



A tutorial on Artificial Intelligence

发布 0.1

Zhi Liu

2020 年 02 月 27 日

Contents

1 引言	3
1.1 手册自述	3
1.1.1 目标	3
1.1.2 撰写与发布	3
2 第一卷数学基础	5
2.1 引言	5
2.2 概念解析	5
2.2.1 空间	5
2.2.1.1 空间简介	5
2.2.1.2 度量空间	5
2.2.1.3 希尔伯特空间	5
2.2.1.4 巴拿赫空间	6
2.2.1.5 拓扑空间	6
2.2.1.6 黎曼空间	6
2.2.2 问题	7
2.2.2.1 简介	7
2.2.2.2 病态与良态	7
2.2.2.2.1 条件数	7
2.2.2.2.2 病态的	8
2.2.2.2.3 良态的	8
2.2.2.3 适定与不适定	8
2.2.2.3.1 不适定问题	8
2.2.2.3.2 适定问题	8
2.2.2.4 逆问题	8
2.2.2.4.1 线性逆问题	8
2.2.2.4.2 非线性逆问题	8
2.2.3 正则化	8
2.2.3.1 正则化简介	8
2.2.3.2 范数正则	9

2.2.3.2.1	常见范数正则	9
2.2.3.2.2	Tikhonov 正则	9
2.2.3.3	稀疏正则	9
2.2.3.4	流形正则	9
2.2.3.5	噪声正则	9
2.2.4	名词术语	9
2.3	数论	10
2.3.1	素数问题	10
2.3.2	名词术语	10
2.4	高等数学	10
2.4.1	引言	10
2.4.2	集合论	10
2.4.2.1	集合的概念与性质	10
2.4.2.1.1	一般集合	10
2.4.2.1.1.1	概念	10
2.4.2.1.1.2	元素与集合的关系	11
2.4.2.1.1.3	集合与集合的关系	11
2.4.2.1.1.4	集合间的运算	11
2.4.2.1.1.5	集合间的运算定律	12
2.4.2.1.1.6	集合的特征函数	12
2.4.2.1.1.7	集合的势与基数	12
2.4.2.1.2	其它集合	12
2.4.2.1.2.1	模糊集	12
2.4.2.1.2.2	粗糙集	12
2.4.2.3	名词术语	13
2.5	矩阵论	13
2.5.1	线性空间与线性变换	13
2.5.1.1	线性空间	13
2.5.1.1.1	基础概念	13
2.5.1.1.1.1	域	13
2.5.1.1.1.2	映射	13
2.5.1.1.2	线性空间概念及性质	13
2.5.1.1.2.1	定义	13
2.5.1.1.2.2	性质	14
2.5.1.1.2.3	线性组合与线性表示	15
2.5.1.1.2.4	线性相关与线性无关	15
2.5.1.1.3	线性空间的基、坐标与维数	15
2.5.1.1.3.1	定义	15
2.5.1.1.4	基变换与坐标变换	16
2.5.1.1.4.1	两个基之间的转换	16
2.5.1.1.4.2	多个基之间的转换	17
2.5.1.1.4.3	坐标变换	17
2.5.1.2	线性子空间	18
2.5.1.2.1	线性子空间	18
2.5.1.2.1.1	什么是线性子空间	18

2.5.1.2.1.2	子空间的运算及其性质	18
2.5.1.2.2	直和	19
2.5.1.2.2.1	什么是直和	19
2.5.1.2.2.2	性质	19
2.5.1.2.3	生成的子空间	19
2.5.1.2.3.1	什么是生成子空间	19
2.5.1.2.3.2	矩阵的列空间、核空间、秩与零度	20
2.5.1.2.3.3	矩阵的 Spark	20
2.5.1.2.3.4	总结	21
2.5.1.3	线性变换及其表示	21
2.5.1.3.1	线性变换	21
2.5.1.3.1.1	什么是线性变换	21
2.5.1.3.1.2	线性变换的特性	22
2.5.1.3.2	线性变换的运算	22
2.5.1.3.2.1	基本线性变换	22
2.5.1.3.2.2	单位变换	22
2.5.1.3.2.3	零变换	22
2.5.1.3.2.4	负变换	22
2.5.1.3.2.5	逆变换	23
2.5.1.3.2.6	加法	23
2.5.1.3.2.7	定义	23
2.5.1.3.2.8	性质	23
2.5.1.3.2.9	数乘	23
2.5.1.3.2.10	定义	23
2.5.1.3.2.11	性质	24
2.5.1.3.2.12	乘法	24
2.5.1.3.2.13	定义	24
2.5.1.3.2.14	性质	24
2.5.1.3.2.15	幂	24
2.5.1.3.2.16	定义	24
2.5.1.3.2.17	性质	25
2.5.1.3.2.18	线性变换的多项式	25
2.5.1.3.2.19	定义	25
2.5.1.3.2.20	性质	25
2.5.1.3.3	线性变换与矩阵	25
2.5.1.3.3.1	线性变换的矩阵表示	25
2.5.1.3.3.2	性质	27
2.5.1.3.3.3	像与原像的坐标转换	27
2.5.1.3.3.4	线性变换在不同基下的矩阵转换	27
2.5.1.3.3.5	相似矩阵	28
2.5.1.3.3.6	什么是相似矩阵	28
2.5.1.3.3.7	性质	28
2.5.1.3.3.8	线性变换多项式的矩阵	28
2.5.1.3.3.9	常见线性变换	29
2.5.1.3.4	总结	32

2.5.1.4 特征值与特征向量	32
2.5.1.4.1 特征值与特征向量	32
2.5.1.4.1.1 线性变换的特征值与特征向量	32
2.5.1.4.1.2 矩阵的特征值与特征向量	32
2.5.1.4.1.3 特征值与特征向量的关系	32
2.5.1.4.1.4 实例	33
2.5.1.4.2 广义特征向量	35
2.5.1.4.2.1 概念与内涵	35
2.5.1.4.2.2 计算步骤	35
2.5.1.4.3 例子(广义特征向量)	36
2.5.1.4.4 特征多项式与最小多项式	36
2.5.1.4.4.1 特征多项式	36
2.5.1.4.4.2 最小多项式	37
2.5.1.4.5 特征子空间与不变子空间	37
2.5.1.4.5.1 什么是特征子空间	37
2.5.1.4.5.2 什么是不变子空间	38
2.5.1.4.6 信号子空间与噪声子空间	38
2.5.1.5 矩阵对角化	38
2.5.1.5.1 多项式矩阵及其标准形	38
2.5.1.5.1.1 多项式矩阵	38
2.5.1.5.1.2 首一多项式	38
2.5.1.5.1.3 多项式矩阵的标准形	38
2.5.1.5.2 不变因子与初等因子	39
2.5.1.5.2.1 不变因子	39
2.5.1.5.2.2 初等因子	39
2.5.1.5.2.3 举例	39
2.5.1.5.3 Jordan 标准型	40
2.5.1.5.3.1 概念与内涵	40
2.5.1.5.3.2 求解方法	40
2.5.1.5.3.3 举例	40
2.5.1.5.3.4 非奇异矩阵 P 的求法	42
2.5.1.5.4 Jordan 标准型求解微分线性方程组	43
2.5.1.5.4.1 计算步骤	43
2.5.1.5.4.2 举例	44
2.5.1.5.5 总结	45
2.5.1.5.5.1 可对角化	45
2.5.1.6 广义特征值与特征向量	45
2.5.1.6.1 广义特征值与特征向量	45
2.5.1.6.1.1 矩阵的广义特征值与特征向量	45
2.5.1.6.1.2 特征值与特征向量的关系	45
2.5.1.6.1.3 广义特征值问题的等价形式	45
2.5.1.6.1.4 第一种形式	45
2.5.1.6.1.5 第二种形式	46
2.5.1.6.1.6 实例	46
2.5.1.7 特殊线性空间	46

2.5.1.7.1	内积空间	46
2.5.1.7.1.1	什么是内积空间	46
2.5.1.7.1.2	欧式空间 (实内积空间)	47
2.5.1.7.1.3	酉空间 (复内积空间)	47
2.5.1.7.1.4	常见内积及其性质	47
2.5.1.7.1.5	定义	47
2.5.1.7.1.6	性质	47
2.5.1.7.1.7	度量矩阵	48
2.5.1.7.1.8	Schmidt 正交化	48
2.5.1.7.1.9	内积空间中线性变换	49
2.5.1.7.1.10	实对称变换	50
2.5.1.7.1.11	实反对称变换	50
2.5.1.7.1.12	正交变换	50
2.5.1.7.1.13	酉对称变换	51
2.5.1.7.1.14	反酉对称变换	52
2.5.1.7.1.15	酉变换	52
2.5.1.7.1.16	Givens 旋转变换	53
2.5.1.7.1.17	Householder 反射变换	54
2.5.1.7.1.18	谱分解	55
2.5.1.7.2	总结	55
2.5.2	范数理论及应用	56
2.5.2.1	向量范数	56
2.5.2.1.1	概念与内涵	56
2.5.2.1.2	向量范数的等价性	56
2.5.2.1.3	常见向量范数	57
2.5.2.1.3.1	普通向量范数	57
2.5.2.1.3.2	加权范数	57
2.5.2.2	矩阵范数	58
2.5.2.2.1	概念与内涵	58
2.5.2.2.2	矩阵范数与向量范数的相容性	58
2.5.2.2.3	常见矩阵范数	58
2.5.2.2.3.1	从属范数	59
2.5.2.3	范数的应用	59
2.5.2.3.1	矩阵的谱半径	59
2.5.2.3.1.1	什么是谱半径	59
2.5.2.3.1.2	性质	60
2.5.2.3.2	矩阵非奇异性条件	60
2.5.2.3.3	逆矩阵的摄动与条件数	60
2.5.2.3.3.1	矩阵的摄动	61
2.5.2.3.3.2	条件数	61
2.5.3	矩阵分析及应用	61
2.5.3.1	矩阵级数	61
2.5.3.1.1	概念	61
2.5.3.2	矩阵序列	61
2.5.3.2.1	概念理解	61

2.5.3.2.2 矩阵序列收敛与矩阵收敛	61
2.5.3.3 矩阵函数	61
2.5.3.3.1 概念	61
2.5.3.3.2 求解方法	61
2.5.3.3.2.1 待定系数法	62
2.5.3.3.2.2 数项级数求和法	62
2.5.3.3.2.3 对角矩阵法	62
2.5.3.3.2.4 Jordan 标准型法	62
2.5.3.3.3 总结	62
2.5.3.4 矩阵微分与积分	63
2.5.3.4.1 概念	63
2.5.3.5 总结	63
2.5.3.5.1 概念对比	63
2.5.3.5.2 收敛定理	63
2.5.4 矩阵分解	64
2.5.4.1 简介	64
2.5.4.2 三角分解	64
2.5.4.2.1 引言	64
2.5.4.2.2 高斯消元过程	65
2.5.4.2.2.1 一般消元过程	65
2.5.4.2.2.2 主元素选取	66
2.5.4.2.2.3 变元求解	67
2.5.4.2.2.4 LU 分解与 LDU 分解	68
2.5.4.2.2.4.1 概念与性质	68
2.5.4.2.2.4.2 计算方法	69
2.5.4.2.2.4.3 高斯消元法	69
2.5.4.2.2.4.4 变元求解法	71
2.5.4.2.2.4.5 代码实现	71
2.5.4.2.2.5 其它三角分解	72
2.5.4.2.2.5.1 Crout 分解	72
2.5.4.2.2.5.2 求解方法	72
2.5.4.2.2.5.3 Doolittle 分解	72
2.5.4.2.2.5.4 求解方法	73
2.5.4.2.2.5.5 Cholesky 分解	73
2.5.4.2.2.5.6 求解方法	73
2.5.4.2.2.5.7 代码实现	74
2.5.4.2.3 满秩分解	75
2.5.4.2.3.1 什么是满秩分解	75
2.5.4.2.3.2 求解方法	75
2.5.4.2.3.2.1 初等行变换法	75
2.5.4.2.4 正交三角分解	76
2.5.4.2.4.1 什么是正交三角分解	76
2.5.4.2.4.2 求解方法	77
2.5.4.2.4.2.1 Schmidt 正交化方法	77
2.5.4.2.4.2.2 Givens 变换方法	78

2.5.4.4.2.3	Householder 变换方法	79
2.5.4.4.2.4	代码实现	80
2.5.4.5	奇异值分解	81
2.5.4.5.1	概念	81
2.5.4.5.1.1	矩阵的奇异值	81
2.5.4.5.1.2	奇异值分解	81
2.5.4.5.2	求解方法	82
2.5.4.5.2.1	实例	82
2.5.4.5.2.2	代码实现	83
2.5.4.5.3	奇异值分解应用	85
2.5.4.5.3.1	SVD 与谱分解	85
2.5.5	特征值估计	85
2.5.5.1	特征值估计	85
2.5.5.1.1	特征值的界	85
2.5.5.1.2	特征值的包含区域	86
2.5.5.1.2.1	盖尔圆	86
2.5.5.1.2.2	什么是盖尔圆	86
2.5.5.1.2.3	定理与结论	86
2.5.5.1.2.4	特征值的隔离问题	86
2.5.5.1.2.5	应用	90
2.5.5.2	特征值的极性	90
2.5.5.2.1	概念与内涵	90
2.5.6	广义逆矩阵	90
2.5.6.1	投影矩阵	90
2.5.6.1.1	投影算子与投影矩阵	90
2.5.6.1.1.1	什么是投影	90
2.5.6.1.1.2	投影算子与投影矩阵	90
2.5.6.1.1.3	线性投影算子	91
2.5.6.1.1.4	正交投影算子与正交投影矩阵	91
2.5.6.1.1.5	性质	91
2.5.6.1.1.6	正交投影阵的求解	91
2.5.6.2	广义逆的定义与性质	91
2.5.6.2.1	Moore-Penrose 逆	91
2.5.6.2.1.1	概念	91
2.5.6.2.1.2	结论与性质	92
2.5.6.3	广义逆计算	92
2.5.6.3.1	Hermite 标准形计算 {1}-逆和 {1,2}-逆	92
2.5.6.3.2	满秩分解法	92
2.5.6.4	广义逆矩阵与线性方程组	93
2.5.6.4.1	基本概念	93
2.5.6.4.2	几种常见解	93
2.5.7	杂项	93
2.5.7.1	常用总结	93
2.5.7.1.1	顺序主子式	93
2.5.7.1.2	伴随矩阵	94

2.5.7.1.3	矩阵的逆	94
2.5.7.1.3.1	性质	94
2.5.7.1.3.2	特殊矩阵的逆	95
2.5.7.1.3.3	二阶方阵的逆	95
2.5.7.1.3.4	三角矩阵的逆	95
2.5.7.1.3.5	求解方法	95
2.5.7.1.3.6	伴随矩阵求逆	95
2.5.7.1.3.7	初等行变换求逆	96
2.5.7.1.3.8	Sherman-Morrison-Woodbury 公式	96
2.5.7.1.4	矩阵的秩	96
2.5.7.1.4.1	性质	96
2.5.7.1.5	矩阵的迹行列式特征值	97
2.5.7.2	初等变换	97
2.5.7.2.1	初等行变换	97
2.5.7.2.2	初等列变换	97
2.5.7.2.3	特殊初等变换	97
2.5.7.3	常见矩阵概念	98
2.5.7.3.1	奇异非奇异	98
2.5.7.3.2	特殊矩阵	98
2.5.7.3.2.1	概念汇总	98
2.5.7.3.2.2	结论	98
2.5.7.4	对角矩阵	99
2.5.7.5	矩阵运算	99
2.5.7.5.1	乘法	99
2.5.7.5.1.1	逐元素积	99
2.5.7.5.1.2	点积/内积	99
2.5.7.5.1.3	Kronecker 积	99
2.5.7.5.1.4	卷积	99
2.5.7.5.1.5	示例	100
2.5.8	名词术语	101
2.6	概率与统计	104
2.6.1	相关分析	104
2.6.1.1	相关的概念	104
2.6.1.1.1	互相关	104
2.6.1.1.1.1	互相关函数	105
2.6.1.1.1.2	互相关矩阵	105
2.6.1.1.1.3	性质	105
2.6.1.1.1.4	互协方差矩阵	105
2.6.1.1.1.5	互相关与互协方差矩阵间关系	106
2.6.1.1.2	自相关	106
2.6.1.1.2.1	自相关矩阵	106
2.6.1.1.2.2	自协方差矩阵	106
2.6.1.1.2.3	自相关与自协方差矩阵间关系	106
2.6.2	回归分析	107
2.6.2.1	回归分析简介	107

