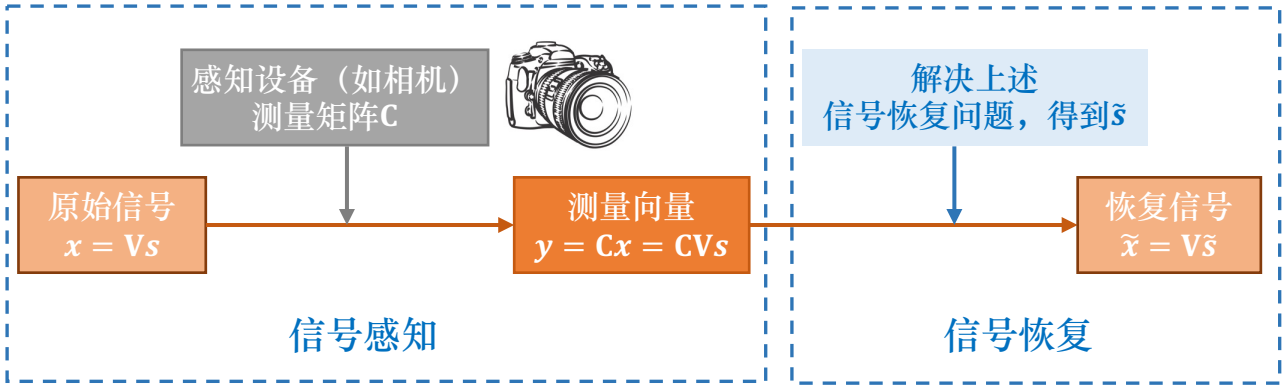
若用L2最小化问题替代L1最小化问题

```
s = pinv(matrixCV)*y;
```

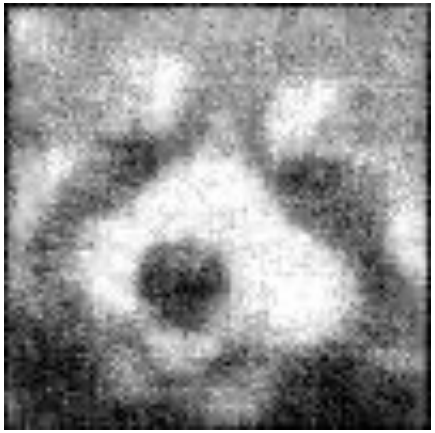
压缩至20%后恢复图像
$(p = 2000, n = 10000)$

```
xrecovery=V*s;

rematrix=reshape(xrecovery,rsize,csize);
imwrite(uint8(rematrix),'recovery.jpg');
```