2.6.2.1.1	什么是回归分析	107
2.6.2.1.2	回归分析的类型	107
2.6.2.2	回归建模	108
2.6.2.2.1	回归分析简介	108
2.6.2.2.1.1	什么是回归分析	108
2.6.2.2.2	线性回归	108
2.6.2.2.2.1	什么是非线性回归	108
2.6.2.2.2.2	多项式线性回归	109
2.6.2.2.3	非线性回归	109
2.6.2.2.3.1	什么是非线性回归	109
2.6.2.2.4	投影追踪回归	109
2.6.2.2.4.1	什么是投影追踪回归	109
2.6.2.2.4.2	实验与分析	110
2.6.2.2.4.3	回归实验	110
2.6.2.2.4.4	实验代码	110
2.6.2.2.4.5	实验结果	110
2.6.2.2.4.6	分类实验	112
2.6.2.2.4.7	实验代码	112
2.6.2.2.4.8	实验结果	112
2.6.2.2.5	贝叶斯回归	112
2.6.2.2.5.1	什么是贝叶斯回归	112
2.6.2.3	估计方法	112
2.6.2.3.1	简介	112
2.6.2.3.2	岭回归	112
2.6.2.3.3	LASSO	112
2.6.2.3.3.1	什么是 LASSO	112
2.6.2.3.4	基追踪	113
2.6.2.3.4.1	什么是基追踪	113
2.6.2.3.5	投影追踪	113
2.6.2.3.5.1	什么是投影追踪	113
2.6.2.3.5.2	实验与分析	113
2.6.3	参考文献	113
2.6.4	名词术语	113
2.7	随机过程	115
2.7.1	概率论基础	115
2.7.1.1	概率空间与随机变量	115
2.7.1.1.1	柯尔莫哥洛夫概率公理化体系	115
2.7.1.1.1.1	是什么	115
2.7.1.1.1.2	为什么	115
2.7.1.1.1.3	怎么办	115
2.7.1.1.1.4	还有什么	116
2.7.1.1.2	傅立叶变换	118
2.7.2	随机过程基础	118
2.7.2.1	What is?	118
2.7.2.1.1	ddddddd	118

2.7.2.1.1.1	ssssssssssss	118
2.7.2.2	Why?	118
2.7.2.2.1	ddddd	118
2.7.2.2.1.1	ssssssssssss	118
2.7.3	布朗运动	118
2.7.3.1	What is?	118
2.7.3.1.1	ddddd	118
2.7.3.1.1.1	ssssssssssss	118
2.7.3.2	Why?	118
2.7.3.2.1	ddddd	118
2.7.3.2.1.1	ssssssssssss	118
2.7.4	跳跃随机过程	118
2.7.4.1	What is?	118
2.7.4.1.1	ddddd	118
2.7.4.1.1.1	ssssssssssss	118
2.7.4.2	Why?	118
2.7.4.2.1	ddddd	118
2.7.4.2.1.1	ssssssssssss	118
2.7.5	平稳过程	118
2.7.5.1	What is?	118
2.7.5.1.1	ddddd	118
2.7.5.1.1.1	ssssssssssss	118
2.7.5.2	Why?	118
2.7.5.2.1	ddddd	118
2.7.5.2.1.1	ssssssssssss	118
2.7.6	马尔可夫链	118
2.7.7	名词术语	118
2.8	拓扑学	119
2.8.1	名词术语	119
2.9	泛函分析	119
2.9.1	线性算子	119
2.9.1.1	卷积算子	119
2.9.1.1.1	普通卷积	119
2.9.1.1.1.1	一维卷积	119
2.9.1.1.1.2	二维卷积	120
2.9.1.1.2	回旋卷积	120
2.9.1.1.2.1	连续形式	120
2.9.1.1.2.2	离散形式	120
2.9.1.1.3	实验分析	121
2.9.1.1.3.1	二维卷积	121
2.9.1.2	相关算子	121
2.9.1.2.1	互相关	121
2.9.1.2.1.1	确定信号的互相关	121
2.9.1.2.1.2	不确定信号的互相关	121
2.9.1.2.1.3	周期信号的互相关	121

2.9.1.2.2	自相关	121
2.9.1.3	卷积与相关	121
2.9.1.3.1	卷积与相关关系	121
2.9.2	名词术语	123
2.10	模糊数学	123
2.10.1	引言	123
2.10.2	模糊集合	123
2.10.2.1	模糊集合	123
2.10.2.1.1	引言(不确定集合)	123
2.10.2.1.2	模糊集的定义	124
2.10.2.1.3	模糊集与明确集	124
2.10.2.1.4	模糊集的表示	124
2.10.2.1.4.1	Zadeh 表示法	125
2.10.2.1.4.2	序偶表示法	125
2.10.2.1.4.3	向量表示法	125
2.10.2.1.5	特殊模糊集	125
2.10.2.1.6	模糊集的运算	125
2.10.2.1.6.1	模糊算子	125
2.10.2.1.6.2	模糊集间的运算定律	126
2.10.2.2	隶属函数	126
2.10.2.2.1	什么是隶属函数	126
2.10.2.2.2	常见隶属函数	127
2.10.2.2.2.1	钟形曲线	127
2.10.2.2.2.2	高斯函数	129
2.10.2.3	模糊逻辑	130
2.10.2.3.1	布尔逻辑与模糊逻辑	130
2.10.2.3.2	布尔逻辑与模糊逻辑算子	130
2.10.2.3.3	模糊规则	130
2.10.2.3.3.1	IF-THEN	130
2.10.2.3.4	模糊逻辑推理步骤	130
2.10.2.4	模糊子代数	131
2.10.2.5	模糊集合理论拓展	131
2.10.2.5.1	Lattice Valued Fuzzy Sets	131
2.10.2.5.2	Intuitionistic Fuzzy Sets	131
2.10.2.5.3	Interval Type II Fuzzy Sets	131
2.10.2.5.4	Fuzzy Sets of Type 2	131
2.10.3	模糊算术	131
2.10.3.1	模糊数	131
2.10.3.2	模糊算术	131
2.10.3.3	模糊分析	131
2.10.3.4	模糊差分方程	131
2.10.4	模糊关系	131
2.10.4.1	模糊关系	131
2.10.4.1.1	明确关系与模糊关系	131
2.10.4.1.2	模糊矩阵	132

2.10.4.1.3 模糊关系的性质	132
2.10.4.2 模糊关系合成	132
2.10.5 模糊推理	132
2.10.5.1 语言变量	132
2.10.5.2 模糊规则	133
2.10.5.3 模糊推理	133
2.10.6 模糊系统	133
2.10.6.1 单入单出模糊系统	133
2.10.6.2 Takagi-Sugeno 模糊系统	134
2.10.6.2.1 Takagi-Sugeno 模糊系统	134
2.10.6.2.2 Takagi-Sugeno 模糊系统的近似特性	135
2.10.6.3 Takagi-Sugeno-Kang 模糊系统	135
2.10.6.3.1 TSKFIS 原理	135
2.10.7 模糊变换	135
2.10.8 参考文献	135
2.10.9 名词术语	135
2.11 优化理论	136
2.11.1 优化问题概述	136
2.11.1.1 优化中的对偶问题	136
2.11.1.1.1 约束优化的对偶问题	136
2.11.2 无约束优化方法	137
2.11.3 约束优化方法	137
2.11.3.1 约束优化方法概述	137
2.11.3.1.1 一般方法	137
2.11.3.2 对偶上升算法	137
2.11.3.2.1 概念与内涵	137
2.11.3.2.2 实验与分析	137
2.11.3.3 KKT 条件	137
2.11.3.3.1 概念与内涵	138
2.11.3.4 拉格朗日乘子法	138
2.11.3.4.1 概念与内涵	138
2.11.3.4.2 拉格朗日函数的对偶问题	139
2.11.3.4.3 约束优化与流形学习	139
2.11.3.4.4 实验与分析	139
2.11.3.5 增广拉格朗日法	139
2.11.3.5.1 概念与内涵	139
2.11.3.5.2 实验与分析	140
2.11.3.6 迭代收缩阈值类算法	140
2.11.3.6.1 概念与内涵	140
2.11.3.6.2 ISTA	140
2.11.3.6.3 FISTA	140
2.11.3.6.4 实例分析	140
2.11.3.6.4.1 图像去模糊与去噪	140
2.11.3.7 交替方向乘子法	140
2.11.3.7.1 ADMM 原理	141

2.11.3.7.2 Scale 版 ADMM 原理	141
2.11.3.7.3 收敛性分析	142
2.11.3.7.4 最优条件与停止准则	142
2.11.3.7.5 实验与分析	142
2.11.3.8 近端算法	142
2.11.3.8.1 什么是近端算法	142
2.11.3.8.1.1 近端算子	142
2.11.3.8.1.2 近端梯度	142
2.11.3.9 分裂增广拉格朗日收缩算法	142
2.11.3.9.1 变量分裂	142
2.11.3.9.2 SALSA 与 C-SALSA	142
2.11.4 微分方法	142
2.11.4.1 简介	142
2.11.4.2 数值微分法	142
2.11.4.3 符号微分法	143
2.11.4.4 自动微分法	143
2.11.5 参考文献	143
2.11.6 名词术语	143
2.12 名词术语	143
3 第二卷物理学基础 147	
3.1 引言	147
3.2 经典力学	148
3.2.1 牛顿力学	148
3.2.2 天体力学	148
3.2.3 天体动力学	148
3.2.4 固体力学	148
3.2.5 流体力学	148
3.2.6 工程力学	148
3.2.7 生物力学	148
3.2.8 纳米力学	148
3.3 电磁学	148
3.3.1 电与磁	148
3.3.1.1 电与电场	148
3.3.1.1.1 什么是电	148
3.3.1.2 磁与磁场	148
3.3.1.2.1 什么是磁	148
3.3.1.2.2 自旋磁矩与磁力矩	148
3.3.1.2.3 磁化	149
3.4 热力学	149
3.5 量子力学	149
3.6 量子光学	149
3.6.1 量子光学成像	149
3.6.1.1 简介	149
3.6.1.2 深度学习与量子成像	150

3.6.2 名词术语	150
3.7 数学解释	150
3.7.1 参考文献	150
3.8 名词术语	150
4 第四卷计算机学	151
4.1 操作系统	151
4.1.1 Linux 操作系统	151
4.1.1.1 常用命令	151
4.1.1.1.1 查找相关	151
4.1.1.1.1.1 查找文件	151
4.1.1.1.1.2 定位文件	151
4.1.1.1.2 更新源	151
4.1.1.2 常用技巧	152
4.1.1.2.1 第三方工具	152
4.1.1.2.1.1 多线程下载	152
4.1.2 Windows 操作系统	154
4.2 程序设计	154
4.2.1 C 程序设计	154
4.2.1.1 引言	154
4.2.1.1.1 什么是 C 语言	154
4.2.1.2 图形交互界面开发	154
4.2.1.2.1 简介	154
4.2.2 Lua 程序设计	154
4.2.2.1 Lua 笔记	154
4.2.2.1.1 Lua 相关资料	154
4.2.2.1.1.1 基础	154
4.2.2.1.2 从这里开始	154
4.2.2.1.2.1 类 Unix 系统	155
4.2.2.1.2.2 Windows	155
4.2.2.1.2.3 方法一	155
4.2.2.1.2.4 方法 2	157
4.2.2.1.3 语法	161
4.2.2.1.3.1 多重赋值	161
4.2.2.1.3.2 协同	163
4.2.2.1.4 GUI 开发	163
4.2.2.2 wxLua 简明教程	163
4.2.3 IDL 程序设计	163
4.2.3.1 引言	163
4.2.3.1.1 什么是 IDL 语言	163
4.2.3.1.2 基于 IDL 语言的库与软件	164
4.2.4 Lisp 程序设计	164
4.2.4.1 引言	164
4.2.4.1.1 什么是 Lisp 语言	164
4.2.4.1.2 开发环境配置	165

4.2.4.1.2.1	安装 Common Lisp 实现	165
4.2.4.1.2.2	安装 SBCL 实现	165
4.2.4.1.2.3	安装 CLISP 实现	165
4.2.4.1.2.4	开发环境配置	168
4.2.4.1.2.5	Emacs + Slime 环境	168
4.2.4.1.2.6	Sublime 环境	168
4.2.4.1.3	Lisp 编程入门	168
4.2.4.1.3.1	交互式界面使用	168
4.2.4.1.3.2	第一个 Lisp 程序	168
4.2.4.1.4	包库管理	169
4.2.4.1.4.1	quicklisp	169
4.2.4.1.4.2	安装	169
4.2.4.1.4.3	设置默认加载	170
4.2.4.1.4.4	使用	170
4.2.4.1.5	生成可独立执行文件	172
4.2.4.2	基础语法	172
4.2.5	Python 程序设计	172
4.2.5.1	简介	172
4.2.5.1.1	资源汇总	172
4.2.5.2	Anaconda 简明教程	172
4.2.5.2.1	安装第三方包	172
4.2.5.2.1.1	本地文件安装	172
4.2.5.2.2	安装包测试	173
4.2.6	Matlab 程序设计	173
4.2.6.1	引言	173
4.2.6.1.1	什么是 Matlab 语言	173
4.2.7	Julia 程序设计	173
4.2.7.1	引言	173
4.2.7.1.1	什么是 Julia 语言	173
4.2.7.1.2	开发环境配置	173
4.2.7.1.2.1	安装 Julia 实现	173
4.2.7.1.2.2	开发环境配置	175
4.2.7.1.2.3	Sublime 环境	175
4.2.7.1.3	Julia 编程入门	175
4.2.7.1.3.1	交互式界面使用	175
4.2.7.1.3.2	第一个 Julia 程序	175
4.2.7.1.4	包库管理	176
4.2.7.1.4.1	quicklisp	176
4.2.7.1.4.2	安装	176
4.2.7.1.4.3	设置默认加载	176
4.2.7.1.4.4	使用	177
4.2.7.1.5	生成可独立执行文件	179
4.2.8	PyTorch 简明教程	179
4.2.8.1	从这里开始	179
4.2.8.1.1	安装	179

4.2.8.1.1.1	在线安装	179
4.2.8.1.1.2	本地安装包安装	179
4.2.8.1.1.3	Anaconda	179
4.2.8.1.1.4	Python	180
4.2.8.1.1.5	本地源码安装	180
4.2.8.1.1.6	Pytorch	180
4.2.8.1.1.7	Torchvision	180
4.2.8.1.2	构建本地文档	180
4.2.8.1.2.1	API 手册	180
4.2.8.1.2.2	Pytorch	180
4.2.8.1.2.3	Torchvision	181
4.2.8.1.3	问题与解决	181
4.2.8.1.3.1	安装	181
4.2.8.1.3.2	PyQt	181
4.2.8.1.3.3	使用	181
4.2.8.1.3.4	Torch 导入错误 1	181
4.2.8.1.3.5	Torch 导入错误 2	182
4.2.8.1.3.6	Torchvision 导入失败	182
4.2.8.2	数据集制作与加载	182
4.2.8.2.1	DataSet	182
4.2.8.2.1.1	使用 Dataset	182
4.2.8.2.1.2	使用 TensorDataset	183
4.2.8.2.2	DataLoader	184
4.2.9	Shell 程序设计	184
4.2.9.1	bash 命令	184
4.2.9.1.1	常用命令	184
4.2.9.1.1.1	tee	184
4.2.9.2	ssh 协议	184
4.2.9.2.1	常用命令	184
4.3	算法设计	185
4.3.1	区域填充	185
4.3.1.1	圆形区域填充	185
4.3.1.1.1	问题分析	185
4.3.1.1.2	填充算法	185
4.3.1.1.2.1	圆面方程	185
4.3.1.2	凸多边形区域填充	185
4.3.1.2.1	问题分析	185
4.3.1.2.2	填充算法	185
4.3.1.2.2.1	扫描线法	185
4.3.1.2.2.2	夹角和法	185
4.3.1.2.2.3	多边形方程法	185
4.3.1.2.3	实验与分析	186
4.3.1.2.3.1	实验 1	186
4.3.1.2.3.2	实验设置	186
4.3.1.2.3.3	实验代码	186

4.3.1.2.3.4	实验结果	189
4.3.1.3	凹多边形区域填充	189
4.3.1.3.1	问题分析	189
4.4	互联部分	191
4.4.1	网站搭建	191
4.5	人工智能处理器	191
4.5.1	基础篇	191
4.5.1.1	引言	191
4.5.1.1.1	什么是人工智能芯片	191
4.5.1.1.2	AI 芯片分类	191
4.5.1.1.3	AI 芯片选型	192
4.5.1.2	实战篇	192
4.5.1.2.1	Atlas200DK 简明教程	192
4.5.1.2.1.1	简介	192
4.5.1.2.1.1.1	硬件配置	192
4.5.1.2.1.1.2	软件配置	193
4.5.1.2.1.1.3	资料汇总	193
4.5.1.2.1.2	环境配置	195
4.5.1.2.1.2.1	准备	195
4.5.1.2.1.2.2	编译依赖	195
4.5.1.2.1.2.3	Java	195
4.5.1.2.1.2.4	MindSpore Studio 安装	195
4.5.1.2.1.2.5	准备工作	196
4.5.1.2.1.2.6	执行安装	196
4.5.1.2.1.3	系统启动	200
4.5.1.2.1.3.1	SD 卡启动镜像制作	200
4.5.1.2.1.3.2	准备工作	200
4.5.1.2.1.3.3	依赖安装	201
4.5.1.2.1.3.4	配置 Atlas200DK 开发板 IP 地址	201
4.5.1.2.1.3.5	制作 SD 卡启动镜像	202
4.5.1.2.1.3.6	连接开发板与上位机	202
4.5.1.2.1.3.7	登录开发板系统	203
4.5.1.2.1.4	MindSpore Studio 开发	204
4.5.1.2.1.4.1	开发概览	204
4.5.1.2.1.4.2	添加与管理开发板设备	205
4.5.1.2.1.4.3	自定义算子	206
4.5.1.2.1.5	深度学习移植实例	206
4.5.1.2.1.5.1	简单例子	206
4.5.1.2.1.6	PyTorch 转 Caffe	206
4.6	嵌入式系统开发	206
5	第五卷信号处理	207
5.1	模拟信号处理	207
5.1.1	名词术语	207
5.2	数字信号处理	208

5.2.1	引言	208
5.2.2	基础内容	208
5.2.2.1	简介	208
5.2.2.2	插值算法	208
5.2.2.2.1	Sinc 插值	208
5.2.2.2.1.1	Sinc 插值原理	208
5.2.2.2.1.2	实验与分析	208
5.2.2.2.1.3	sinc 函数特性	208
5.2.2.2.1.4	实验代码	208
5.2.2.2.1.5	实验结果	209
5.2.2.2.1.6	sinc 插值	210
5.2.2.2.1.7	实验代码	210
5.2.2.2.1.8	实验结果	211
5.2.3	谱估计理论	211
5.2.3.1	简介	211
5.2.3.1.1	What?	211
5.2.3.1.2	Why?	211
5.2.3.1.3	How?	211
5.2.3.2	经典谱估计	211
5.2.3.2.1	经典谱估计	211
5.2.3.2.1.1	谱估计	211
5.2.3.2.1.2	经典谱估计	213
5.2.3.2.2	窗函数及其性质	213
5.2.3.2.2.1	概念	213
5.2.3.2.2.2	类型	213
5.2.3.2.2.3	hamming	213
5.2.3.2.2.4	cosine	213
5.2.3.2.2.5	blackman	213
5.2.3.2.2.6	gaussian	213
5.2.3.2.2.7	tukey	213
5.2.3.2.2.8	kaiser	213
5.2.3.2.2.9	实例	213
5.2.3.2.2.10	窗函数比较	213
5.2.3.2.2.11	实验内容	213
5.2.3.2.2.12	实验代码	214
5.2.3.2.2.13	实验结果	222
5.2.3.2.3	傅立叶变换	222
5.2.3.2.3.1	概念与内涵	222
5.2.3.2.3.2	实验分析	222
5.2.3.2.3.3	仿真数据实验	222
5.2.3.2.3.4	实验代码	222
5.2.3.2.3.5	实验结果	222
5.2.3.2.3.6	真实数据实验	222
5.2.3.2.4	傅立叶变换与卷积	222
5.2.3.2.4.1	概念与内涵	222

5.2.3.2.4.2	FFT 实现卷积	222
5.2.3.2.4.3	实验与分析	225
5.2.3.2.4.4	FFT 实现二维矩阵卷积	225
5.2.3.2.4.5	FFT 实现二维图像卷积	226
5.2.3.3	现代谱估计	227
5.2.3.3.1	现代谱估计	227
5.2.3.3.1.1	现代谱估计	227
5.2.3.3.1.2	概念与内涵	227
5.2.3.3.1.3	现代普估计方法	227
5.2.3.3.2	基于滤波器的谱估计	228
5.2.3.3.2.1	简介	228
5.2.3.3.3	基于信号子空间的谱估计	228
5.2.3.3.3.1	简介	228
5.2.3.3.4	基于噪声子空间的谱估计	228
5.2.3.3.4.1	多重信号分类	228
5.2.3.3.4.2	多信号分类用于频率估计	229
5.2.3.3.4.3	实例分析	229
5.2.3.3.4.4	仿真信号	229
5.2.3.3.4.5	实验内容	229
5.2.3.3.4.6	实验代码	229
5.2.3.3.4.7	实验结果	229
5.2.3.4	智能谱估计	231
5.2.3.4.1	智能谱估计简介	231
5.2.3.4.1.1	什么是智能谱估计	231
5.2.4	时频分析	231
5.2.4.1	简介	231
5.2.4.1.1	What?	231
5.2.4.1.2	Why?	231
5.2.4.1.3	How?	231
5.2.4.2	经典时频分析	231
5.2.4.2.1	经典时频分析	231
5.2.4.2.1.1	经典时频分析	231
5.2.4.2.1.2	概念与内涵	231
5.2.4.2.1.3	经典时频分析方法	231
5.2.4.2.2	短时傅立叶变换	231
5.2.4.2.2.1	短时傅立叶变换	231
5.2.4.2.2.2	连续时间 STFT	231
5.2.4.2.2.3	离散时间 STFT	232
5.2.4.2.2.4	频谱表示	232
5.2.4.2.2.5	逆短时傅立叶变换	232
5.2.4.2.2.6	连续时间逆 STFT	232
5.2.4.2.2.7	离散时间逆 STFT	233
5.2.4.2.2.8	实现步骤	233
5.2.4.2.2.9	实例分析	233
5.2.4.2.2.10	实验 1: 仿真信号	233

5.2.4.2.2.11	实验内容	233
5.2.4.2.2.12	实验代码	233
5.2.4.2.2.13	实验结果	234
5.2.4.2.2.14	实验分析	237
5.2.4.2.2.15	实验 2: 真实信号	242
5.2.4.2.3	小波变换	242
5.2.4.2.3.1	小波与小波变换	242
5.2.4.2.3.2	小波及其性质	242
5.2.4.2.3.3	小波变换	242
5.2.4.2.3.4	尺度与频率的关系	242
5.2.4.2.3.5	连续小波变换	243
5.2.4.2.3.6	连续小波种类	243
5.2.4.2.3.7	Mexh 小波	243
5.2.4.2.3.8	Morlet 小波	244
5.2.4.2.3.9	算法步骤	244
5.2.4.2.3.10	离散小波变换	244
5.2.4.2.3.11	离散小波种类	244
5.2.4.2.3.12	Haar 小波	245
5.2.4.2.3.13	连续逆小波变换	245
5.2.4.2.3.14	实例分析	245
5.2.4.2.3.15	仿真信号	245
5.2.4.2.3.16	实验内容	245
5.2.4.2.3.17	实验代码	246
5.2.4.2.3.18	实验结果	246
5.2.4.2.3.19	真实信号	248
5.2.4.2.4	S 变换	248
5.2.4.3	总结	248
5.2.4.3.1	不同时频变换间的关系	248
5.2.5	名词术语	248
5.3	非线性信号处理	249
5.4	统计信号处理	249
5.4.1	引言	249
5.4.2	估计理论	250
5.4.2.1	简介	250
5.4.3	滤波理论	250
5.4.4	名词术语	250
5.5	稀疏信号处理	250
5.5.1	引言	250
5.5.1.1	资源库	250
5.5.2	稀疏采样与重构	250
5.5.2.1	稀疏性分析	250
5.5.2.1.1	稀疏信号	250
5.5.2.2	采样	251
5.5.2.3	重构	251
5.5.2.3.1	有限等距性	251

5.5.3 表示字典	251
5.5.3.1 字典的类型	251
5.5.3.1.1 什么是字典	251
5.5.3.1.2 字典的种类	251
5.5.3.1.2.1 完备与过完备	251
5.5.3.1.2.2 冗余与非冗余	252
5.5.3.1.2.3 正交与非正交	252
5.5.3.1.2.4 总结	252
5.5.3.2 离散余弦变换类字典	252
5.5.3.2.1 什么是离散余弦变换	252
5.5.3.2.2 一维离散余弦变换	252
5.5.3.2.3 多维离散余弦变换	253
5.5.3.2.4 过完备 DCT 字典	254
5.5.3.2.4.1 一维过完备 DCT 字典	254
5.5.3.2.4.2 多维过完备 DCT 字典	254
5.5.3.2.5 实验与分析	255
5.5.3.2.5.1 一维离散余弦变换	255
5.5.3.2.5.2 原始定义	255
5.5.3.2.5.3 矩阵法实现	256
5.5.3.2.5.4 图像的 1 维 DCT 变换	256
5.5.3.2.5.5 二维离散余弦变换	256
5.5.3.2.5.6 仿真数据	256
5.5.3.2.5.7 真实图像数据	258
5.5.3.2.5.8 DCT 过完备字典	258
5.5.4 信号稀疏分解	261
5.5.4.1 简介	261
5.5.4.1.1 什么是信号稀疏分解	261
5.5.4.1.2 常见稀疏分解算法	261
5.5.4.2 匹配追踪	261
5.5.4.2.1 什么是匹配追踪	261
5.5.4.2.2 匹配追踪算法	262
5.5.4.2.3 实验与分析	262
5.5.4.3 正交匹配追踪	262
5.5.4.3.1 什么是正交匹配追踪	262
5.5.4.3.2 正交匹配追踪算法	262
5.5.4.3.3 实验与分析	263
5.5.4.3.3.1 仿真数据实验	264
5.5.4.3.3.2 真实数据实验	271
5.5.5 线性压缩感知	271
5.5.5.1 简介	271
5.5.5.2 线性压缩感知	271
5.5.5.2.1 线性压缩感知概述	271
5.5.5.2.2 一维线性压缩感知	272
5.5.5.2.2.1 稀疏信号的压缩采样与恢复	272
5.5.5.2.2.2 非稀疏信号的压缩采样与恢复	272

5.5.5.2.2.3	第一种方式	272
5.5.5.2.2.4	第二种方式	272
5.5.5.2.2.5	二维压缩感知	273
5.5.5.2.3	复压缩感知	273
5.5.5.2.4	压缩感知与流形	273
5.5.5.2.5	压缩感知分析实验	273
5.5.5.3	线性压缩感知信号模型	278
5.5.5.4	线性压缩感知观测	278
5.5.5.4.1	感知矩阵的设计	278
5.5.5.4.1.1	零空间条件	278
5.5.5.4.1.2	有限等距性质	278
5.5.5.4.1.3	相干性/一致性条件	279
5.5.5.4.1.4	稳定性	279
5.5.5.4.1.5	讨论	280
5.5.5.4.2	感知矩阵的构造	280
5.5.5.4.3	常见感知矩阵	280
5.5.5.5	线性压缩感知重构	280
5.5.5.5.1	重构算法	280
5.5.5.5.1.1	OMP	280
5.5.5.6	线性压缩感知实践	281
5.5.5.6.1	稀疏信号实验	281
5.5.5.6.1.1	一维仿真信号	281
5.5.5.6.1.2	二维图像信号	287
5.5.5.6.1.3	实验设置	287
5.5.5.6.1.4	实验代码	288
5.5.5.6.1.5	实验结果	288
5.5.5.6.2	非稀疏信号 CS	292
5.5.5.6.2.1	压缩感知方式实验	292
5.5.5.6.2.2	一维压缩感知实验	295
5.5.5.6.2.3	无稀疏表示	295
5.5.5.6.2.4	含稀疏表示	296
5.5.5.6.2.5	二维压缩感知实验 1	307
5.5.5.6.2.6	二维压缩感知实验 2	307
5.5.5.6.3	复压缩感知实验	307
5.5.5.6.3.1	核磁共振成像	307
5.5.5.6.3.2	实验代码	307
5.5.5.6.3.3	实验结果	307
5.5.6	学习式压缩感知	307
5.5.6.1	引言	307
5.5.6.1.1	Resources	307
5.5.7	学习式压缩感知	310
5.5.7.1	学习式压缩感知	310
5.5.8	深度学习与压缩感知	310
5.5.8.1	基于 GAN 的压缩感知	310
5.5.8.2	基于 ISTA 的压缩感知网络	310

5.5.8.3	基于 ADMM 的压缩感知网络	310
5.5.8.4	基于 SALSA 的压缩感知网络	310
5.5.8.5	深度压缩感知	310
5.5.9	参考文献	310
5.5.10	名词术语	310
5.6	数字图像处理	311
5.6.1	图像压缩	311
5.6.2	图像增强	311
5.6.2.1	图像模糊	311
5.6.2.1.1	模糊算子	311
5.6.2.2	直方图均衡化	311
6	第六卷人工智能	313
6.1	简介	313
6.2	优化方法	313
6.2.1	简介	313
6.2.2	名词术语	313
6.3	机器学习	314
6.3.1	简介	314
6.3.1.1	What is?	314
6.3.1.1.1	ddddd	314
6.3.1.1.1.1	ssssssssss	314
6.3.1.2	Why?	314
6.3.1.2.1	ddddd	314
6.3.1.2.1.1	ssssssssss	314
6.3.1.3	How?	314
6.3.1.3.1	materials	314
6.3.1.3.1.1	books	314
6.3.2	监督学习	314
6.3.2.1	监督学习实践 1 - 回归与优化	314
6.3.2.1.1	线性回归	314
6.3.2.1.1.1	线性回归预测房价	314
6.3.2.1.2	Logistic Regression	316
6.3.2.1.2.1	Logistic Regression 手写体分类	316
6.3.2.1.3	Vectorization	317
6.3.2.1.4	Debugging: Gradient Checking	318
6.3.2.1.5	Softmax Regression	319
6.3.3	无监督学习	320
6.3.3.1	无监督学习实践 1 - PCA 与白化	320
6.3.3.1.1	主成分分析 (PCA)	320
6.3.3.1.2	白化 (Whitening)	320
6.3.3.1.3	PCA/Whitening 计算步骤	320
6.3.3.1.4	二维数据 PCA 实验	321
6.3.3.1.5	PCA 与白化图像数据实验	327
6.3.3.1.5.1	PCA 实验	327

6.3.3.1.5.2	PCA 白化实验	327
6.3.4	名词术语	330
6.4	神经网络	333
6.4.1	简介	333
6.4.1.1	What is?	333
6.4.1.1.1	ddddd	333
6.4.1.1.1.1	ssssssssss	333
6.4.1.2	Why?	333
6.4.1.2.1	ddddd	333
6.4.1.2.1.1	ssssssssss	333
6.4.1.3	How?	333
6.4.1.3.1	materials	333
6.4.1.3.1.1	books	333
6.4.2	神经网络组件	333
6.4.2.1	激活函数单元	333
6.4.2.1.1	什么是激活函数	333
6.4.2.1.2	为什么要有激活函数	333
6.4.2.1.3	经典激活函数分类	333
6.4.2.1.3.1	恒等函数	335
6.4.2.1.3.2	tanh	335
6.4.2.1.3.3	Sigmoid	335
6.4.2.1.3.4	softplus	337
6.4.2.1.3.5	softsign	337
6.4.2.1.3.6	elu	337
6.4.2.1.3.7	relu	339
6.4.2.1.3.8	relu6	339
6.4.2.1.3.9	leaky relu	339
6.4.2.1.3.10	selu	341
6.4.2.1.3.11	crelu	341
6.4.2.1.3.12	Swish	342
6.4.2.1.4	新颖激活函数	343
6.4.2.2	卷积单元	343
6.4.2.2.1	经典卷积运算	343
6.4.2.2.1.1	经典二维卷积	343
6.4.2.2.1.2	经典膨胀二维卷积运算	343
6.4.2.2.2	经典二维转置卷积运算	344
6.4.2.2.3	新型卷积	344
6.4.2.2.3.1	平衡卷积	344
6.4.2.2.4	实验分析	345
6.4.2.2.4.1	卷积与相关	345
6.4.2.2.4.2	实验说明	345
6.4.2.2.4.3	实验结果	345
6.4.2.3	池化单元	347
6.4.2.3.1	经典二维池化运算	347
6.4.2.3.2	膨胀二维池化运算	347