# 实验二：压缩感知




原图片

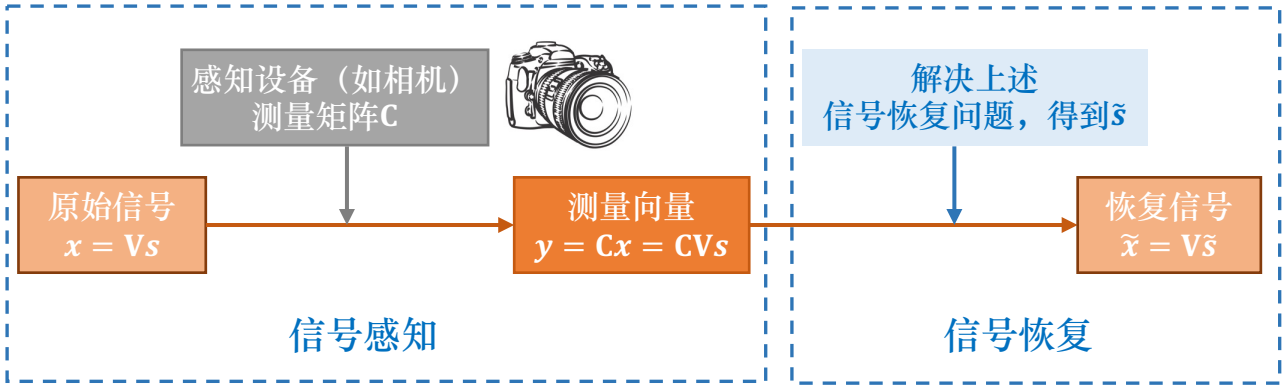


压缩至30%后恢复图像
$(p = 3000, n = 10000)$

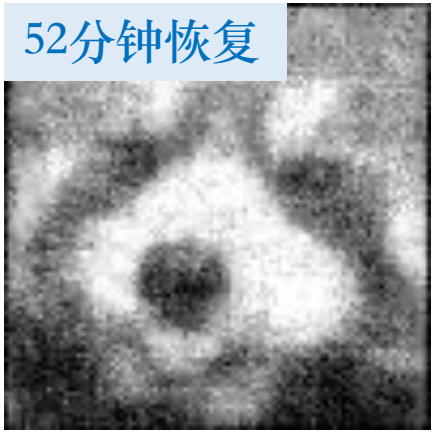
压缩至20%后恢复图像
$(p = 2000, n = 10000)$

压缩至10%后恢复图像
$(p = 1000, n = 10000)$

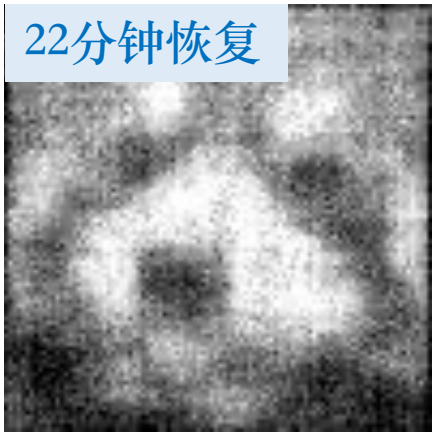
# 实验二：压缩感知



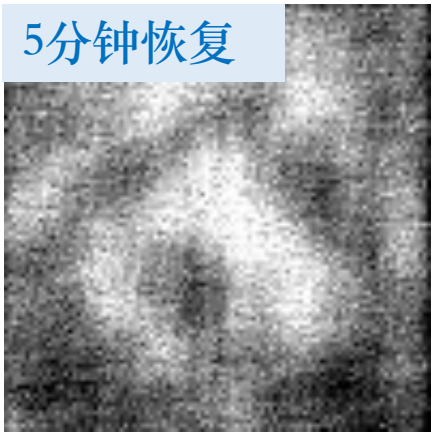

原图片




压缩至30%后恢复图像
$(p = 3000, n = 10000)$



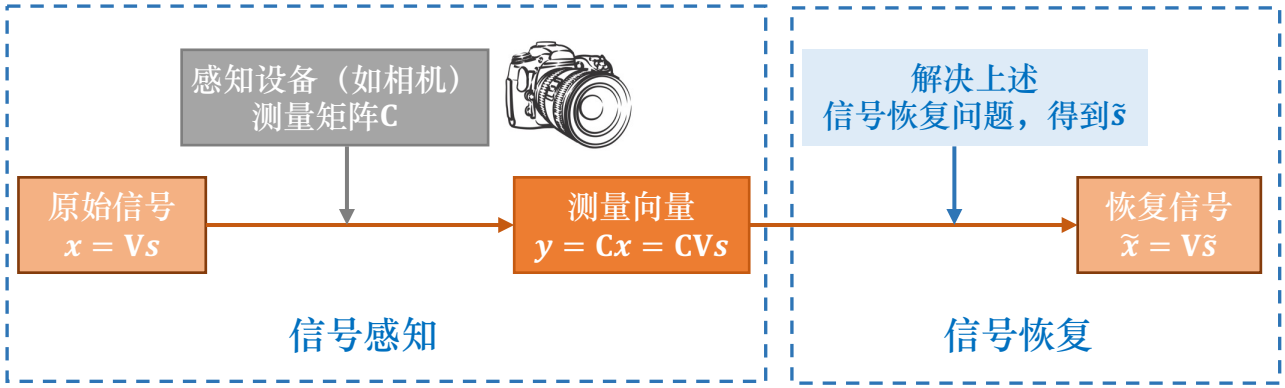

压缩至20%后恢复图像
$(p = 2000, n = 10000)$



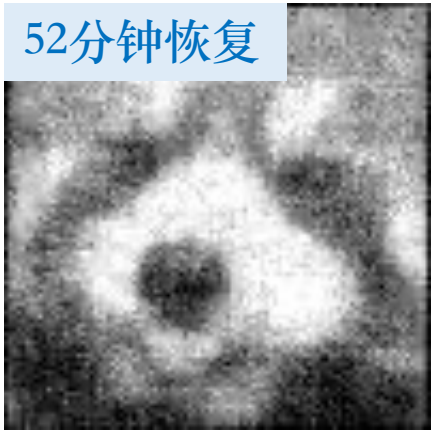

压缩至10%后恢复图像
$(p = 1000, n = 10000)$

信号恢复环节的最优化问题（L0/L1最小化问题）的求解需要消耗不少的计算资源

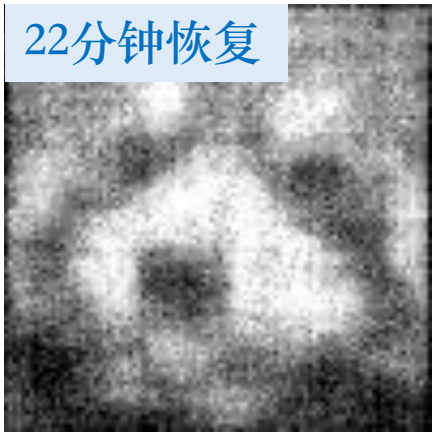
# 实验二：压缩感知



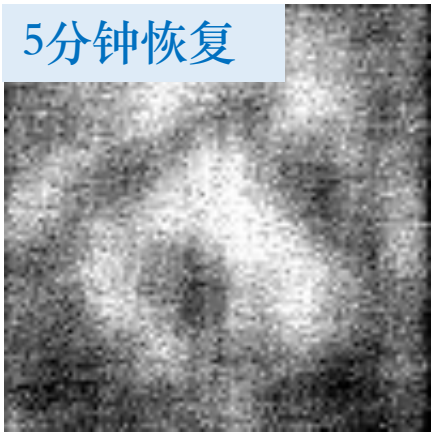

原图片




压缩至30%后恢复图像
$(p = 3000, n = 10000)$




压缩至20%后恢复图像
$(p = 2000, n = 10000)$




压缩至10%后恢复图像
$(p = 1000, n = 10000)$

信号恢复环节的最优化问题（L0/L1最小化问题）的求解需要消耗不少的计算资源

（1）实际中采用的求解算法更高效（如OMP、CoSaMp等算法）

（2）主要适用于对感知过程在资源消耗、感知时长等方面要求高的场景（如MRI）

（3）一些应用的瓶颈是在感知端，而后期恢复信号/图像一般有更充足的计算资源

# 压缩感知

## 基于正交贪心策略的求解法

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\boldsymbol{CVs} = \boldsymbol{y}$.

- 基于线性规划的求解：准确度高、算力消耗大

Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information
EJ Candès, J Romberg, T Tao - IEEE Transactions on …, 2006 - ieeexplore.ieee.org
This paper considers the model problem of reconstructing an object from incomplete frequency samples. Consider a discrete-time signal f/spl isin/C/sup N/and a randomly chosen set of frequencies/spl Omega/. Is it possible to reconstruct f from the partial knowledge of its Fourier coefficients on the set/spl Omega/? A typical result of this paper is as follows. Suppose that f is a superposition of| T| spikes f (t)=/spl sigma//sub/spl tau//spl isin/T/f (/spl tau/)/spl delta/(t-/spl tau/) obeying| T|/spl les/C/sub M//spl middot/(log N)/sup …
☆ Save 99 Cite Cited by 17967 Related articles All 32 versions

- 基于正交贪心策略的求解：准确度略逊、算力消耗小
  - OMP算法
  - CoSaMp算法

Signal recovery from random measurements via orthogonal matching pursuit
JA Tropp, AC Gilbert - IEEE Transactions on information theory, 2007 - ieeexplore.ieee.org
This paper demonstrates theoretically and empirically that a greedy algorithm called orthogonal matching pursuit (OMP) can reliably recover a signal with m nonzero entries in dimension d given O (m ln d) random linear measurements of that signal. This is a massive improvement over previous results, which require O (m 2) measurements. The new results for OMP are comparable with recent results for another approach called basis pursuit (BP). In some settings, the OMP algorithm is faster and easier to implement, so it is an attractive …
☆ Save 99 Cite Cited by 9783 Related articles All 34 versions

CoSaMP: Iterative signal recovery from incomplete and inaccurate samples
D Needell, JA Tropp - Applied and computational harmonic analysis, 2009 - Elsevier
Compressive sampling offers a new paradigm for acquiring signals that are compressible with respect to an orthonormal basis. The major algorithmic challenge in compressive sampling is to approximate a compressible signal from noisy samples. This paper describes a new iterative recovery algorithm called CoSaMP that delivers the same guarantees as the best optimization-based approaches. Moreover, this algorithm offers rigorous bounds on computational cost and storage. It is likely to be extremely efficient for practical problems …
☆ Save 99 Cite Cited by 4776 Related articles All 24 versions

# 正交贪心算法

求解问题

$$\min \|\boldsymbol{s}\|_0 \quad \text{s.t. } \mathbf{C}\mathbf{V}\boldsymbol{s} = \boldsymbol{y}.$$

## 正交匹配追踪算法（Orthogonal Matching Pursuit, OMP）

*Algorithm 3 (OMP for Signal Recovery):*

INPUT:

- An $N \times d$ measurement matrix $\mathbf{\Phi}$
- An $N$-dimensional data vector $\boldsymbol{v}$
- The sparsity level $m$ of the ideal signal

OUTPUT:

- An estimate $\hat{\boldsymbol{s}}$ in $\mathbb{R}^d$ for the ideal signal
- A set $\Lambda_m$ containing $m$ elements from $\{1, \ldots, d\}$
- An $N$-dimensional approximation $\boldsymbol{a}_m$ of the data $\boldsymbol{v}$
- An $N$-dimensional residual $\boldsymbol{r}_m = \boldsymbol{v} - \boldsymbol{a}_m$

PROCEDURE:

1) Initialize the residual $\boldsymbol{r}_0 = \boldsymbol{v}$, the index set $\Lambda_0 = \emptyset$, and the iteration counter $t = 1$.
2) Find the index $\lambda_t$ that solves the easy optimization problem

$$\lambda_t = \arg\max_{j=1,\ldots,d} |\langle \boldsymbol{r}_{t-1}, \boldsymbol{\varphi}_j \rangle| .$$

   If the maximum occurs for multiple indices, break the tie deterministically.
3) Augment the index set and the matrix of chosen atoms: $\Lambda_t = \Lambda_{t-1} \cup \{\lambda_t\}$ and $\mathbf{\Phi}_t = [\mathbf{\Phi}_{t-1} \quad \boldsymbol{\varphi}_{\lambda_t}]$. We use the convention that $\mathbf{\Phi}_0$ is an empty matrix.
4) Solve a least squares problem to obtain a new signal estimate:

$$\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} \|\boldsymbol{v} - \mathbf{\Phi}_t \boldsymbol{x}\|_2 .$$

5) Calculate the new approximation of the data and the new residual

$$\boldsymbol{a}_t = \mathbf{\Phi}_t \boldsymbol{x}_t$$
$$\boldsymbol{r}_t = \boldsymbol{v} - \boldsymbol{a}_t .$$

6) Increment $t$, and return to Step 2 if $t < m$.
7) The estimate $\hat{\boldsymbol{s}}$ for the ideal signal has nonzero indices at the components listed in $\Lambda_m$. The value of the estimate $\hat{\boldsymbol{s}}$ in component $\lambda_j$ equals the $j$th component of $\boldsymbol{x}_t$.

图片取自JA. Tropp, AC. Gilbert <Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit>

# 正交贪心算法

求解问题 $$\min \|\boldsymbol{s}\|_0 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

正交匹配追踪算法（Orthogonal Matching Pursuit, OMP）

主要利用两点性质：

（1）稀疏性：向量$\boldsymbol{s}$只有少数元素不为零（或不接近零）

（2）“不相干”：矩阵$\mathbf{CV}$的各列向量间相关性弱

# 正交贪心算法

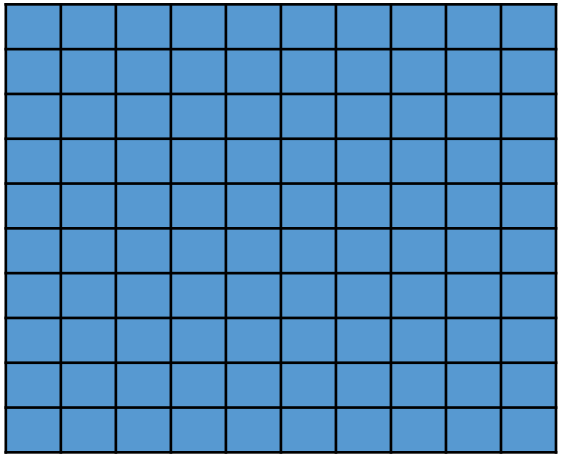
求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

正交匹配追踪算法（Orthogonal Matching Pursuit, OMP）

主要利用两点性质：

（1）稀疏性：向量$\boldsymbol{s}$只有少数元素不为零（或不接近零）

（2）“不相干”：矩阵$\mathbf{CV}$的各列向量间相关性弱



$n \times n$矩阵 $\mathbf{V}$

正交矩阵：
各列向量的内积为零

# 正交贪心算法

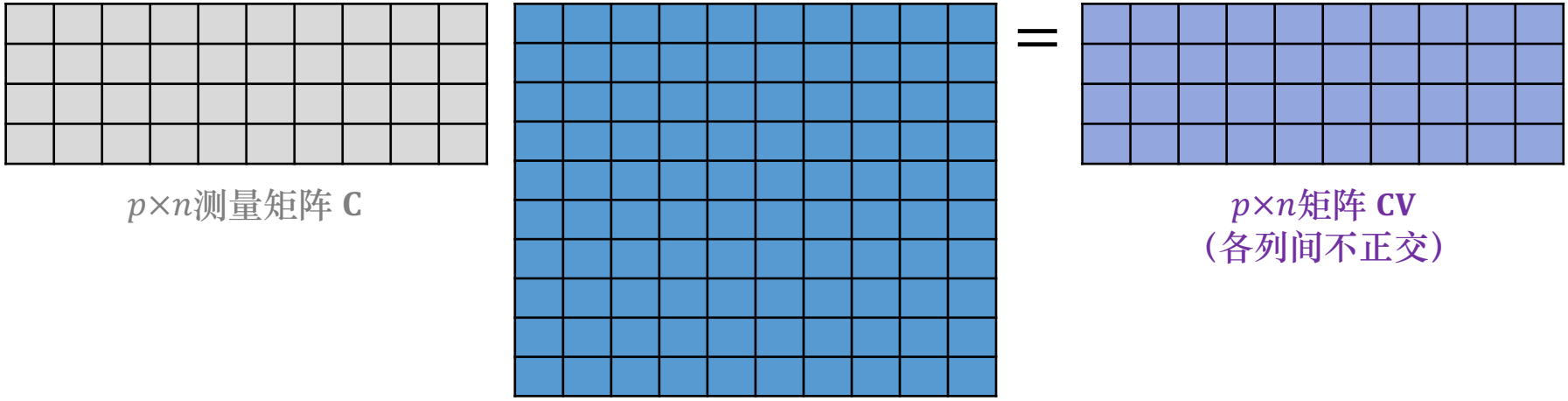
求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

正交匹配追踪算法（Orthogonal Matching Pursuit, OMP）

主要利用两点性质：

（1）稀疏性：向量$\boldsymbol{s}$只有少数元素不为零（或不接近零）

（2）“不相干”：矩阵CV的各列向量间相关性弱



$p \times n$测量矩阵 C

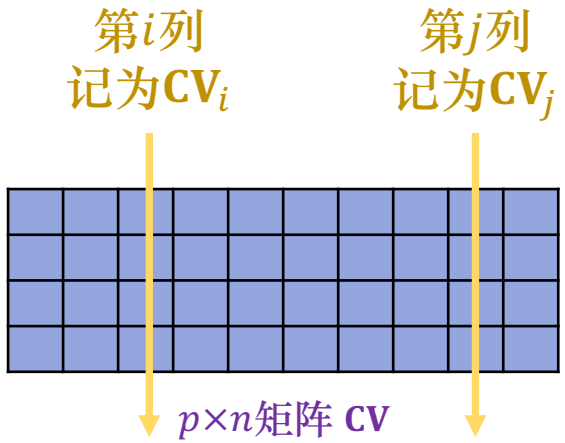
$n \times n$矩阵 V

$p \times n$矩阵 CV
（各列间不正交）

当矩阵C满足若干性质时（如各元素独立随机依标准正态分布取值）：
矩阵CV的各列相关性弱（即内积的绝对值很小，理解为“近似正交”）

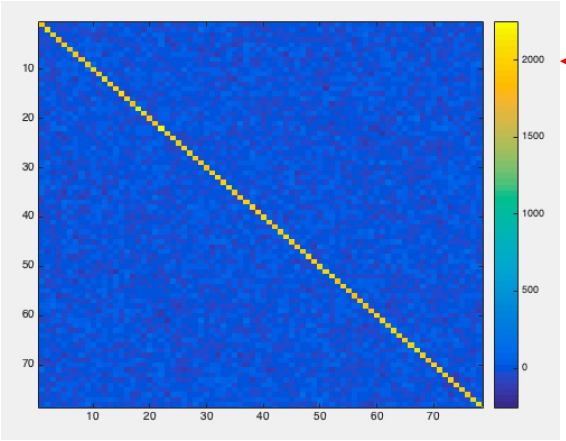
# 正交贪心算法

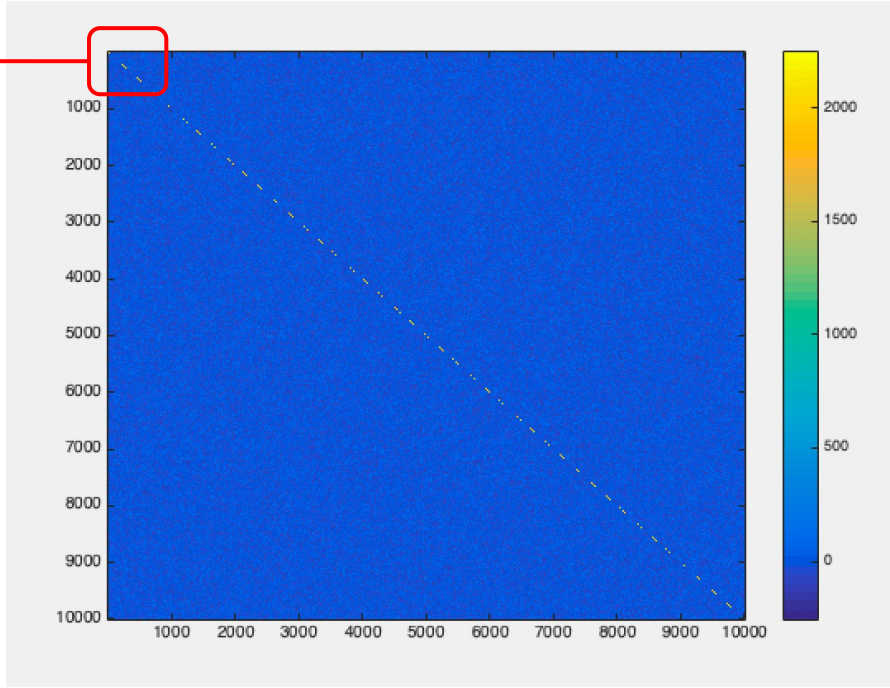
求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.