6.4.2.3.3	金字塔池化	347
6.4.2.3.3.1	空间金字塔池化	347
6.4.2.3.3.2	金字塔场景池化	347
6.4.2.3.3.3	扩张空间金字塔池化	349
6.4.2.4	正则化单元	349
6.4.2.5	跳跃连接	349
6.4.2.5.1	前向跳跃连接	349
6.4.2.5.1.1	经典前向跳跃连接	349
6.4.2.5.1.2	结构分析	349
6.4.2.5.1.3	梯度传播分析	351
6.4.2.5.2	后向跳跃连接	352
6.4.2.5.2.1	梯度传播	352
6.4.2.6	递归单元	352
6.4.2.7	残差单元	352
6.4.2.7.1	经典残差块分析	352
6.4.2.7.1.1	残差块结构	352
6.4.2.7.1.2	梯度传播分析	352
6.4.2.8	参考文献	353
6.4.3	神经网络常用损失函数	353
6.4.3.1	基于误差的损失函数	353
6.4.3.1.1	什么是误差	353
6.4.3.1.2	均方误差损失	353
6.4.3.1.3	绝对误差损失	353
6.4.3.1.4	光滑绝对误差损失	353
6.4.3.2	基于熵的损失函数	353
6.4.3.2.1	什么是熵	353
6.4.3.2.1.1	离散变量的熵	354
6.4.3.2.2	交叉熵损失函数	354
6.4.3.2.2.1	离散变量的交叉熵	354
6.4.3.2.3	相对熵损失函数	354
6.4.3.2.3.1	离散变量的相对熵	354
6.4.3.2.4	二值交叉熵损失函数	354
6.4.4	简单神经网络	355
6.4.4.1	单层感知器	355
6.4.4.1.1	什么是感知器	355
6.4.4.1.2	单层感知器	355
6.4.4.1.3	实例	355
6.4.4.2	多层感知器	355
6.4.4.2.1	概念与内涵	355
6.4.4.2.2	实例	355
6.4.4.2.3	MNIST 手写体分类	355
6.4.4.2.4	CFAR10 物体分类	355
6.4.4.3	自编码神经网络	355
6.4.5	支撑矢量机	355
6.4.5.1	支撑矢量机	355

6.4.5.2 模糊支撑矢量机	356
6.4.5.2.1 FSVM 原理	356
6.4.5.2.2 隶属函数选择	356
6.4.5.2.2.1 基于时间特性	356
6.4.5.2.2.2 基于类别中心	357
6.4.5.2.2.3 基于训练误差	357
6.4.5.2.3 实验及结果	357
6.4.5.3 参考文献	357
6.4.5.4 名词术语	357
6.4.6 极速学习机	357
6.4.6.1 经典极速学习机	357
6.4.6.1.1 什么是极速学习机	357
6.4.6.1.1.1 实验与分析	358
6.4.6.1.1.2 实验数据	358
6.4.6.1.1.3 实验代码	358
6.4.6.1.1.4 实验结果	358
6.4.6.1.2 基于局部感受野的极速学习机	358
6.4.6.1.2.1 说明	358
6.4.6.1.2.2 摘要内容	358
6.4.6.1.2.3 引言部分	359
6.4.6.1.2.4 回顾 ELM, CNN 和 HTM	359
6.4.6.1.2.4.1 极速学习机	359
6.4.6.1.2.4.2 ELM 特征映射	361
6.4.6.1.2.4.3 ELM 特征学习	361
6.4.6.1.2.4.4 卷积神经网络	362
6.4.6.1.2.4.5 卷积	363
6.4.6.1.2.4.6 池化	363
6.4.6.1.2.4.7 层级实时记忆	363
6.4.6.1.2.5 ELM-LRF 原理	364
6.4.6.1.2.5.1 A. 全连接与局部连接 (Full and Local Connections)	364
6.4.6.1.2.5.2 B. 基于局部感受野的 ELM	365
6.4.6.1.2.5.3 C. 组合节点	365
6.4.6.1.2.6 局部感受野的实现	366
6.4.6.1.2.6.1 A. ELM-LRF 的特殊组合节点	366
6.4.6.1.2.6.2 B. 随机输入权重	366
6.4.6.1.2.6.3 C. 平方根池化 (square/square-root pooling) 结构	367
6.4.6.1.2.6.4 D. 基于输出权重的闭式解	368
6.4.6.1.2.7 讨论	368
6.4.6.1.2.7.1 A. 普适近似和分类能力	368
6.4.6.1.2.7.2 B. ELM-LRF 与 HTM 和 CNN 的关系	369
6.4.6.1.2.8 实验	369
6.4.6.1.2.8.1 A. 测试误差	370
6.4.6.1.2.8.2 B. 训练时间	370
6.4.6.1.2.8.3 C. 特征图	371
6.4.6.1.2.8.4 D. 随机输入权重的正交化	371

6.4.6.2.9	结论	371
6.4.6.2.10	代码实现	371
6.4.6.2.10.1	Source Code	371
6.4.6.2.10.2	Experimental Results	373
6.4.6.2.10.3	MNIST 数据集	373
6.4.6.2.10.4	NORB 数据集	375
6.4.6.2.10.5	论文中代码结果	375
6.4.6.2.10.6	本人实现代码结果	376
6.4.6.2.11	参考文献	384
6.4.6.3	模糊极速学习机	384
6.4.6.4	基于局部感受野的模糊极速学习机	384
6.4.6.4.1	简介	384
6.4.6.4.2	FELM-LRF 原理	385
6.4.6.4.3	隶属函数选择	385
6.4.6.4.4	实验及结果	385
6.4.6.5	在线极速学习机	385
6.4.6.5.1	简介	385
6.4.6.5.2	OSELM 原理	385
6.4.6.6	基于局部感受野的在线极速学习机	387
6.4.6.6.1	简介	387
6.4.6.6.2	OSELM-LRF 原理	387
6.4.6.7	参考文献	389
6.4.6.8	名词术语	389
6.4.7	基于能量的神经网络	390
6.4.7.1	Hopfield 神经网络	390
6.4.7.2	玻尔兹曼机与受限玻尔兹曼机	390
6.4.8	卷积神经网络	390
6.4.8.1	经典卷积神经网络	390
6.4.8.1.1	卷积神经网络举例	390
6.4.8.1.1.1	LeNet	390
6.4.8.1.1.2	AlexNet	390
6.4.8.1.1.3	VGGNet	392
6.4.8.1.1.4	GoogleNet	392
6.4.9	递归神经网络	392
6.4.9.1	递归神经网络	392
6.4.10	生成式神经网络	392
6.4.10.1	生成式神经网络	392
6.4.11	残差神经网络	392
6.4.11.1	残差神经网络	392
6.4.12	分形神经网络	392
6.4.12.1	分形神经网络	392
6.4.13	随机神经网络	393
6.4.13.1	随机神经网络	393
6.4.14	复杂度分析	393
6.4.14.1	衡量指标	393

6.4.14.1.1	运算量计算	393
6.4.14.1.1.1	卷积层	393
6.4.14.1.1.2	全连接层	393
6.4.14.2	实例分析	393
6.4.14.2.1	SSD300 目标检测网络分析	393
6.4.15	参考文献	395
6.4.16	名词术语	395
6.5	深度学习	395
6.5.1	简介	395
6.5.2	名词术语	395
6.6	模糊神经系统	396
6.6.1	简介	396
6.6.2	神经模糊	396
6.6.2.1	神经模糊基础	396
6.6.3	经典模糊神经网络	396
6.6.3.1	自适应神经模糊推理系统	396
6.6.3.2	基于 ELM 的模糊推理系统	396
6.6.3.2.1	SLFN 与 ANFIS 的等效性	396
6.6.4	参考文献	397
6.6.5	名词术语	397
6.7	课程学习	397
6.7.1	引言	397
6.7.1.1	什么是课程学习	397
6.7.2	名词术语	397
6.8	自步学习	397
6.8.1	引言	397
6.8.1.1	什么是自步学习	397
6.8.2	自步学习基础	398
6.8.3	自步函数	398
6.8.3.1	自步函数设计准则	398
6.8.3.2	常用自步函数	398
6.8.3.2.1	二值加权	398
6.8.3.2.2	线性加权	398
6.8.3.2.3	对数加权	399
6.8.3.2.4	混合加权	399
6.8.3.3	可学习自步函数	399
6.8.4	参考文献	399
6.8.5	名词术语	399
7	第七卷雷达信号处理	401
7.1	引言	401
7.1.1	雷达种类	401
7.1.2	推荐资源	401
7.2	辐射源信号分析	402
7.2.1	雷达辐射源信号	402

7.2.1.1	概述	402
7.2.1.1.1	雷达辐射源信号描述方式	402
7.2.1.1.1.1	脉冲重复间隔	403
7.2.1.2	多雷达辐射源信号模型	403
7.2.1.2.1	雷达辐射源信号描述方式	403
7.2.1.3	单雷达信号模型	404
7.2.1.3.1	发射信号	404
7.2.1.3.2	接收信号	404
7.2.1.3.3	噪声信号	404
7.2.1.3.4	杂波反射信号	404
7.2.1.3.5	合成信号	404
7.2.1.4	雷达辐射源信号类型	404
7.2.1.4.1	概述	404
7.2.2	辐射源信号仿真	404
7.2.3	辐射源信号分选	404
7.2.3.1	引言	404
7.2.3.2	传统方法	405
7.2.3.2.1	一般集合	405
7.2.4	参考文献	405
7.2.5	名词术语	405
7.3	脉冲雷达	405
7.4	连续波雷达	405
7.4.1	引言	405
7.4.2	系统概述	405
7.4.2.1	三角波泄漏	405
7.4.2.1.1	三角波泄漏分析	405
7.4.2.1.1.1	泄漏原因	405
7.4.2.1.1.2	解决方法	406
7.4.2.1.1.3	频域动态压缩方法	406
7.4.2.1.1.4	自适应处理方法	406
7.4.2.1.2	三角波泄漏自适应对消验证	406
7.4.2.1.2.1	实现方式及结果对比	406
7.4.2.1.2.2	自己实现	406
7.4.2.1.2.3	matlab 实现	408
7.4.2.1.2.4	结果对比	410
7.4.2.1.2.5	滤波效果分析	410
7.4.2.1.2.6	参考信号	410
7.4.2.1.2.7	权重计算方法	414
7.4.2.1.2.8	滤波器阶数	414
7.4.2.1.2.9	步长因子	420
7.4.2.1.3	如何使用学习好的权重	422
7.4.2.1.3.1	利用学习好的权重滤波	422
7.4.2.1.3.2	基于学习好的权重再滤波	423
7.4.3	探测	424
7.4.3.1	简介	424

7.4.3.2	距离与速度测量	424
7.4.3.2.1	三角波测距与测速	424
7.4.3.2.2	锯齿波测距	425
7.4.3.2.3	距离速度模糊	425
7.4.3.2.3.1	测速模糊	426
7.4.3.2.3.2	解决方案	427
7.4.3.2.3.3	多目标配对	427
7.4.3.3	角度测量	427
7.4.3.3.1	简介	427
7.4.3.3.2	相位差测角	427
7.4.3.3.2.1	相位差法测角原理概览	427
7.4.3.3.2.2	相位差法测角原理详解	427
7.4.3.3.2.3	相位差法测角范围	430
7.4.3.3.2.4	相位差法测角误差及多值性	430
7.4.4	成像	430
7.4.4.1	简介	430
7.4.4.2	单频连续波雷达成像	432
7.4.4.2.1	一般集合	432
7.4.4.2.1.1	概念	432
7.4.5	识别	432
7.4.6	参考文献	432
7.5	合成孔径雷达	432
7.5.1	合成孔径雷达简介	432
7.5.1.1	What is?	432
7.5.2	SAR 系统概述	433
7.5.2.1	相关概念汇总	433
7.5.2.1.1	SAR 分类	433
7.5.2.1.2	SAR 成像几何关系	433
7.5.2.1.3	符号列表	436
7.5.2.1.4	SAR 数据获取	437
7.5.2.1.4.1	采样参数的选取	437
7.5.2.1.5	SAR 成像模式	438
7.5.2.1.6	斑点噪声 (Speckle)	438
7.5.2.1.7	公式总结	439
7.5.2.1.8	术语解释	439
7.5.2.1.8.1	Foot Print	439
7.5.2.2	SAR 系统组成	439
7.5.2.2.1	硬件系统	439
7.5.2.2.2	脉冲时序与采样	439
7.5.2.2.3	几何建模	442
7.5.2.2.3.1	关键参数计算	442
7.5.2.2.3.2	卫星速度与波束掠过地面的速度	442
7.5.2.2.3.3	距离向波束宽度计算	444
7.5.2.3	线性调频信号	444
7.5.2.3.1	线性调频信号	444

7.5.2.3.2	解调后的信号	446
7.5.2.3.3	实验仿真	446
7.5.2.3.3.1	生成分析 LFM	446
7.5.2.3.3.2	实验代码	446
7.5.2.3.3.3	实验结果	448
7.5.2.4	SAR 回波信号及其性质	450
7.5.2.4.1	SAR 回波信号	450
7.5.2.4.2	回波信号解调	452
7.5.2.4.3	SAR 冲击响应	453
7.5.2.4.4	SAR 回波信号的频谱	454
7.5.2.4.4.1	距离维频谱	454
7.5.2.4.4.2	方位维频谱	454
7.5.2.4.4.3	距离多普勒域频谱	454
7.5.2.4.4.4	二维频谱	454
7.5.2.5	脉冲压缩	454
7.5.3	SAR 仿真技术	454
7.5.3.1	回波信号模型	454
7.5.3.1.1	SAR 回波信号	454
7.5.3.1.1.1	目标回波分析	454
7.5.3.1.1.2	目标回波基础理论	456
7.5.3.1.2	时域回波仿真	456
7.5.3.1.2.1	点目标	456
7.5.3.1.2.2	面目标	457
7.5.3.2	SAR 回波频谱分析	457
7.5.3.2.1	仿真数据	457
7.5.3.2.2	真实数据	457
7.5.4	SAR 成像理论	457
7.5.4.1	SAR 成像简介	457
7.5.4.1.1	不同成像方法结果对比	457
7.5.4.2	距离多普勒成像	458
7.5.4.2.1	距离多普勒方法	458
7.5.4.2.1.1	小斜视下的 RDA	458
7.5.4.2.1.2	SAR 原始数据	458
7.5.4.2.1.3	距离压缩	460
7.5.4.2.1.4	方位向傅里叶变换	461
7.5.4.2.1.5	距离徙动校正	461
7.5.4.2.1.6	方位压缩	462
7.5.4.2.1.7	多视处理	462
7.5.4.2.1.8	大斜视下的 RDA	462
7.5.4.2.1.9	二次距离压缩	463
7.5.4.2.1.10	多普勒相位补偿	463
7.5.4.2.1.11	距离徙动的改进	463
7.5.4.2.1.12	方位匹配滤波器的改进	464
7.5.4.2.2	实验与分析	464
7.5.4.2.2.1	仿真数据实验	464