在前述实验中，有$p = 2000, n = 10000$

下图中第$i$行第$j$列显示$\mathbf{CV}_i^T\mathbf{CV}_j$




说明CV的不同列向量的内积的绝对值很小

# 正交贪心算法

求解问题

$$\min \|\boldsymbol{s}\|_0 \quad \text{s.t. } \mathbf{CV}\boldsymbol{s} = \boldsymbol{y}.$$

例：真实图像的$n\times 1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

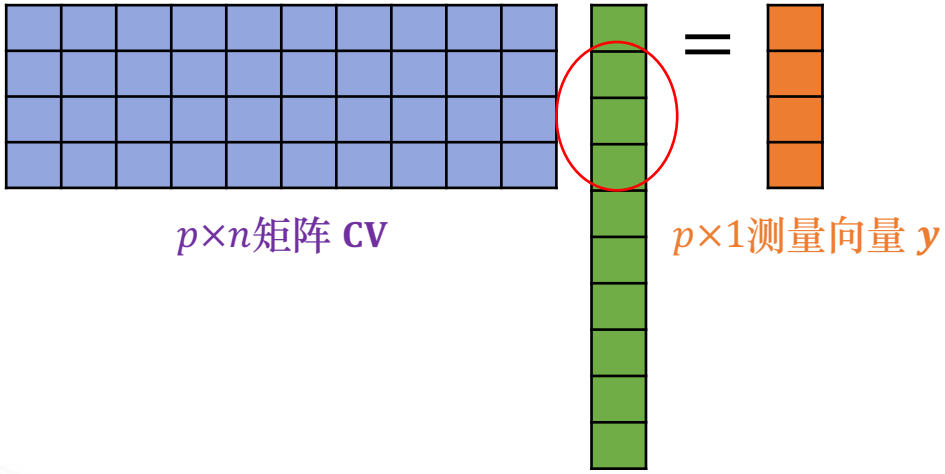
# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

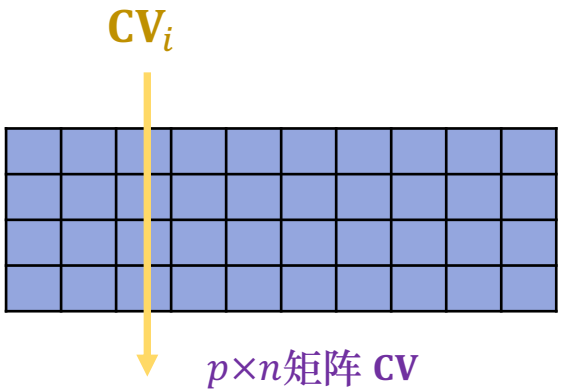
例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$



# 正交贪心算法




求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s}=\boldsymbol{y}$.

例：真实图像的$n\times 1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s}=[0,0.8,0.12,0.16,0,\dots,0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times 1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y}=\mathbf{CV}\boldsymbol{s}=\sum_{i=1}^{n}s_i\mathbf{CV}_i=0.8\mathbf{CV}_2+0.12\mathbf{CV}_3+0.16\mathbf{CV}_4$

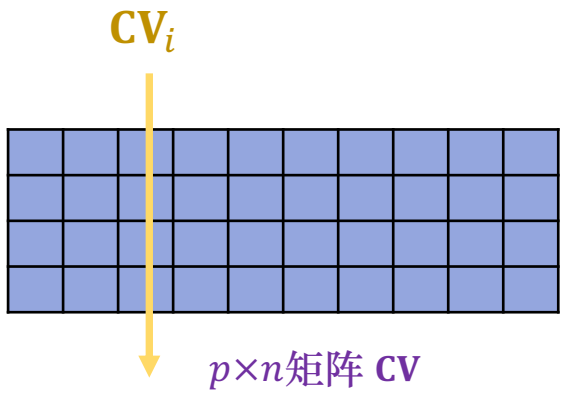
判断向量$\boldsymbol{s}$非零位置的方法：用$\mathbf{CV}_1^T,\mathbf{CV}_2^T,\dots,\mathbf{CV}_n^T$分别乘以向量$\boldsymbol{y}$

指绝对值，下略

用$\mathbf{CV}_1^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_1^T\mathbf{CV}_2+0.12\mathbf{CV}_1^T\mathbf{CV}_3+0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

CV不同列向量的内积绝对值很小

# 正交贪心算法




求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s}=\boldsymbol{y}$.

例：真实图像的$n\times 1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s}=[0,0.8,0.12,0.16,0,\dots,0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times 1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y}=\mathbf{CV}\boldsymbol{s}=\sum_{i=1}^{n}s_i\mathbf{CV}_i=0.8\mathbf{CV}_2+0.12\mathbf{CV}_3+0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$非零位置的方法：用$\mathbf{CV}_1^T,\mathbf{CV}_2^T,\dots,\mathbf{CV}_n^T$分别乘以向量$\boldsymbol{y}$

用$\mathbf{CV}_1^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_1^T\mathbf{CV}_2+0.12\mathbf{CV}_1^T\mathbf{CV}_3+0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_2^T\mathbf{CV}_2+0.12\mathbf{CV}_2^T\mathbf{CV}_3+0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_3^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_3^T\mathbf{CV}_2+0.12\mathbf{CV}_3^T\mathbf{CV}_3+0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_4^T\mathbf{CV}_2+0.12\mathbf{CV}_4^T\mathbf{CV}_3+0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_n^T\mathbf{CV}_2+0.12\mathbf{CV}_n^T\mathbf{CV}_3+0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

CV不同列向量的内积绝对值很小

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$非零位置的方法：用$\mathbf{CV}_1^T, \mathbf{CV}_2^T, \dots, \mathbf{CV}_n^T$分别乘以向量$\boldsymbol{y}$

用$\mathbf{CV}_1^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_1^T\mathbf{CV}_2 + 0.12\mathbf{CV}_1^T\mathbf{CV}_3 + 0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_2^T\mathbf{CV}_2 + 0.12\mathbf{CV}_2^T\mathbf{CV}_3 + 0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_3^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_3^T\mathbf{CV}_2 + 0.12\mathbf{CV}_3^T\mathbf{CV}_3 + 0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_4^T\mathbf{CV}_2 + 0.12\mathbf{CV}_4^T\mathbf{CV}_3 + 0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_n^T\mathbf{CV}_2 + 0.12\mathbf{CV}_n^T\mathbf{CV}_3 + 0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

理想情况：这三项（绝对）值最大，由此推断出向量$\boldsymbol{s}$的非零元素位置

实际情况：未必能一次同时找到这三项
因为向量$\boldsymbol{s}$有很多接近零但非零的元素、矩阵$\mathbf{CV}$的各列向量内积绝对值很小但不为零。

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$非零位置的方法：用$\mathbf{CV}_1^T, \mathbf{CV}_2^T, \dots, \mathbf{CV}_n^T$分别乘以向量$\boldsymbol{y}$

贪心算法：
每回合确定一个非零位置

用$\mathbf{CV}_1^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_1^T\mathbf{CV}_2 + 0.12\mathbf{CV}_1^T\mathbf{CV}_3 + 0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_2^T\mathbf{CV}_2 + 0.12\mathbf{CV}_2^T\mathbf{CV}_3 + 0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_3^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_3^T\mathbf{CV}_2 + 0.12\mathbf{CV}_3^T\mathbf{CV}_3 + 0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_4^T\mathbf{CV}_2 + 0.12\mathbf{CV}_4^T\mathbf{CV}_3 + 0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_n^T\mathbf{CV}_2 + 0.12\mathbf{CV}_n^T\mathbf{CV}_3 + 0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

第一回合，对比$n$个乘积，假设$\mathbf{CV}_2^T\boldsymbol{y}$（绝对）值最大：将第二个元素确定为$\boldsymbol{s}$的第一个非零位置

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$非零位置的方法：用$\mathbf{CV}_1^T, \mathbf{CV}_2^T, \dots, \mathbf{CV}_n^T$分别乘以向量$\boldsymbol{y}$

用$\mathbf{CV}_1^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_1^T\mathbf{CV}_2 + 0.12\mathbf{CV}_1^T\mathbf{CV}_3 + 0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_2^T\mathbf{CV}_2 + 0.12\mathbf{CV}_2^T\mathbf{CV}_3 + 0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_3^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_3^T\mathbf{CV}_2 + 0.12\mathbf{CV}_3^T\mathbf{CV}_3 + 0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_4^T\mathbf{CV}_2 + 0.12\mathbf{CV}_4^T\mathbf{CV}_3 + 0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以向量$\boldsymbol{y}$得$0.8\mathbf{CV}_n^T\mathbf{CV}_2 + 0.12\mathbf{CV}_n^T\mathbf{CV}_3 + 0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

贪心算法：
每回合确定一个非零位置

第一回合，对比$n$个乘积，假设$\mathbf{CV}_2^T\boldsymbol{y}$（绝对）值最大：将第二个元素确定为$\boldsymbol{s}$的第一个非零位置

在后续回合中，要抹去$\mathbf{CV}_2$相关的项对乘积绝对值大小对比的影响

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

已确定$\boldsymbol{s}$第二个元素非零，为在后续回合中抹去$\mathbf{CV}_2$相关项对乘积大小对比的影响，需修改向量$\boldsymbol{y}$

定义残差向量$\boldsymbol{r}$ $\boldsymbol{r} = \boldsymbol{y} - \tilde{s}_2 \mathbf{CV}_2$

估计的$s_2$（即0.8）的值

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

已确定$\boldsymbol{s}$第二个元素非零，为在后续回合中抹去$\mathbf{CV}_2$相关项对乘积大小对比的影响，需修改向量$\boldsymbol{y}$

定义残差向量$\boldsymbol{r}$ $\boldsymbol{r} = \boldsymbol{y} - \tilde{s}_2 \mathbf{CV}_2$

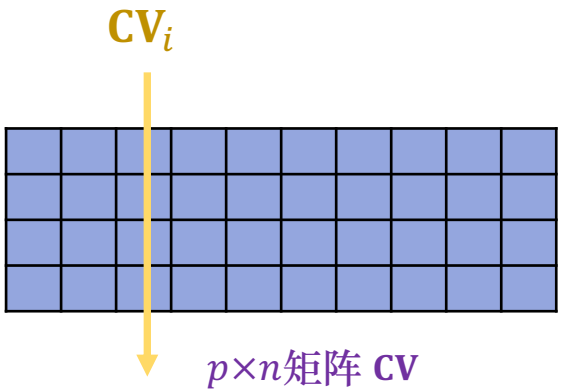
估计的$s_2$（即0.8）的值

$$\tilde{s}_2 = \underset{s_2}{\operatorname{argmin}} \|s_2 \mathbf{CV}_2 - \boldsymbol{y}\|_2$$

本质问题和线性回归相同（一个特征、$p$个数据）

是凸优化问题，而且有解析解，易计算

# 正交贪心算法




求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s}=\boldsymbol{y}$.

例：真实图像的$n\times 1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s}=[0,0.8,0.12,0.16,0,\dots,0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times 1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y}=\mathbf{CV}\boldsymbol{s}=\sum_{i=1}^{n}s_i\mathbf{CV}_i=0.8\mathbf{CV}_2+0.12\mathbf{CV}_3+0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$的第二个非零位置的方法：用$\mathbf{CV}_1^T,\mathbf{CV}_2^T,\dots,\mathbf{CV}_n^T$分别乘以残差向量$\boldsymbol{r}$

用$\mathbf{CV}_1^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_1^T\mathbf{CV}_2+0.12\mathbf{CV}_1^T\mathbf{CV}_3+0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以残差向量r得$(0.8-\tilde{s}_2)\mathbf{CV}_2^T\mathbf{CV}_2+0.12\mathbf{CV}_2^T\mathbf{CV}_3+0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_3^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_3^T\mathbf{CV}_2+0.12\mathbf{CV}_3^T\mathbf{CV}_3+0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_4^T\mathbf{CV}_2+0.12\mathbf{CV}_4^T\mathbf{CV}_3+0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_n^T\mathbf{CV}_2+0.12\mathbf{CV}_n^T\mathbf{CV}_3+0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

接近0

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$的第二个非零位置的方法：用$\mathbf{CV}_1^T, \mathbf{CV}_2^T, \dots, \mathbf{CV}_n^T$分别乘以残差向量$\boldsymbol{r}$

用$\mathbf{CV}_1^T$乘以残差向量$\boldsymbol{r}$得$(0.8 - \tilde{s}_2)\mathbf{CV}_1^T\mathbf{CV}_2 + 0.12\mathbf{CV}_1^T\mathbf{CV}_3 + 0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以残差向量r得$(0.8 - \tilde{s}_2)\mathbf{CV}_2^T\mathbf{CV}_2 + 0.12\mathbf{CV}_2^T\mathbf{CV}_3 + 0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_3^T$乘以残差向量$\boldsymbol{r}$得$(0.8 - \tilde{s}_2)\mathbf{CV}_3^T\mathbf{CV}_2 + 0.12\mathbf{CV}_3^T\mathbf{CV}_3 + 0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以残差向量$\boldsymbol{r}$得$(0.8 - \tilde{s}_2)\mathbf{CV}_4^T\mathbf{CV}_2 + 0.12\mathbf{CV}_4^T\mathbf{CV}_3 + 0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以残差向量$\boldsymbol{r}$得$(0.8 - \tilde{s}_2)\mathbf{CV}_n^T\mathbf{CV}_2 + 0.12\mathbf{CV}_n^T\mathbf{CV}_3 + 0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

第二回合，对比$n$个乘积，假设$\mathbf{CV}_3^T\boldsymbol{r}$（绝对）值最大：将第三个元素确定为$\boldsymbol{s}$的第二个非零位置

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$的第二个非零位置的方法：用$\mathbf{CV}_1^T, \mathbf{CV}_2^T, \dots, \mathbf{CV}_n^T$分别乘以残差向量$\boldsymbol{r}$

用$\mathbf{CV}_1^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_1^T\mathbf{CV}_2 + 0.12\mathbf{CV}_1^T\mathbf{CV}_3 + 0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以残差向量r得$(0.8-\tilde{s}_2)\mathbf{CV}_2^T\mathbf{CV}_2 + 0.12\mathbf{CV}_2^T\mathbf{CV}_3 + 0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_3^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_3^T\mathbf{CV}_2 + 0.12\mathbf{CV}_3^T\mathbf{CV}_3 + 0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很大

用$\mathbf{CV}_4^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_4^T\mathbf{CV}_2 + 0.12\mathbf{CV}_4^T\mathbf{CV}_3 + 0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_n^T\mathbf{CV}_2 + 0.12\mathbf{CV}_n^T\mathbf{CV}_3 + 0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

第二回合，对比$n$个乘积，假设$\mathbf{CV}_3^T\boldsymbol{r}$（绝对）值最大：将第三个元素确定为$\boldsymbol{s}$的第二个非零位置