7.5.4.2.2.2	实验说明	464
7.5.4.2.2.3	实验结果	464
7.5.4.2.2.4	真实数据实验	464
7.5.4.2.2.5	实验数据	464
7.5.4.2.2.6	实验说明	464
7.5.4.2.2.7	实验代码	464
7.5.4.2.2.8	实验结果	464
7.5.4.3	调频变标成像	472
7.5.4.3.1	调频变标原理	472
7.5.4.3.2	调频变标成像	472
7.5.4.3.2.1	CSA 方法概览	472
7.5.4.3.2.2	调频变标	472
7.5.4.3.2.3	距离压缩与距离徙动校正	473
7.5.4.3.2.4	方位向处理	473
7.5.4.3.3	实验与分析	473
7.5.4.3.3.1	仿真数据实验	473
7.5.4.3.3.2	实验说明	473
7.5.4.3.3.3	实验结果	474
7.5.4.3.3.4	真实数据实验	477
7.5.4.3.3.5	实验数据	477
7.5.4.3.3.6	实验说明	477
7.5.4.3.3.7	实验代码	477
7.5.4.3.3.8	实验结果	477
7.5.4.4	正则化成像方法	481
7.5.4.4.1	正则化成像	481
7.5.4.4.2	实验与分析	481
7.5.4.4.2.1	实验说明	481
7.5.4.4.2.2	实验代码	481
7.5.4.4.2.3	实验结果	483
7.5.4.5	压缩感知成像	483
7.5.4.5.1	压缩感知 SAR 成像	483
7.5.4.5.2	实验与分析	487
7.5.4.5.2.1	仿真数据	487
7.5.4.5.2.2	实验说明	487
7.5.4.5.2.3	实验代码	488
7.5.4.5.2.4	实验结果	488
7.5.4.5.2.5	真实数据	488
7.5.4.5.2.6	实验说明	488
7.5.4.5.2.7	实验代码	495
7.5.4.5.2.8	实验结果	495
7.5.5	SAR 超分辨成像	495
7.5.5.1	SAR 超分辨成像简介	495
7.5.5.1.1	不同成像方法结果对比	495
7.5.5.2	带宽外推超分辨 SAR 成像	497
7.5.5.2.1	带宽外推	497

7.5.5.3 经典谱估计超分辨 SAR 成像	497
7.5.5.3.1 经典谱估计	497
7.5.5.4 现代谱估计超分辨 SAR 成像	497
7.5.5.4.1 现代谱估计	497
7.5.5.5 正则化超分辨成像方法	498
7.5.5.5.1 实验与分析	498
7.5.5.5.1.1 实验说明	498
7.5.5.5.1.2 实验代码	499
7.5.5.5.1.3 实验结果	499
7.5.5.6 压缩感知超分辨 SAR 成像	499
7.5.5.6.1 压缩感知超分辨 SAR 成像	499
7.5.5.6.2 实验与分析	499
7.5.5.6.3 实验与分析	499
7.5.5.6.3.1 仿真数据	499
7.5.5.6.3.2 实验说明	499
7.5.5.6.3.3 实验代码	503
7.5.5.6.3.4 实验结果	503
7.5.5.6.3.5 真实数据	504
7.5.5.6.3.6 实验说明	504
7.5.5.6.3.7 实验代码	504
7.5.5.6.3.8 实验结果	504
7.5.6 深度学习与 SAR 成像	504
7.5.6.1 深度学习与 SAR 成像简介	504
7.5.6.2 学习式压缩感知成像	504
7.5.6.2.1 动机与贡献	504
7.5.6.2.2 原理与方法	511
7.5.6.2.2.1 学习式压缩感知框架	511
7.5.6.2.2.2 有限等距性约束	511
7.5.6.2.3 学习式压缩成像	511
7.5.6.2.3.1 有限等距性约束	511
7.5.6.2.4 实验与分析	511
7.5.6.3 基于 IQGAN 的生成式超分辨 SAR 成像	511
7.5.6.3.1 动机与贡献	511
7.5.6.3.2 原理与方法	512
7.5.6.3.2.1 模型框架	512
7.5.6.3.2.2 相位保持	512
7.5.6.3.2.3 超分辨成像	512
7.5.6.3.3 实验与分析	512
7.5.6.4 学习式压缩感知成像	512
7.5.6.4.1 动机与贡献	512
7.5.6.4.1.1 SAR 数据特性	512
7.5.7 SAR 图像超分辨	512
7.5.8 实验集锦	512
7.5.8.1 基础分析实验	512
7.5.8.1.1 实验说明	512

7.5.8.1.1.1	实验参数	513
7.5.8.1.2	调频方向分析	514
7.5.8.1.3	分辨率分析	514
7.5.8.1.4	距离徙动校正	521
7.5.8.2	子区域成像实验	527
7.5.8.2.1	实验说明	527
7.5.8.2.2	仿真数据实验	527
7.5.8.2.2.1	点目标数据实验	527
7.5.8.2.2.2	实验代码	527
7.5.8.2.2.3	实验结果	527
7.5.8.2.3	真实数据实验	527
7.5.8.2.3.1	RADARSAT1 数据实验	527
7.5.8.2.3.2	实验代码	530
7.5.8.2.3.3	实验结果	534
7.5.9	参考文献	534
7.5.10	名词术语	534
7.6	极化合成孔径雷达	544
7.6.1	极化 SAR 成像理论	544
7.6.2	极化 SAR 超分辨成像	544
7.6.3	极化 SAR 图像超分辨	544
7.7	共用技术	544
7.7.1	杂波抑制	544
7.7.1.1	动目标指示器	544
7.7.2	雷达散射截面积	544
7.7.2.1	基础概念	544
7.7.2.1.1	什么是雷达散射截面积	544
7.7.2.1.2	雷达散射系数的分类	545
7.7.3	名词术语	545
7.8	补充内容	545
7.8.1	雷达产品	545
7.8.1.1	简介	545
7.8.1.1.1	SAR 雷达产品概览	545
7.8.1.2	雷达数据获取	545
7.8.1.2.1	合成孔径雷达	545
7.8.1.2.1.1	Alaska Satellite Facility	545
7.8.1.2.1.2	公开数据介绍	545
7.8.1.2.1.3	公开数据检索与下载	546
7.8.1.2.1.4	公开数据格式	546
7.8.1.2.1.5	WInSAR	546
7.8.1.3	雷达产品数据格式简介	546
7.8.1.3.1	CEOS	546
7.8.1.4	RADARSAT 产品介绍	546
7.8.1.4.1	RADARSAT1 产品	548
7.8.1.4.1.1	RADARSAT1 雷达参数与工作模式	548
7.8.1.4.1.2	RADARSAT1 参数分析	549

7.8.1.4.1.3	RADARSAT1 数据读取	550
7.8.1.4.1.4	数据简介	550
7.8.1.4.1.5	数据格式与读取	551
7.8.1.4.1.6	AGC 补偿	552
7.8.1.4.1.7	RADARSAT1 数据成像	552
7.8.1.4.1.8	全部区域成像	552
7.8.1.4.1.9	部分区域成像	558
7.8.1.4.2	RADARSAT2 产品介绍	558
7.8.1.4.2.1	RADARSAT2 雷达参数	558
7.8.1.5	ERS 系列产品介绍	558
7.8.1.5.1	雷达参数与工作模式	558
7.8.1.5.2	数据获取与读取	562
7.8.1.5.3	原始回波数据读取	562
7.8.1.5.3.1	目录及文件格式	562
7.8.1.5.3.2	数据读取与解析	563
7.8.1.5.4	单视复数数据读取	565
7.8.1.5.5	原始数据成像	569
7.8.1.6	Sentinel 系列产品介绍	576
7.8.1.6.1	Sentinel-1 产品	576
7.8.1.6.1.1	Sentinel-1 雷达参数与工作模式	576
7.8.1.6.1.2	Sentinel-1 产品格式	577
7.8.1.6.1.3	传感器类型与产品类型	577
7.8.1.6.1.4	Sentinel 产品数据命名格式	579
7.8.1.6.1.5	数据下载与处理	579
7.8.1.6.1.6	SNAP	580
7.8.1.6.1.7	Sentinel1 Level0 级数据解析	580
7.8.1.6.2	Sentinel-2 雷达	581
7.8.1.7	AIRSAR 产品介绍	581
7.8.1.7.1	AIRSAR 简介	581
7.8.1.7.1.1	AIRSAR 雷达参数与工作模式	582
7.8.1.7.2	数据读取	582
7.8.1.7.3	数据成像	582
7.8.1.8	UAVSAR 产品介绍	582
7.8.1.8.1	UAVSAR 简介	582
7.8.1.8.1.1	UAVSAR 雷达参数	583
7.8.1.8.2	工作模式与数据	583
7.8.1.8.3	数据读取	583
7.8.1.8.4	数据成像	583
7.8.2	雷达数据集	583
7.8.2.1	简介	583
7.8.2.1.1	雷达数据集概览	583
7.8.2.2	毫米波雷达数据集	583
7.8.2.2.1	The Oxford Radar RobotCar Dataset	583
7.8.3	雷达库与软件	584
7.8.3.1	简介	584

7.8.3.1.1	雷达信号处理库与软件概览	584
7.8.3.2	ASF MapReady	584
7.8.3.2.1	简介	584
7.8.3.2.2	安装 ASF MapReady	584
7.8.3.2.2.1	Ubuntu 下配置安装	584
7.8.3.2.2.3	ASF MapReady 源码分析	585
7.8.3.2.3.1	数据读取	585
7.8.3.2.4	ASF MapReady 应用举例	588
7.8.3.2.4.1	数据导入	588
7.8.3.2.4.2	导出 ERS2 Level0 级数据	588
7.8.3.3	SAR Training Processor	588
7.8.3.3.1	基础教程	588
7.8.3.3.1.1	源码安装	588
7.8.3.3.1.2	源码简析	588
7.8.3.3.2	问题与解决	588
7.8.3.3.2.1	使用	588
7.8.3.3.2.2	内存溢出	588
7.8.3.4	STEP 与 SNAP	589
7.8.3.4.1	简介	589
7.8.3.4.2	环境配置	589
7.8.3.4.2.1	安装	589
7.8.3.4.2.2	源码构建安装	589
7.8.3.4.2.3	安装包安装	589
7.8.3.4.3	Level1 数据处理示例	591
7.8.3.4.4	Python 与 SNAP	591
7.8.3.4.4.1	简介	591
7.8.3.4.4.2	配置 Python 以使用 snappy	591
7.8.3.4.4.3	在 Python 中调用 SNAP 函数	593
7.8.3.4.4.4	开发 SNAP 的 Python 处理器插件	593
7.8.3.4.4.5	使用 Python 接口导出原始 SAR 数据	593
7.8.3.4.5	总结	593
7.8.3.4.5.1	链接	593
7.8.3.5	PolSARpro	593
7.8.3.5.1	简介	593
7.8.3.5.2	基础教程	593
7.8.3.5.2.1	安装	593
7.8.3.5.3	进阶教程	594
7.8.3.6	Other	594
7.8.3.6.1	pyroSAR	594
7.8.3.6.2	MATLAB SAR	594
7.8.3.6.3	SarPy	595
7.8.3.6.4	sumo	595
7.8.3.6.5	Sentinel Level0 数据提取	595
7.8.4	参考文献	595
7.9	名词术语	595

8 第八卷医学信号处理	597
8.1 核磁共振成像	597
8.1.1 核磁共振简介	597
8.1.1.1 资源	597
8.1.2 NMR 系统概述	598
8.1.3 NMR 成像理论	598
8.1.4 参考文献	598
8.1.5 名词术语	598
8.2 名词术语	598
9 第九卷天文学	599
9.1 天体测量学	599
9.2 天体力学	599
9.3 天体物理学	599
9.3.1 地球物理学	599
9.3.1.1 引言	599
9.3.1.2 人造地球卫星	599
9.3.1.2.1 卫星轨道参数	599
9.3.1.2.2 常见卫星轨道	599
9.3.1.2.2.1 地球同步轨道	600
9.3.1.2.2.2 半同步轨道	600
9.3.1.2.2.3 极地轨道	600
9.3.1.2.2.4 太阳同步轨道	600
10 第十卷应用实践	601
10.1 变化检测	601
10.1.1 引言	601
10.1.1.1 资源	601
10.1.1.2 数据集	601
10.1.1.3 文章与代码	602
10.1.2 基于深度学习的变化检测	602
10.1.2.1 基于深度学习的变化检测方法概述	602
10.1.2.1.1 目前进展	602
10.1.2.1.2 下一步方向	602
10.1.2.2 基于孪生网络的变化检测	602
10.1.2.2.1 基于全卷积孪生网络的变化检测	602
10.1.2.2.2 基于多尺度卷积孪生网络的变化检测	602
10.1.2.2.3 基于扩张金字塔	603
10.1.2.3 自步变化学习	603
10.1.2.3.1 动机与贡献	603
10.1.2.3.2 模型与方法	604
10.1.2.3.3 实验与分析	604
10.2 路径规划	604
10.2.1 引言	604
10.2.1.1 什么是路径规划	604

10.2.1.2 常用方法	604
10.2.1.3 有用的资源	604
10.2.1.3.1 book	604
10.2.1.3.2 software	604
10.2.1.3.3 library	605
10.2.2 覆盖规划	605
10.2.2.1 引言	605
10.2.2.1.1 什么是区域覆盖规划	605
10.2.2.1.2 区域覆盖规划算法	605
10.2.2.2 几何法	605
10.2.2.2.1 初始规划	605
10.2.2.2.2 障碍物躲避	605
10.2.2.2.2.1 最短圆弧法	605
10.2.2.2.3 价值区域穿越	606
10.2.2.2.3.1 矩形插入法	606
10.2.2.2.4 实验与分析	607
10.2.2.2.4.1 实验 1	607
10.2.2.2.4.2 实验代码	607
10.2.2.2.4.3 实验结果	609
10.2.3 动态规划	609
10.2.4 参考文献	609
10.2.5 名词术语	609
10.3 评价方法	610
10.3.1 经典评价方法	610
10.3.1.1 信息检索	610
10.3.1.1.1 常用数据检索指标	610
10.3.1.1.1.1 基本概念	610
10.3.1.1.1.2 查准率	610
10.3.1.1.1.3 查全率	610
10.3.1.1.1.4 F-measure	611
10.3.1.1.1.5 虚警率	612
10.3.1.1.1.6 漏检率	612
10.3.1.1.1.7 总结	612
10.3.1.1.2 数据分类	613
10.3.1.1.2.1 混淆矩阵	613
10.3.1.1.2.2 Kappa 系数	613
10.3.1.1.3 相似性度量指标	613
10.3.1.1.3.1 集合方法	613
10.3.1.1.3.1.1 Jaccard	613
10.3.1.1.3.1.2 Jaccard 指数与距离	613
10.3.1.1.3.1.3 加权 Jaccard 指数与距离	614
10.3.1.1.3.1.4 概率 Jaccard 指数与距离	614
10.3.1.1.3.1.5 Dice 系数	614
10.3.1.1.3.2 结构相似性度量	615
10.3.1.1.4 图像质量评价	615

10.3.1.4.1	误差类	615
10.3.1.4.1.1	均方误差	615
10.3.1.4.2	信噪比类	615
10.3.1.4.3	结构相似性度量	615
10.3.1.4.3.1	SSIM	615
10.3.1.4.3.2	MSSSIM	616
10.3.1.4.3.3	GSSIM	616
10.3.1.4.4	实验与分析	616
10.3.1.4.4.1	核心代码	616
10.3.1.5	计算复杂度/性能	618
10.3.1.5.1	操作数相关指标	618
10.3.1.5.2	处理器的衡量指标	619
10.3.1.5.2.1	传统指标	619
10.3.1.5.3	计算平台与模型性能指标	619
10.3.2	名词术语	621
10.4	补充内容	621
10.4.1	数据集	621
10.4.1.1	数据集概览	621
10.4.1.1.1	混合	621
10.4.1.1.2	遥感影像	621
10.4.1.1.2.1	综合	621
10.4.1.2	变化检测数据集	622
10.4.1.2.1	遥感影像	622
10.4.1.2.2	自然场景	623
10.4.1.3	图像分割数据集	623
10.4.1.3.1	视网膜图像	623
10.4.1.4	雷达数据获取	623
10.4.1.4.1	公开数据	623
11	第十一卷补充内容	625
11.1	引言	625
11.2	写作与排版	625
11.2.1	LaTex 简明教程	625
11.2.1.1	LaTex 基础	625
11.2.1.1.1	数学相关	625
11.2.1.1.1.1	定义定理证明等	625
11.2.1.2	LaTex 进阶	626
11.2.1.3	问题解决	626
11.2.1.3.1	安装	626
11.2.1.3.2	使用	626
11.2.1.3.2.1	Extra alignment tab has been changed to cr	626
11.2.2	文献管理与引用	626
11.2.2.1	简介	626
11.2.2.2	文献管理	626
11.2.2.2.1	常见文献管理软件	626

11.2.2.2.2 推荐的管理软件	628
11.2.2.2.3 文献引用格式	628
11.2.2.2.3.1 常见格式介绍	628
11.2.2.2.3.2 注意事项	628
11.2.2.3 文献引用	630
11.2.2.3.1 在文章中引用文献	630
11.2.2.3.1.1 引用样式	630
11.2.2.3.1.2 期刊缩写	630
11.3 生活常识	631
11.3.1 电子电器篇	631
11.3.1.1 声音设备	631
11.3.1.1.1 耳机	631
11.3.1.1.2 红外遥控	632
11.3.1.1.2.1 红外遥控原理	632
11.3.1.1.2.1.1 红外发射	632
11.3.1.1.2.1.2 红外接收	632
11.3.1.1.2.2 手机方案	632
11.3.1.1.2.2.1 3.5mm 耳机无源方案	632
11.3.1.1.2.2.2 3.5mm 耳机有源方案	632
11.3.1.1.2.2.3 USB 有源方案	632
11.3.1.1.2.2.4 应用软件	632
12 名词术语	633
13 Indices and tables	637
Bibliography	639

- 邮箱 (email) : zhiliu.mind@gmail.com
- 博客 (blog) : <http://blog.csdn.net/enjoyyl>
- 主页 (homepage) : iridescent.ink (<https://iridescent.ink/>)



提示: PDF 版文档可以点[这里查看](#)

写于 2018 年 06 月 10 日

CHAPTER 1

引言

1.1 手册自述

1.1.1 目标

掌握雷达基本原理及雷达数据智能化处理的方法。

1.1.2 撰写与发布

该手册采用 reStructuredText 标记语言撰写, 使用 Sphinx 进行发布.