在后续回合中，要抹去$\mathbf{CV}_2, \mathbf{CV}_3$相关的项对乘积绝对值大小对比的影响

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i\mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

已确定$\boldsymbol{s}$第二、第三元素非零，为抹去$\mathbf{CV}_2, \mathbf{CV}_3$相关项对乘积大小对比的影响，需修改向量$\boldsymbol{y}$

更新残差向量$\boldsymbol{r}$ $\boldsymbol{r} = \boldsymbol{y} - \tilde{s}_2\mathbf{CV}_2 - \tilde{s}_3\mathbf{CV}_3$

估计的$s_2, s_3$的值

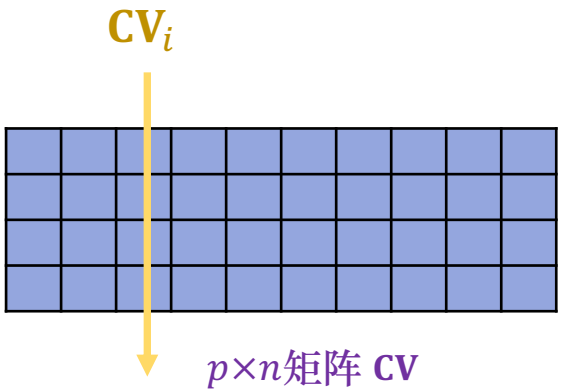
注意对$\tilde{s}_2$，要重新估计

$$(\tilde{s}_2, \tilde{s}_3) = \underset{(s_2, s_3)}{\operatorname{argmin}} \|s_2\mathbf{CV}_2 + s_3\mathbf{CV}_3 - \boldsymbol{y}\|_2$$

本质问题和线性回归相同（两个特征、$p$个数据）

是凸优化问题，而且有解析解，易计算

# 正交贪心算法




求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s}=\boldsymbol{y}$.

例：真实图像的$n\times 1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零，$\boldsymbol{s}=[0,0.8,0.12,0.16,0,\dots,0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times 1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y}=\mathbf{CV}\boldsymbol{s}=\sum_{i=1}^{n}s_i\mathbf{CV}_i=0.8\mathbf{CV}_2+0.12\mathbf{CV}_3+0.16\mathbf{CV}_4$

判断向量$\boldsymbol{s}$的第三个非零位置的方法：用$\mathbf{CV}_1^T,\mathbf{CV}_2^T,\dots,\mathbf{CV}_n^T$分别乘以更新后的残差向量$\boldsymbol{r}$

用$\mathbf{CV}_1^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_1^T\mathbf{CV}_2+(0.12-\tilde{s}_3)\mathbf{CV}_1^T\mathbf{CV}_3+0.16\mathbf{CV}_1^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_2^T$乘以残差向量r得$(0.8-\tilde{s}_2)\mathbf{CV}_2^T\mathbf{CV}_2+(0.12-\tilde{s}_3)\mathbf{CV}_2^T\mathbf{CV}_3+0.16\mathbf{CV}_2^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_3^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_3^T\mathbf{CV}_2+(0.12-\tilde{s}_3)\mathbf{CV}_3^T\mathbf{CV}_3+0.16\mathbf{CV}_3^T\mathbf{CV}_4$：值很小

用$\mathbf{CV}_4^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_4^T\mathbf{CV}_2+(0.12-\tilde{s}_3)\mathbf{CV}_4^T\mathbf{CV}_3+0.16\mathbf{CV}_4^T\mathbf{CV}_4$：值很大

…

用$\mathbf{CV}_n^T$乘以残差向量$\boldsymbol{r}$得$(0.8-\tilde{s}_2)\mathbf{CV}_n^T\mathbf{CV}_2+(0.12-\tilde{s}_3)\mathbf{CV}_n^T\mathbf{CV}_3+0.16\mathbf{CV}_n^T\mathbf{CV}_4$：值很小

接近0　　接近0

第三回合：
将第四个元素确定为$\boldsymbol{s}$的第三个非零位

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

例：真实图像的$n\times1$向量$\boldsymbol{s}$仅在第二、第三、第四元素取值非零， $\boldsymbol{s} = [0, 0.8, 0.12, 0.16, 0, \dots, 0]^T$

如何能根据$\boldsymbol{y}$定位出向量$\boldsymbol{s}$在这三处取值非零?

$p\times1$测量向量$\boldsymbol{y}$满足$\boldsymbol{y} = \mathbf{CV}\boldsymbol{s} = \sum_{i=1}^{n} s_i \mathbf{CV}_i = 0.8\mathbf{CV}_2 + 0.12\mathbf{CV}_3 + 0.16\mathbf{CV}_4$

已确定$\boldsymbol{s}$第二、第三、第四元素非零，如何还原$\boldsymbol{s}$?

$$(\tilde{s}_2, \tilde{s}_3, \tilde{s}_4) = \underset{(s_2, s_3, s_4)}{\operatorname{argmin}} \|s_2\mathbf{CV}_2 + s_3\mathbf{CV}_3 + s_4\mathbf{CV}_4 - \boldsymbol{y}\|_2$$

输出还原的向量$\tilde{\boldsymbol{s}} = [0, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, 0, \dots, 0]^T$

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

**正交匹配追踪算法（Orthogonal Matching Pursuit Algorithm, OMP）**

输入：矩阵$\mathbf{CV}$、向量$\boldsymbol{y}$、稀疏向量稀疏度$K$

输出：向量$\tilde{\boldsymbol{s}}$

0 初始化：残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y}$，非零位置集合$\mathcal{O} \leftarrow \emptyset$

1 for $i = 1, 2, \ldots, K$

2 计算$o = \underset{i}{\operatorname{argmax}} \left|\mathbf{CV}_i^T \boldsymbol{r}\right|$ $o$是该回合找到的非零元素的位置

3 更新非零位置集合$\mathcal{O} \leftarrow \mathcal{O} \cup \{o\}$

4 计算$(\tilde{s}_i, i \in \mathcal{O}) = \underset{(s_i, i \in \mathcal{O})}{\operatorname{argmin}} \|\sum_{i \in \mathcal{O}} s_i \mathbf{CV}_i - \boldsymbol{y}\|_2$ 估计非零元素的值

5 更新残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y} - \tilde{s}_i \mathbf{CV}_i$

6 $\tilde{s}_i = 0, \forall i \notin \mathcal{O}$

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

根据经验取或尝试不同的$K$值比较效果

**正交匹配追踪算法（Orthogonal Matching Pursuit Algorithm, OMP）**

输入：矩阵$\mathbf{CV}$、向量$\boldsymbol{y}$、稀疏向量稀疏度$K$

输出：向量$\tilde{\boldsymbol{s}}$

0 初始化：残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y}$，非零位置集合$\mathcal{O} \leftarrow \emptyset$

1 for $i = 1, 2, \ldots, K$

2 计算$o = \underset{i}{\operatorname{argmax}} \left|\mathbf{CV}_i^T \boldsymbol{r}\right|$ $o$是该回合找到的非零元素的位置

3 更新非零位置集合$\mathcal{O} \leftarrow \mathcal{O} \cup \{o\}$

4 计算$(\tilde{s}_i, i \in \mathcal{O}) = \underset{(s_i, i \in \mathcal{O})}{\operatorname{argmin}} \|\sum_{i \in \mathcal{O}} s_i \mathbf{CV}_i - \boldsymbol{y}\|_2$ 估计非零元素的值

5 更新残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y} - \tilde{s}_i \mathbf{CV}_i$

6 $\tilde{s}_i = 0, \forall i \notin \mathcal{O}$

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

/Users/haoranyu/Documents/MATLAB/OMP.m

EDIT... PUBL... VI...

```
1  function [s]=OMP(K,y,matrixCV)
2  res=y;
3  [p,n]=size(matrixCV);
4  backmatrix=zeros(p,n);
5  s=zeros(n,1);
6  saveindex=zeros(K);
7
8  for i=1:K
9    [~,index]=max(abs(matrixCV'*res));
10   saveindex(i)=index;
11
12   backmatrix(:,i)=matrixCV(:,index);
13   partmatrix=backmatrix(:,1:i);
14   parts=pinv(partmatrix)*y;
15
16   for j=1:i
17   l=saveindex(j);
18   s(l)=parts(j);
19   end
20
21   res=y-matrixCV*s;
22 end
```