属性	值
开发语言	reStructuredText (http://docutils.sourceforge.net/rst.html)
发布工具	Sphinx (http://www.sphinx-doc.org/en/stable/)

- 邮箱 (email) : zhiliu.mind@gmail.com
- 博客 (blog) : <http://blog.csdn.net/enjoyyl>
- 主页 (homepage) : iridescent.ink

Contact QR Code

写于 2018 年 03 月 10 日

CHAPTER 2

第一卷数学基础

2.1 引言

涉及高等数学, 线性代数, 矩阵论, 概率论, 随机过程, 泛函分析等.

2.2 概念解析

2.2.1 空间

2.2.1.1 空间简介

在数学中, 并未单独定义空间 (*Space*) 的概念, 维基百科上给出的不同空间关系结构图示如下

2.2.1.2 度量空间

度量空间 (*Metric Space*)

2.2.1.3 希尔伯特空间

希尔伯特空间 (*Hilbert Space*)

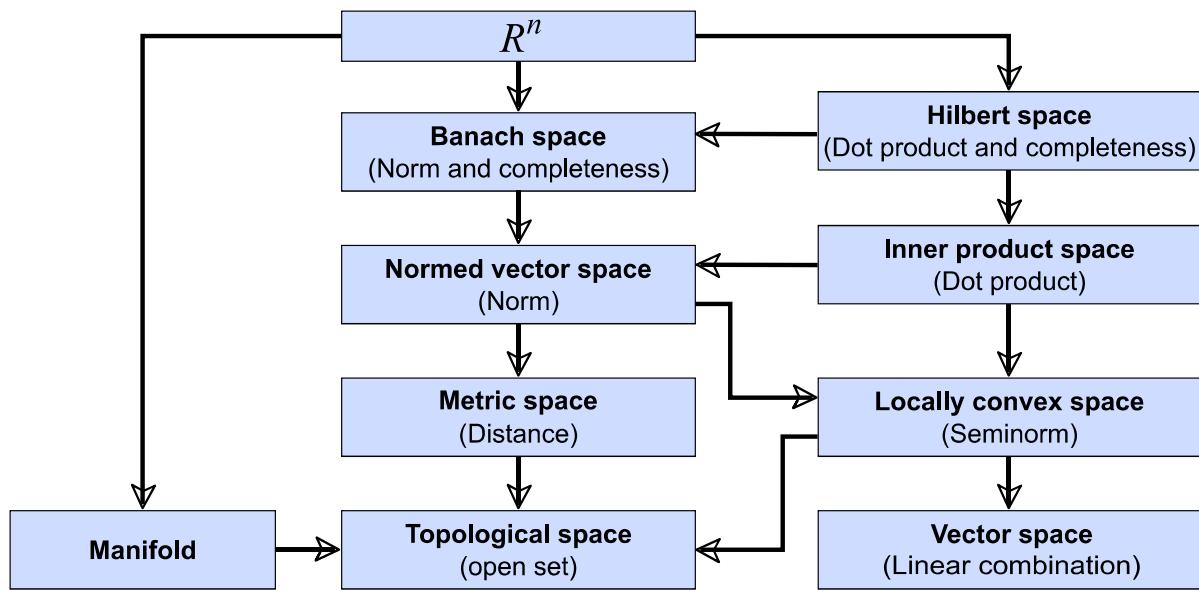


图 2.1: Type of Spaces.

Type of Spaces.

2.2.1.4 巴拿赫空间

巴拿赫空间 (*Banach Space*)

2.2.1.5 拓扑空间

拓扑空间 (*Topological Space*)

2.2.1.6 黎曼空间

黎曼空间 (*Riemannian Space*)

2.2.2 问题

2.2.2.1 简介

主要阐述常见的问题概念, 如: 病态问题, 良态问题, 不适定问题, 适定问题, 线性逆问题非线性逆问题等等.

2.2.2.2 病态与良态

问题的病态与良态是根据 **问题输出随问题输入扰动的变化程度** 定义的, 这种扰动程度通过 **条件数** 来定义, 故先说明条件数的含义. 具有小条件数的问题是良态的, 相反, 具有大条件数的问题是病态的.

http://en.volupedia.org/wiki/Condition_number

2.2.2.2.1 条件数

Definition 1 (条件数) 没有问题 $f(x)$, 解算法 $\tilde{f}(x)$ 和输入 x . 记输入的扰动 (或误差) 为 δx 解的扰动 (或误差) 为 δf , 则有绝对误差 $\|\delta f\| = \|f(x) - \tilde{f}(x)\|$ 和相对误差 $\|\delta f\|/\|f(x)\|$.

绝对条件数 定义为

$$\lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta f(x)\|}{\|\delta x\|}$$

相对条件数 定义为

$$\lim_{\epsilon \rightarrow 0} \sup_{\|\delta x\| \leq \epsilon} \frac{\|\delta f(x)\|/\|f(x)\|}{\|\delta x\|/\|x\|}$$

2.2.2.2 病态的

病态的 (*Ill-conditioned*)

2.2.2.3 良态的

良态的 (*Well-conditioned*)

2.2.3 适定与不适定

2.2.3.1 不适定问题

不适定问题 (Ill-posed Problem)

2.2.3.2 适定问题

适定问题 (Well-posed Problem)

2.2.4 逆问题

逆问题 (*Inverse Problem*) 按线性和非线性可以分为: 线性逆问题 (Linear Inverse Problem), 非线性逆问题 (Nonlinear Inverse Problem). 按适定性可分为适定性问题 (Well-posed Problem), 不适定性问题 (Ill-posed Problem).

<http://www.inverseproblems.info>

2.2.4.1 线性逆问题

2.2.4.2 非线性逆问题

2.2.3 正则化

2.2.3.1 正则化简介

在数学, 统计学和计算机科学中, 特别是在机器学习和逆问题的求解中, 正则化通常通过增加额外的信息解决不适定问题或防止过度拟合. 在不同领域有着不同的正则化技术, 大多是基于范数的正则, 主要有如下几类:

- 基于范数的正则: $\ell_{1/2}, \ell_0, \ell_1, \ell_2$, 核范数 (Nuclear Norm), 拉普拉斯范数 (Laplacian Norm), 谱范数 (Spectral Norm) 正则, 以及 Tikhonov 正则 (*Tikhonov Regularization*, 在统计学中称为岭回归) 等等.
- 基于噪声的正则: 如在神经网络中的加噪声, dropout 等技术.
- 稀疏正则 (Sparse Regularization): $\ell_{1/2}, \ell_0, \ell_1$, 列稀疏正则等等
- 流形正则 (*Manifold Regularization*): 通常为拉普拉斯范数正则.

- 其它正则: 及早停止 (Early stopping) 也是一种正则化技术.

2.2.3.2 范数正则

2.2.3.2.1 常见范数正则

2.2.3.2.2 Tikhonov 正则

Tikhonov 正则 (*Tikhonov Regularization*)

2.2.3.3 稀疏正则

2.2.3.4 流形正则

2.2.3.5 噪声正则

2.2.4 名词术语

Banach Space 巴拿赫空间 ([Banach Space](http://en.volupedia.org/wiki/Banach_space) (http://en.volupedia.org/wiki/Banach_space))

Hilbert Space 希尔伯特空间 ([Hilbert Space](http://en.volupedia.org/wiki/Hilbert_space) (http://en.volupedia.org/wiki/Hilbert_space))

ill-conditioned 病态的 ([ill-conditioned](http://en.volupedia.org/wiki/Condition_number) (http://en.volupedia.org/wiki/Condition_number))

ill-posed 不适定的 ([ill-posed](http://www.inverseproblems.info) (<http://www.inverseproblems.info>))

Inverse Problem 逆问题 ([Inverse Problem](http://www.inverseproblems.info) (<http://www.inverseproblems.info>)) 分为线性逆问题和非线性逆问题.

Manifold Regularization 流形正则化 ([Manifold Regularization](http://en.volupedia.org/wiki/Manifold_regularization) (http://en.volupedia.org/wiki/Manifold_regularization))

Metric Space 度量空间 ([Metric Space](http://en.volupedia.org/wiki/Metric_space) (http://en.volupedia.org/wiki/Metric_space))

Regularization 正则化 ([Regularization](http://en.volupedia.org/wiki/Regularization_(mathematics)) ([http://en.volupedia.org/wiki/Regularization_\(mathematics\)](http://en.volupedia.org/wiki/Regularization_(mathematics))))

Riemannian Space 黎曼空间 ([Riemannian Space](http://en.volupedia.org/wiki/Riemannian_geometry) (http://en.volupedia.org/wiki/Riemannian_geometry))

Space 空间 ([Space](http://en.volupedia.org/wiki/Space_(mathematics)) ([http://en.volupedia.org/wiki/Space_\(mathematics\)](http://en.volupedia.org/wiki/Space_(mathematics))))

Sparsity Regularization 稀疏正则化 ([Sparsity Regularization](http://en.volupedia.org/wiki/Structured_sparsity_regularization#Spa) (http://en.volupedia.org/wiki/Structured_sparsity_regularization#Spa))

Tikhonov Regularization Tikhonov 正则化 ([Tikhonov Regularization](http://en.volupedia.org/wiki/Tikhonov_regularization) (http://en.volupedia.org/wiki/Tikhonov_regularization))

Topological Space 拓扑空间 ([Topological Space](http://en.volupedia.org/wiki/Topological_space) (http://en.volupedia.org/wiki/Topological_space))

Total Variation 全变分 ([Total Variation](http://en.volupedia.org/wiki/Total_variation) (http://en.volupedia.org/wiki/Total_variation))

Total Variation Denoising 全变分去噪 ([Total Variation Denoising](http://en.volupedia.org/wiki/Total_variation_denoising) (http://en.volupedia.org/wiki/Total_variation_denoising))

Total Variation Regularization Total Variation 正则化 (Tikhonov Regularization (http://en.volupedia.org/wiki/Total_variation_denoising))，简称 TV 正则.

well-conditioned 良态的 ([well-conditioned](http://en.volupedia.org/wiki/Condition_number) (http://en.volupedia.org/wiki/Condition_number))

well-posed 适定的 ([well-posed](http://en.volupedia.org/wiki/Well-posed_problem) (http://en.volupedia.org/wiki/Well-posed_problem))

2.3 数论

2.3.1 素数问题

2.3.2 名词术语

Prime Number 素数 ([Prime Number](http://en.volupedia.org/wiki/Prime_Number) (http://en.volupedia.org/wiki/Prime_Number))

2.4 高等数学

2.4.1 引言

让我们开始吧!!!

2.4.2 集合论

2.4.2.1 集合的概念与性质

2.4.2.1.1 一般集合

2.4.2.1.1.1 概念

定义: 集合 ([set](#)) 是指具有某种特定性质的具体的或抽象的对象构成的总体. 其中, 构成集合的这些对象称为该集合的 **元素**.

注解: 将上述定义的集合称为一般集合. 一个元素是否在一个集合中, 在于该元素是否具备上述特定性质, 该性质是具体的明确的, 如果该性质不具体不明确呢?

一般地集合是指具有如下性质的集合:

1. 确定性: 即任给一元素, 它要么在该集合中, 要么不在.
2. 互异性: 即该集合中的任意两个元素不同, 每个元素只能出现一次.
3. 无序性: 即该集合中的元素无顺序关系.

注解: 关于上述性质, 思考如下:

0. 上述定义的一般集合的概念可以概括所有集合, 即对象全体吗?
 1. 确定性: 如果元素以一定的可能性在呢? 元素不确定, 上述特定性质不确定?
 2. 互异性: 如果有两个及以上的元素相同呢? 如果出现多次呢? 可否定义集合规则: 可以出现 N 次的对象全体?(可以, 不过还是普通集合); 可否定义集合规则: 可以出现随机次数的对象全体?(可以, 不过还是普通集合); 如给定对象全体 $\{1, 2, 2, 3, 3, 3\}$, 如果不指明特定性质, 是不能判断它是否是一般的集合. 如果定义集合为: 出现次数等于其数值且小于 4 的正整数集, 上述对象化为一般集合 $\{1, 2, 3\}$.
 3. 无序性: 若有序呢? 可否定义集合规则: 对象全体?
-

2.4.2.1.1.2 元素与集合的关系

给定集合 S , 元素 x , 定义如下关系:

1. 属于 (\in): $x \in S$
2. 不属于 (\notin): $x \notin S$

2.4.2.1.1.3 集合与集合的关系

给定集合 A, B , 定义如下关系:

- 包含 (\subseteq): $A \subseteq B$, 称 A 包含于 B 或 B 包含 A .
- 真包含 (\subsetneq): $A \subsetneq B$, 称 A 真包含于 B 或 B 真包含 A .
- 等于 ($=$): $A = B$, 集合 A 中的元素都在集合 B 中, 且集合 B 中的元素都在集合 A 中.
- 不等于 (\neq): $A \neq B$, 集合 A 中, 至少有一个元素不在集合 B 中.

定义: 子集 (*Subset*), 一般地, 对于集合 A, B , 若集合 A 中的元素都是集合 B 中的元素, 即 $\forall x \in A$, 有 $x \in B$, 则称这两个集合具有包含关系, 称 A 包含于 (\subseteq) B 或 B 包含 (\supseteq) A . 空集 (没有任何元素, 记为 \emptyset) 是任何集合的子集.

定义: 真子集 (*Proper Subset*) 如果集合 A 是 B 的子集, 且 $A \neq B$, 即 B 中至少有一个元素不属于 A , 那么 A 就是 B 的真子集, 可记作: $A \subsetneq B$.

2.4.2.1.1.4 集合间的运算

- 交 (\cap): $A \cap B = \{x \mid x \in A \text{ 且 } x \in B\}$, 如 $A \cap B$.
- 并 (\cup): $A \cup B = \{x \mid x \in A \text{ 或 } x \in B\}$, 如 $\{1, 2\} \cup \{3, 4, 5\} = \{1, 2, 3, 4, 5\}$.
- 和 ($+$): $A + B = \{x + y \mid x \in A, y \in B\}$, 如 $\{1, 2\} + \{2, 4\} = \{3, 4, 5, 6\}$.
- 差 ($-$): $A - B = \{x \mid x \in A, x \notin B\}$, 也称相对补集, 如 $\{1, 2\} + \{2, 4\} = \{1\}$.
- 补 (C), 设 U 是一个集合, A 是 U 的一个子集, 由 U 中所有不属于 A 的元素组成的集合, 叫做子集 A 在 U 中的补集, 也称绝对补集, 记作 $C_U A$.