OMP Ln 7 Col 1

## 正交匹配追踪算法

输入：矩阵$\mathbf{CV}$、向量$\boldsymbol{y}$、稀疏向量稀疏度$K$

输出：向量$\tilde{\boldsymbol{s}}$

0 初始化：残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y}$，非零位置集合$\mathcal{O} \leftarrow \emptyset$

1 for $i = 1, 2, \ldots, K$

2 计算$o = \underset{i}{\operatorname{argmax}} \left|\mathbf{CV}_i^T \boldsymbol{r}\right|$

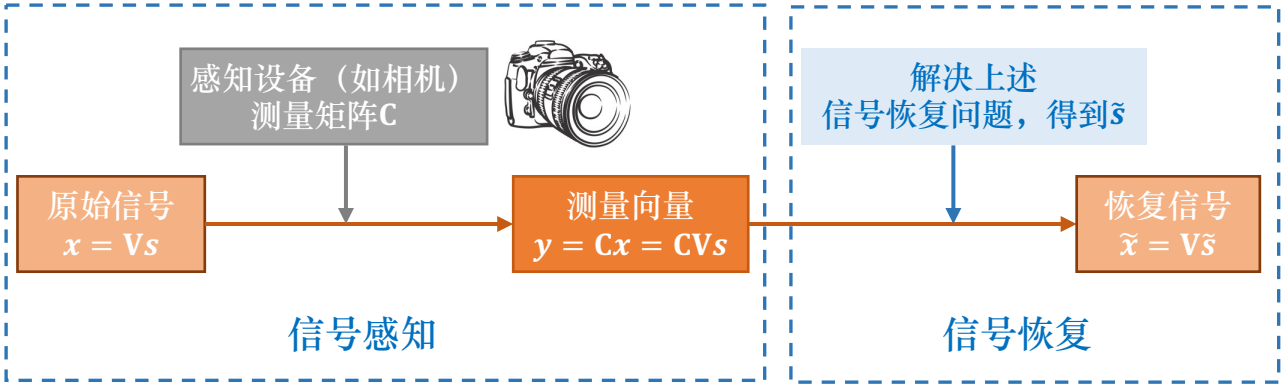
3 更新非零位置集合$\mathcal{O} \leftarrow \mathcal{O} \cup \{o\}$

4 计算$(\tilde{s}_i, i \in \mathcal{O}) = \underset{(s_i, i \in \mathcal{O})}{\operatorname{argmin}} \|\sum_{i \in \mathcal{O}} s_i \mathbf{CV}_i - \boldsymbol{y}\|_2$

5 更新残差向量$\boldsymbol{r} \leftarrow \boldsymbol{y} - \tilde{s}_i \mathbf{CV}_i$

6 $\tilde{s}_i = 0, \forall i \notin \mathcal{O}$

# 实验三：OMP算法

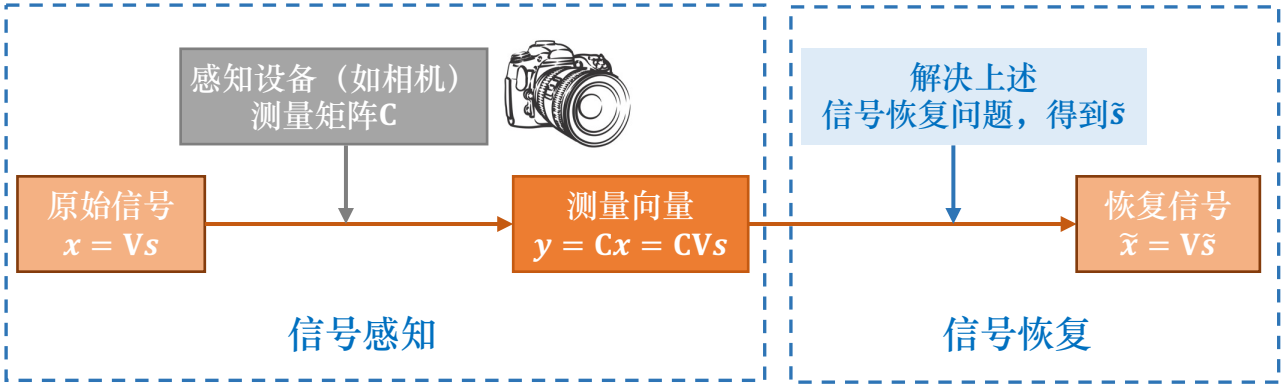



```
A=imread('animal','png');
Abw=rgb2gray(A);
[rsize,csize]=size(Abw);
n=rsize*csize;
p=round(0.2*n);
```

压缩至20%

```
x=reshape(Abw,[],1);
C=normrnd(0,1,p,n);
y=C*double(x);
```

# 实验三：OMP算法



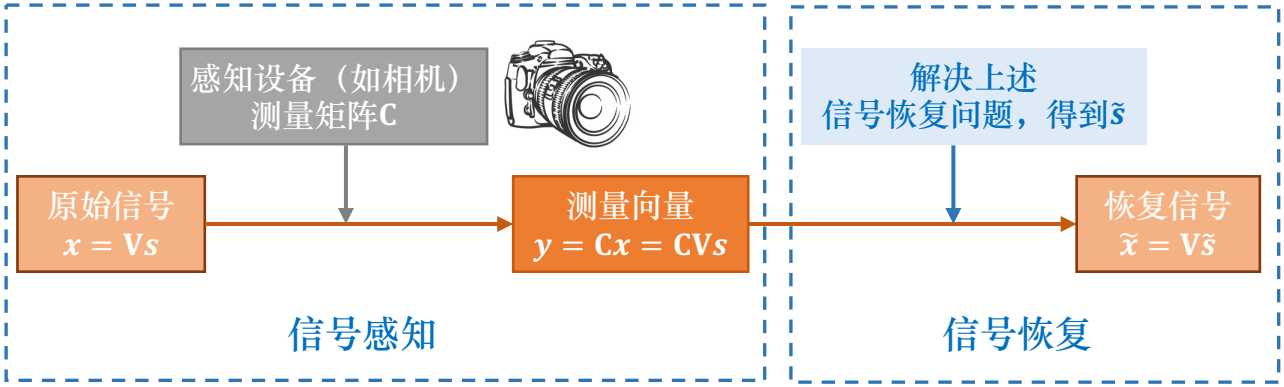

```
A=imread('animal','png');
Abw=rgb2gray(A);
[rsize,csize]=size(Abw);
n=rsize*csize;
p=round(0.2*n);
```

压缩至20%

```
x=reshape(Abw,[],1);
C=normrnd(0,1,p,n);
y=C*double(x);

V=dct(eye(n,n));
matrixCV=C*V;
s=OMP(1000,y,matrixCV);
```

利用OMP算法求稀疏向量（稀疏度设为1000）

# 实验三：OMP算法



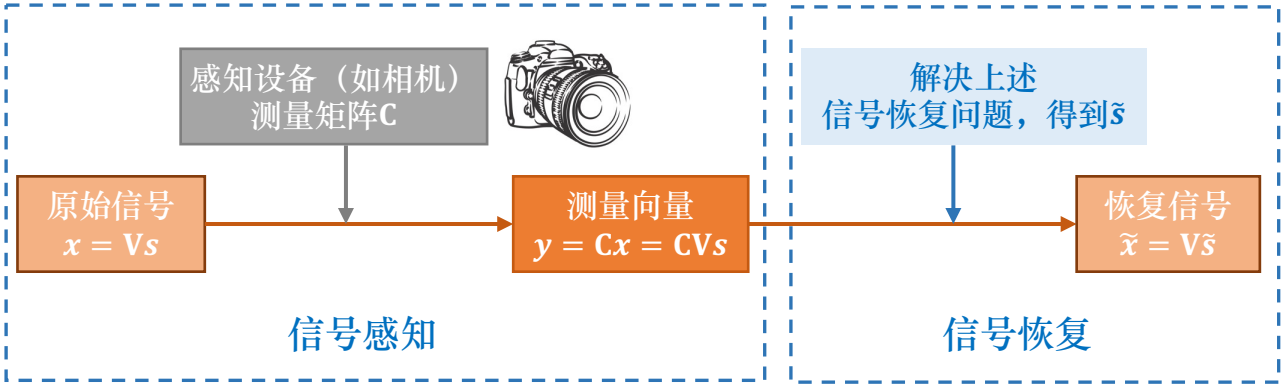

```
A=imread('animal','png');
Abw=rgb2gray(A);
[rsize,csize]=size(Abw);
n=rsize*csize;
p=round(0.2*n);
```

压缩至20%

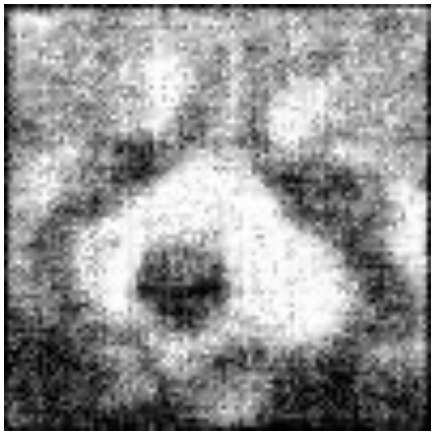
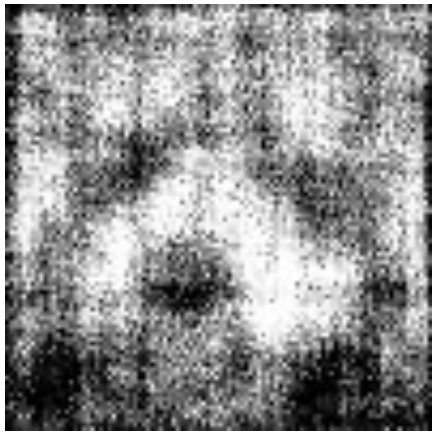
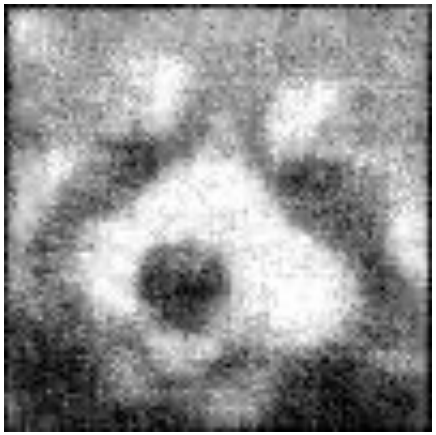
```
x=reshape(Abw,[],1);
C=normrnd(0,1,p,n);
y=C*double(x);

V=dct(eye(n,n));
matrixCV=C*V;
s=OMP(1000,y,matrixCV);

xrecovery=V*s;

rematrix=reshape(xrecovery,rsize,csize);
imwrite(uint8(rematrix),'recovery.jpg');
```

压缩至20%后恢复图像
$(p = 2000, n = 10000)$

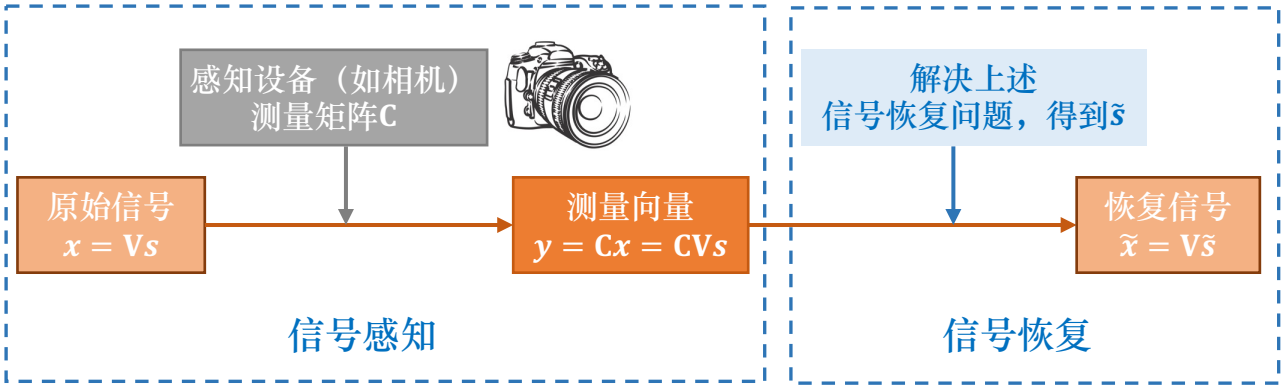
# 实验三：OMP算法




原图片

| | 压缩至30%后恢复图像 $(p = 3000, n = 10000)$ | 压缩至20%后恢复图像 $(p = 2000, n = 10000)$ | 压缩至10%后恢复图像 $(p = 2000, n = 10000)$ |
|---|---|---|---|
| OMP方法 |  |  |  |
| 线性规划方法 |  |  |  |

# 实验三：OMP算法




原图片

| | 压缩至30%后恢复图像 $(p = 3000, n = 10000)$ | 压缩至20%后恢复图像 $(p = 2000, n = 10000)$ | 压缩至10%后恢复图像 $(p = 2000, n = 10000)$ |
|---|---|---|---|
| OMP方法 | 5分钟恢复 $(K = 1000)$ | 4分钟恢复 $(K = 1000)$ | 半分钟恢复 $(K = 500)$ |
| 线性规划方法 | 52分钟恢复 | 22分钟恢复 | 5分钟恢复 |

# 正交贪心算法

求解问题 $\min \|\boldsymbol{s}\|_0$ s.t. $\mathbf{CV}\boldsymbol{s} = \boldsymbol{y}$.

- 基于线性规划的求解：准确度高、算力消耗大
- 基于正交贪心策略的求解：准确度略逊、算力消耗小
  - OMP算法

Signal recovery from random measurements via orthogonal matching pursuit
JA Tropp, AC Gilbert - IEEE Transactions on information theory, 2007 - ieeexplore.ieee.org
This paper demonstrates theoretically and empirically that a greedy algorithm called orthogonal matching pursuit (OMP) can reliably recover a signal with m nonzero entries in dimension d given O (m ln d) random linear measurements of that signal. This is a massive improvement over previous results, which require O (m 2) measurements. The new results for OMP are comparable with recent results for another approach called basis pursuit (BP). In some settings, the OMP algorithm is faster and easier to implement, so it is an attractive …
☆ Save 99 Cite Cited by 9783 Related articles All 34 versions

受OMP算法启发，有许多拓展算法（如每回合确定若干非零元素位置、以此减少循环次数；实现自适应确定$K$值）

相关论文的主要贡献是在理论上证明算法的准确性、收敛性、鲁棒性等

# 本讲小结

- 信号恢复最优化问题
- 基于线性规划及基于正交贪心策略的求解

计算机学院
余皓然
2024/6/6

# 主要参考资料

Tim Roughgarden and Gregory Valiant <CS 168 - The Modern Algorithmic Toolbox> Lecture Notes

Steven L. Brunton, J. Nathan Kutz <Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control> Book

JA. Tropp, AC. Gilbert <Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit> Paper

清华大学张颢 <现代数字信号处理2> 课堂视频

# 谢谢！



计算机学院
余皓然
2024/6/6