2.4.2.1.1.5 集合间的运算定律

1. 交换律: $A \cap B, A \cup B, A + B$
2. 结合律: $A \cap (B \cap C) = (A \cap B) \cap C, A \cup (B \cup C) = (A \cup B) \cup C, (A + B) + C = A + (B + C)$
3. 分配律: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C), A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
4. 对偶律: xxx
5. 同一律: $A \cap U = A, A \cup \emptyset = A$
6. 零一律: $A \cap \emptyset = \emptyset, A \cup U = U$
7. 吸收律: $A \cap (A \cup B) = A, A \cup (A \cap B) = A$
8. 等幂律: $A \cap A = A, A \cup A = A$
9. 反演律 (德·摩根律): $C_U A \cap B = C_U A \cup C_U B, C_U A \cup B = C_U A \cap C_U B$

2.4.2.1.1.6 集合的特征函数

Definition 2 (集合的特征函数) 设有集合 X 和 A , 且 A 是 X 的子集, 即 $A \subset X$, 称函数

$$f(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \quad (2.1)$$

为集合 A 在 X 中的 **特征函数** (*Characteristic Function*).

2.4.2.1.1.7 集合的势与基数

在集合论中, 集合 A 的势 (cardinality) 用于指集合的元素的个数, 通常记作 $|A|$ 或 $\text{card}A$ 或 $n(A)$.

基数 (cardinal number)

2.4.2.1.2 其它集合

2.4.2.1.2.1 模糊集

具体参见[模糊集合](#) (页 123)

2.4.2.1.2.2 粗糙集

具体参见[模糊集合](#) (页 123)

2.4.3 名词术语

Crisp Set 明确集合 ([Crisp Set](http://en.volupedia.org/wiki/Crisp_set) (http://en.volupedia.org/wiki/Crisp_set)) 指经典集合, 主要强调元素是否属于集合是确定的.

Fuzzy Set 模糊集合 ([Fuzzy Set](http://en.volupedia.org/wiki/Fuzzy_set) (http://en.volupedia.org/wiki/Fuzzy_set)) 与经典集合不同, 主要强调元素是否属于集合是不确定的, 用隶属函数与隶属度来表示.

Rough Set 粗糙集合 ([Rough Set](http://en.volupedia.org/wiki/Rough_set) (http://en.volupedia.org/wiki/Rough_set)) 与经典集合不同, 主要强调元素是否属于集合是不确定的, 用两个上下隶属函数与隶属度来表示.

2.5 矩阵论

2.5.1 线性空间与线性变换

2.5.1.1 线性空间

2.5.1.1.1 基础概念

2.5.1.1.1.1 域

- 域: 一种集合
- 数域: 满足四则运算封闭
- 事件域: 满足交并补运算封闭

2.5.1.1.1.2 映射

- 像: 映射后, 相当于因变量
- 原像: 映射前, 相当于自变量
- 相等: 变换后相同, σ_1, σ_2 是集合 S_1 到 S_2 的映射, 若 $\forall a \in S_1$, 有 $\sigma_1(a) = \sigma_2(a)$, 则 $\sigma_1 = \sigma_2$
- 乘积: 设 σ, τ 分别为 S 到 S_1, S_1 到 S_2 的映射, 则映射的乘积定义为 $(\tau\sigma)(a) = \tau(\sigma(a))$

2.5.1.1.2 线性空间概念及性质

2.5.1.1.2.1 定义

涉及一个集合和一个数域, 五种运算, 分别为:

- 集合内
 - 加法 (元素的加法): \oplus
- 数域内
 - 加法 (数的加法): $+$

- 乘法 (数的乘法): \times 或省略
- 集合与数域间
 - 乘法 (数乘): \odot

Definition 3 (线性空间) 给定数域 \mathbb{K} , 非空集合 \mathbb{V} , 定义集合内的加法运算 (\oplus) 和数域与集合间的数乘运算 (\odot) , 若满足以下条件, 则称 \mathbb{V} 是数域 \mathbb{K} 上的 **线性空间** (Linear Space) .

1. 运算封闭性:

- 加法封闭性: 对 $\forall \mathbf{x}, \mathbf{y} \in \mathbb{V}$, 有唯一的和 $\mathbf{x} \oplus \mathbf{y} \in \mathbb{V}$
 - 数乘封闭性: 对 $\forall k \in \mathbb{K}, \forall \mathbf{x} \in \mathbb{V}$, 有唯一的元素 $\forall k\mathbf{x} \in \mathbb{V}$
2. 交换律: $\mathbf{x} \oplus \mathbf{y} = \mathbf{y} \oplus \mathbf{x}$
 3. 结合律: $(\mathbf{x} \oplus \mathbf{y}) \oplus \mathbf{z} = \mathbf{x} \oplus (\mathbf{y} \oplus \mathbf{z})$
 4. 有零元: $\forall \mathbf{x} \in \mathbb{V}, \exists \mathbf{0} \in \mathbb{V}$ 使得 $\mathbf{x} \oplus \mathbf{0} = \mathbf{x}$
 5. 有负元: $\forall \mathbf{x} \in \mathbb{V}, \exists \mathbf{y} \in \mathbb{V}$ 使得 $\mathbf{x} \oplus \mathbf{y} = \mathbf{0}$
 6. 乘 1 律: $1 \odot \mathbf{x} = \mathbf{x}$
 7. 数因子分配律: $k \odot (\mathbf{x} + \mathbf{y}) = k \odot \mathbf{x} + k \odot \mathbf{y}$
 8. 数乘分配律: $(k + l) \odot \mathbf{x} = k \odot \mathbf{x} + l \odot \mathbf{x}$
 9. 数乘结合律: $k \odot (l \odot \mathbf{x}) = (kl) \odot \mathbf{x}$

2.5.1.1.2.2 性质

$\forall \mathbf{x} \in \mathbb{V}, 0, 1 \in \mathbb{K}$, 有:

1. $0 \odot \mathbf{x} = \mathbf{0}$: 数 0 乘以 \mathbb{V} 中任意元素 \mathbf{x} 的结果为 \mathbb{V} 中零元素 $\mathbf{0}$
2. $1 \odot \mathbf{x} = \mathbf{x}$: 数 1 乘以 \mathbb{V} 中任意元素 \mathbf{x} 的结果仍为元素 \mathbf{x}
3. $-1 \odot \mathbf{x} = -\mathbf{x}$: 数 -1 乘以 \mathbb{V} 中任意元素 \mathbf{x} 的结果为 \mathbf{x} 的负元素 $-\mathbf{x}$

注解:

- $(0 \odot \mathbf{x}) \oplus \mathbf{x} = (0 + 1) \odot \mathbf{x} = \mathbf{x} \implies 0 \odot \mathbf{x} = \mathbf{0}$
 - $(-1 \odot \mathbf{x}) \oplus \mathbf{x} = (-1 + 1) \odot \mathbf{x} = \mathbf{x} = 0 \odot \mathbf{x} = \mathbf{0} \implies -1 \odot \mathbf{x}$ 为 \mathbf{x} 的负元素
 - $k \odot (\mathbf{x} \oplus (-1 \odot \mathbf{x})) = k \odot \mathbf{x} \oplus k \odot (-\mathbf{x}) = k \odot \mathbf{x} \oplus (-1) \odot (k \odot \mathbf{x}) = \mathbf{0}$
-

2.5.1.1.2.3 线性组合与线性表示

Definition 4 (线性组合) 设 x_1, x_2, \dots, x_m 为线性空间 \mathbb{V} 中的 m 个元素, $x \in \mathbb{V}$, 若存在数 $c_1, c_2, \dots, c_m \in \mathbb{K}$, 使

$$x = c_1 \odot x_1 \oplus c_2 \odot x_2 \oplus \dots \oplus c_m \odot x_m$$

则称 x 为 x_1, x_2, \dots, x_m 的 **线性组合** (Linear Combination), 也称 x 可由 x_1, x_2, \dots, x_m **线性表示** (Linear Representation).

注解: 对比稀疏表示, 字典学习:

- 原子可看作上述向量;
- 字典可看作上述向量组;
- 稀疏表示系数就是上述线性组合系数.

2.5.1.1.2.4 线性相关与线性无关

Definition 5 (线性无关) 设 x_1, x_2, \dots, x_m 为线性空间 \mathbb{V} 中的 m 个元素, 若 存在一组不全为零的数 $c_1, c_2, \dots, c_m \in \mathbb{K}$, 使

$$c_1 \odot x_1 \oplus c_2 \odot x_2 \oplus \dots \oplus c_m \odot x_m = \mathbf{0}$$

则称 x_1, x_2, \dots, x_m **线性相关** (Linearly Dependent), 否则, 称其 **线性无关** (Linearly Independent).

提示: 若 x_1, x_2, \dots, x_m 线性无关, 则当且仅当 $c_1, c_2, \dots, c_m \in \mathbb{K}$ 全为零时, $c_1 \odot x_1 \oplus c_2 \odot x_2 \oplus \dots \oplus c_m \odot x_m = \mathbf{0}$ 成立.

2.5.1.1.3 线性空间的基、坐标与维数

2.5.1.1.3.1 定义

Definition 6 (线性空间的基、坐标与维数) 设 \mathbb{V} 是数域 \mathbb{K} 上的线性空间, $x_1, x_2, \dots, x_n (n \geq 1)$ 是属于 \mathbb{V} 的任意 n 个元素, 若它们满足:

- (1) $x_1, x_2, \dots, x_n (n \geq 1)$ 线性无关;
- (2) $\forall x \in \mathbb{V}$, 均可由 $x_1, x_2, \dots, x_n (n \geq 1)$ 线性表示, 记为

$$x = \xi_1 \odot x_1 \oplus \xi_2 \odot x_2 \oplus \dots \oplus \xi_n \odot x_n$$

则称

- $x_1, x_2, \dots, x_n (n \geq 1)$ 是 \mathbb{V} 一个 基 或 基底;
 - $\xi_1, \xi_2, \dots, \xi_n (n \geq 1)$ 为 x 在该基下的 坐标.
 - \mathbb{V} 中最大线性无关的元素的数目为线性空间 \mathbb{V} 的 维数, 记为 $\dim(\mathbb{V})$
-

提示:

1. 基不一定是向量, 可能是矩阵, 或者其它更为抽象的;
 2. 坐标是数.
-

2.5.1.1.4 基变换与坐标变换

从一个基到另一个基时, 坐标的变换.

提示: 以下部分, 在不引起混淆时, 对集合内的加法 (\oplus) 与数域内的加法 (+), 及数乘 (\odot) 与数域内的乘法 (\times 或省略) 不作区分.

2.5.1.1.4.1 两个基之间的转换

基变换: 设 x_1, x_2, \dots, x_n 是 \mathbb{V}^n 的旧基, y_1, y_2, \dots, y_n 是 \mathbb{V}^n 的新基, 则有

$$\left\{ \begin{array}{l} y_1 = c_{11} \odot x_1 \oplus c_{21} \odot x_2 \oplus \cdots \oplus c_{n1} \odot x_n \\ y_2 = c_{12} \odot x_1 \oplus c_{22} \odot x_2 \oplus \cdots \oplus c_{n2} \odot x_n \\ \vdots \\ y_n = c_{1n} \odot x_1 \oplus c_{2n} \odot x_2 \oplus \cdots \oplus c_{nn} \odot x_n \end{array} \right.$$

写成矩阵形式为

$$(y_1, y_2, \dots, y_n) = (x_1, x_2, \dots, x_n) \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = (x_1, x_2, \dots, x_n) C$$

即:

$$(y_1, y_2, \dots, y_n) = (x_1, x_2, \dots, x_n) C$$

同理可得从新基变回旧基的表达式:

$$(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n) A = (x_1, x_2, \dots, x_n) CA$$

显然, $A = C^{-1}$

2.5.1.1.4.2 多个基之间的转换

设有线性空间: \mathbb{V}^n , 及其三个基:

- 基 1: $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$
- 基 2: $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n$
- 基 3: $\mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_n$

且设由基 3 到基 1 的过度矩阵为 C_1 , 且由基 3 到基 2 的过度矩阵为 C_2 , 则由基 1 到基 2 的过度矩阵为 $C_1^{-1}C_2$ 由基 2 到基 1 的过度矩阵为 $C_2^{-1}C_1$, 即

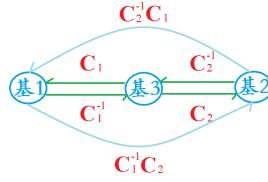


图 2.2: 多个基之间的转换
多个基之间的转换关系

$$(\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n) = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n) C_1^{-1} C_2$$

$$(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n) = (\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n) C_2^{-1} C_1$$

2.5.1.1.4.3 坐标变换

坐标变换: 设向量 z 在上述旧基与新基下可表示为:

$$z = \xi_1 \mathbf{x}_1 + \xi_2 \mathbf{x}_2 + \dots + \xi_n \mathbf{x}_n = \eta_1 \mathbf{y}_1 + \eta_2 \mathbf{y}_2 + \dots + \eta_n \mathbf{y}_n$$

则

$$z = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \mathbf{C} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix}$$

从而有

$$(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \mathbf{C} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix}$$

由此得坐标变换公式

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} = \mathbf{C}^{-1} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix}$$

2.5.1.2 线性子空间

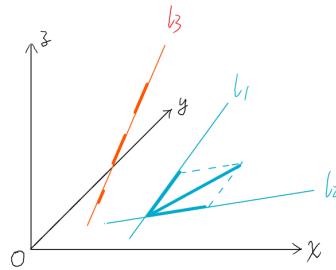
2.5.1.2.1 线性子空间

Definition 7 (线性子空间) 设 \mathbb{V}_1 是数域 \mathbb{K} 上的线性空间 \mathbb{V} 的非空子集，若对任意的 $k \in \mathbb{K}$, $\mathbf{x}, \mathbf{y} \in \mathbb{V}_1$, 满足 \mathbb{V} 中加法与数乘运算的封闭性，则称 \mathbb{V}_1 是 \mathbb{V} 的 **线性子空间** (Linear Subspace)，即满足：

1. 加法运算 \oplus 封闭性: $\mathbf{x} \oplus \mathbf{y} \in \mathbb{V}_1$
2. 数乘运算 \odot 封闭性: $k \odot \mathbf{x} \in \mathbb{V}_1$

注解: 举例: 对于三维欧式线性空间 \mathbb{V}^3 :

1. \mathbb{V}^3 中的任意一条直线构成其子空间 \mathbb{V}_1
2. \mathbb{V}^3 中的任意两条不同的直线不构成其子空间



2.5.1.2.2 子空间的运算及其性质

设 $\mathbb{V}_1, \mathbb{V}_2$ 是 \mathbb{V} 的子空间，定义如下三种运算：

- 交 (\cap): $\mathbb{V}_1 \cap \mathbb{V}_2$;
- 并 (\cup): $\mathbb{V}_1 \cup \mathbb{V}_2$;
- 和 ($+$): $\mathbb{V}_1 + \mathbb{V}_2$.

性质：

- $\mathbb{V}_1 \cap \mathbb{V}_2 \subseteq \mathbb{V}_1 \cup \mathbb{V}_2 \subseteq \mathbb{V}_1 + \mathbb{V}_2$
- $\mathbb{V}_1 \cap \mathbb{V}_2 = \mathbb{V}_2 \cap \mathbb{V}_1, (\mathbb{V}_1 \cap \mathbb{V}_2) \cap \mathbb{V}_3 = \mathbb{V}_1 \cap (\mathbb{V}_2 \cap \mathbb{V}_3)$
- $\mathbb{V}_1 + \mathbb{V}_2 = \mathbb{V}_2 + \mathbb{V}_1, (\mathbb{V}_1 + \mathbb{V}_2) + \mathbb{V}_3 = \mathbb{V}_1 + (\mathbb{V}_2 + \mathbb{V}_3)$
- 交 (\cap): $\mathbb{V}_1 \cap \mathbb{V}_2$ 是 \mathbb{V} 的子空间；
- 和 ($+$): $\mathbb{V}_1 + \mathbb{V}_2$ 是 \mathbb{V} 的子空间。
- 并 (\cup): $\mathbb{V}_1 \cup \mathbb{V}_2$ 不是 \mathbb{V} 的子空间；

- 维数公式:

$$\underbrace{\dim \mathbb{V}_1}_{n_1} + \underbrace{\dim \mathbb{V}_2}_{n_2} = \underbrace{\dim(\mathbb{V}_1 + \mathbb{V}_2)}_{n} + \underbrace{\dim(\mathbb{V}_1 \cap \mathbb{V}_2)}_{m}$$

注解: 举例:

- \mathbb{V}_1 : x 轴上的点
 - \mathbb{V}_2 : y 轴上的点
 - $\mathbb{V}_1 \cap \mathbb{V}_2$: 原点
 - $\mathbb{V}_1 \cup \mathbb{V}_2$: x 轴和 y 轴上的点
 - $\mathbb{V}_1 + \mathbb{V}_2$: 二维平面
 - 维数公式: $n_1 + n_2 = n + m \Rightarrow 1 + 1 = 2 + 0$
-

2.5.1.2.2 直和

2.5.1.2.2.1 什么是直和

和的特例，即存在唯一分解。

Definition 8 (直和) 若 $\forall z \in \mathbb{V}_1 + \mathbb{V}_2$, 有唯一的 $x \in \mathbb{V}_1, y \in \mathbb{V}_2$, 则称 $\mathbb{V}_1 + \mathbb{V}_2$ 为 **直和** (Direct Sum), 记为 $\mathbb{V}_1 + \mathbb{V}_2$ 或 $\mathbb{V}_1 \oplus \mathbb{V}_2$.

2.5.1.2.2.2 性质

- $\mathbb{V}_1 \oplus \mathbb{V}_2 \Leftrightarrow \mathbb{V}_1 \cap \mathbb{V}_2 = L(\mathbf{0}) \Leftrightarrow \dim(\mathbb{V}_1) + \dim(\mathbb{V}_2) = \dim(\mathbb{V}_1 + \mathbb{V}_2)$

2.5.1.2.3 生成的子空间

2.5.1.2.3.1 什么是生成子空间

Definition 9 (生成子空间) 设 x_1, x_2, \dots, x_m 是数域 \mathbb{K} 上的线性空间 \mathbb{V} 上的一组向量, 其所有线性组合的集合

$$\mathbb{V}_1 = \{k_1 x_1 + k_2 x_2 + \dots + k_m x_m\}, k_i \in \mathbb{K}, i = 1, 2, \dots, m$$

是 \mathbb{V} 的一个线性子空间, 称其为由 x_1, x_2, \dots, x_m 张成/生成的子空间 (Linear Span), 记为

$$L(x_1, x_2, \dots, x_m) = \{k_1 x_1 + k_2 x_2 + \dots + k_m x_m\}$$

2.5.1.2.3.2 矩阵的列空间、核空间、秩与零度

设有矩阵 $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, $a_i (i = 1, 2, \dots, n)$ 为 A 的第 i 个列向量, 称矩阵 A 的所有列向量张成的空间 $L(a_1, a_2, \dots, a_n)$ 为矩阵 A 的 **列空间 (值域)**, 记为

$$\mathcal{R}(A) = L(a_1, a_2, \dots, a_n)$$

则矩阵 A 列空间的维度定义为矩阵 A 的 **秩**

$$\text{rank } A = \dim \mathcal{R}(A)$$

设有矩阵 $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, $a_i (i = 1, 2, \dots, n)$, 称集合 $\{x | Ax = \mathbf{0}\}$ 为矩阵 A 的 **核空间 (Kernel Space)** 或 **零空间 (Null Space)**, 记为

$$\mathcal{N}(A) = \{x | Ax = \mathbf{0}\}$$

则矩阵 A 核空间的维度定义为矩阵 A 的 **零度**

$$n(A) = \dim \mathcal{N}(A)$$

提示: 设 u, v 是方程组 $Ax = b$ 的两个解, 则有

$$\begin{aligned} Au &= b \\ Av &= b \end{aligned} \Rightarrow A(u - v) = b - b = \mathbf{0}$$

故任意不同的两个线性方程组的解的差位于矩阵 A 的零空间中.

进一步地, 设 v 是方程组 $Ax = b$ 的一个解, 那么其任意解集可表示为

$$\{v + x | Av = b \wedge x \in \mathcal{N}(A)\} \quad (2.2)$$

2.5.1.2.3.3 矩阵的 Spark

设有矩阵 A , 称其列空间中线性相关的向量数目为矩阵 A 的 **Spark**, 记为 $\text{Spark}(A)$.

提示:

- 矩阵的 Rank 是最大线性无关的列数
- 矩阵的 Spark 是最小线性相关的列数

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Leftrightarrow \begin{cases} \text{Rank}(A) = 3 \\ \text{Spark}(A) = 2 \end{cases}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Leftrightarrow \begin{cases} \text{Rank}(A) = 3 \\ \text{Spark}(A) = 3 \end{cases}$$

$$\mathbf{A}_3 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \Leftrightarrow \begin{cases} \text{Rank}(\mathbf{A}) = 3 \\ \text{Spark}(\mathbf{A}) = 4 \end{cases}$$

2.5.1.2.3.4 总结

- 矩阵的列空间/值域:

$$\mathcal{R}(\mathbf{A}) = \{\mathbf{Ax} | \mathbf{x} \in R^n\}$$

- 矩阵的核空间/零空间:

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} | \mathbf{Ax} = 0\}$$

- 矩阵的秩 (列空间的维度): $\text{rank } \mathbf{A} = \dim \mathcal{R}(\mathbf{A})$
- 矩阵的零度 (核空间的维度): $n(\mathbf{A}) = \dim \mathcal{N}(\mathbf{A})$
- $\text{rank } \mathbf{A} + n(\mathbf{A}) = n$
- $\text{rank } \mathbf{A}^T + n(\mathbf{A}^T) = m$
- $n(\mathbf{A}) - n(\mathbf{A}^T) = n - m$

2.5.1.3 线性变换及其表示

2.5.1.3.1 线性变换

2.5.1.3.1.1 什么是线性变换

Definition 10 (变换) 设 \mathbb{V} 是数域 \mathbb{K} 上的线性空间, T 是 \mathbb{V} 到自身的一个映射, 使得对于 $\forall \mathbf{x} \in \mathbb{V}$, \mathbb{V} 中都有唯一的向量 \mathbf{y} 与之对应, 则称 T 是 \mathbb{V} 的一个 **变换** 或 **算子**, 记为

$$T\mathbf{x} = \mathbf{y}$$

称 \mathbf{y} 为 \mathbf{x} 在 T 下的象, \mathbf{x} 为 \mathbf{y} 的原象.

警告:

- T 是 \mathbb{V} 到自身的一个映射: 即变换前后都在线性空间 \mathbb{V} 中, **如果变换前后不在同一线性空间中?**
- 定义中说明了 \mathbb{V} 中的元素为向量, 实际上 \mathbb{V} 中元素不一定是向量.

Definition 11 (线性变换) 设 T 是数域 K 上线性空间 \mathbb{V} 中的一个变换, 对于 $\forall k, l \in K$, 若满足

$$T(k\mathbf{x} \oplus l\mathbf{y}) = kT(\mathbf{x}) \oplus lT(\mathbf{y})$$

则称 T 是 \mathbb{V} 的线性变换.

2.5.1.3.1.2 线性变换的特性

若线性空间 \mathbb{V} 中的 x_1, x_2, \dots, x_m 线性相关, 则线性变换 T 对其变换后仍线性相关.

即若存在不全为零的数 $c_1, c_2, \dots, c_m \in K$, 使得

$$c_1 \odot x_1 \oplus c_2 \odot x_2 \oplus \dots \oplus c_m \odot x_m = \mathbf{0}$$

成立, 则亦有下式成立

$$c_1 \odot (Tx_1) \oplus c_2 \odot (Tx_2) \oplus \dots \oplus c_m \odot (Tx_m) = T\mathbf{0} = \mathbf{0}$$

2.5.1.3.2 线性变换的运算

涉线性变换的加法, 数乘, 乘法, 逆与幂, 以及单位变换, 零变换, 负变换, 逆变换, 多项式. 有关对称变换, 正交变换,酉变换参见[特殊线性空间](#)(页 46)小节

提示: 可以证明上述变换均为线性变换!

2.5.1.3.2.1 基本线性变换

2.5.1.3.2.2 单位变换

将线性空间 \mathbb{V} 中的任一元素 x 变为自身的变换称为 **单位变换**, 即

$$(T_e)x = x, (\forall x \in \mathbb{V})$$

2.5.1.3.2.3 零变换

将线性空间 \mathbb{V} 中的任一元素 x 变为零元素 $\mathbf{0}$ 的变换称为 **零变换**, 即

$$(T_0)x = \mathbf{0}, (\forall x \in \mathbb{V})$$

2.5.1.3.2.4 负变换

设 T 是线性空间 \mathbb{V} 的线性变换, 线性变换 T 的 **负变换**定义为

$$(-T)x = -(Tx), (\forall x \in \mathbb{V})$$

2.5.1.3.2.5 逆变换

设 T, T^{-1} 是线性空间 \mathbb{V} 的线性变换, 若满足下述条件, 则称线性变换 T 与 T^{-1} 互为 **逆变换** (Inverse transformation), 即

$$(TT^{-1})\mathbf{x} = (T^{-1}T)\mathbf{x} = \mathbf{x}, (\forall \mathbf{x} \in \mathbb{V})$$

亦即:

$$(TT^{-1}) = (T^{-1}T) = T_e$$

2.5.1.3.2.6 加法

2.5.1.3.2.7 定义

设 T_1, T_2 是线性空间 \mathbb{V} 的两个线性变换, 线性变换的 **加法** 定义为

$$(T_1 + T_2)\mathbf{x} = T_1\mathbf{x} \oplus T_2\mathbf{x}, (\forall \mathbf{x} \in \mathbb{V})$$

2.5.1.3.2.8 性质

1. 交换律: $T_1 + T_2 = T_2 + T_1$
2. 结合律: $(T_1 + T_2) + T_3 = T_1 + (T_2 + T_3)$
3. 有零元: $T + T_0 = T$
4. 有负元: $T + (-T) = T_0$

提示: 对照线性空间定义中的加法的性质.

2.5.1.3.2.9 数乘

2.5.1.3.2.10 定义

设 $k \in K, T$ 为线性空间 \mathbb{V} 的线性变换, 定义数 k 与变换 T 的乘积 (**数乘**) kT 为

$$(kT)\mathbf{x} = k \odot (T\mathbf{x}), (\forall \mathbf{x} \in \mathbb{V})$$

2.5.1.3.2.11 性质

1. 乘 1 律: $1T = T$
2. 数因子分配律: $k(T_1 + T_2) = kT_1 + kT_2$
3. 数乘分配率: $(k + l)T = kT + lT$
4. 数乘结合率: $(kl)T = k(lT)$

提示: 对照线性空间定义中的数乘的性质.

2.5.1.3.2.12 乘法

2.5.1.3.2.13 定义

设 T_1, T_2 是线性空间 \mathbb{V} 的两个线性变换, 定义 T_1 与 T_2 的 **乘积** T_1T_2 为

$$(T_1T_2)\mathbf{x} = T_1(T_2\mathbf{x}), (\forall \mathbf{x} \in \mathbb{V})$$

2.5.1.3.2.14 性质

1. 结合律: $(T_1T_2)T_3 = T_1(T_2T_3)$
2. 分配律 1: $T_1(T_2 + T_3) = T_1T_2 + T_1T_3$
3. 分配率 2: $(T_1 + T_2)T_3 = T_1T_3 + T_2T_3$

2.5.1.3.2.15 幂

2.5.1.3.2.16 定义

设 n 是正整数, T 是线性空间 \mathbb{V} 中的线性变换, 定义其 n 次幂为

$$T^n = T^{n-1}T, (n = 2, 3, \dots)$$

定义 T 的零次幂为:

$$T^0 = T_e$$

2.5.1.3.2.17 性质

1. 性质 1: $T^{m+n} = T^m T^n$
2. 性质 2: $(T^m)^n = T^{mn}$
3. 性质 3: $T^{(-n)} = (T^{-1})^n$

2.5.1.3.2.18 线性变换的多项式

2.5.1.3.2.19 定义

数量 t 的 m 次多项式可以定义为

$$f(t) = a_m t^m + a_{m-1} t^{m-1} + \cdots + a_1 t + a_0$$

类似地, 线性变换 T 的 m 次多项式可以定义为

$$f(T) = a_m T^m + a_{m-1} T^{m-1} + \cdots + a_1 T + a_0$$

2.5.1.3.2.20 性质

1. 性质 1: 若 $h(t) = f(t)g(t)$, 则 $h(T) = f(T)g(T)$
2. 性质 2: 若 $p(t) = f(t) + g(t)$, 则 $p(T) = f(T) + g(T)$
3. 性质 3: $f(T)g(T) = g(T)f(T)$ (同一线性变换的多项式相乘可交换)

2.5.1.3.3 线性变换与矩阵

2.5.1.3.3.1 线性变换的矩阵表示

设 T 是线性空间 \mathbb{V} 中的线性变换, $\mathbf{x} \in \mathbb{V}^n$, 且 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 是 \mathbb{V}^n 的一个基 (原象基组), 则

$$\mathbf{x} = a_1 \odot \mathbf{x}_1 \oplus a_2 \odot \mathbf{x}_2 \oplus \cdots \oplus a_n \odot \mathbf{x}_n$$

$$T\mathbf{x} = a_1 \odot (T\mathbf{x}_1) \oplus a_2 \odot (T\mathbf{x}_2) \oplus \cdots \oplus a_n \odot (T\mathbf{x}_n)$$

故 \mathbf{x} 在变换 T 下得到的象 $T\mathbf{x}$ 可以由基象组线性表示, 而基像组仍在线性空间 \mathbb{V} 中, 故可用原象组线性表示为

$$\left\{ \begin{array}{l} T\mathbf{x}_1 = a_{11} \odot \mathbf{x}_1 \oplus a_{21} \odot \mathbf{x}_2 \oplus \cdots \oplus a_{n1} \odot \mathbf{x}_n \\ T\mathbf{x}_2 = a_{12} \odot \mathbf{x}_1 \oplus a_{22} \odot \mathbf{x}_2 \oplus \cdots \oplus a_{n2} \odot \mathbf{x}_n \\ \vdots \\ T\mathbf{x}_n = a_{1n} \odot \mathbf{x}_1 \oplus a_{2n} \odot \mathbf{x}_2 \oplus \cdots \oplus a_{nn} \odot \mathbf{x}_n \end{array} \right.$$

将上述方程组矩阵化可得, 线性变换 T 在基 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 下的矩阵表示为

$$T(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (T\mathbf{x}_1, T\mathbf{x}_2, \dots, T\mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) A$$

其中

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

矩阵 \mathbf{A} 的第 i 列为 $T\mathbf{x}_i$ 的坐标. 当 \oplus 表示普通向量加法, \odot 表示普通数与向量乘法时, 上式退化为普通矩阵乘.

注解: 举个例子, 定义线性空间 $\mathbb{R}^{2 \times 2}$ 中的线性变换

$$T\mathbf{X} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{X}$$

求线性变换 T 在基 $\mathbf{E}_{11}, \mathbf{E}_{12}, \mathbf{E}_{21}, \mathbf{E}_{22}$ 下的矩阵.

解: 由题意有

$$\left\{ \begin{array}{l} T\mathbf{E}_{11} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} a & 0 \\ c & 0 \end{bmatrix} = a\mathbf{E}_{11} + c\mathbf{E}_{21} \\ T\mathbf{E}_{12} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & a \\ 0 & c \end{bmatrix} = a\mathbf{E}_{12} + c\mathbf{E}_{22} \\ T\mathbf{E}_{21} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} b & 0 \\ d & 0 \end{bmatrix} = b\mathbf{E}_{11} + d\mathbf{E}_{21} \\ T\mathbf{E}_{22} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & b \\ 0 & d \end{bmatrix} = b\mathbf{E}_{12} + d\mathbf{E}_{22} \end{array} \right.$$

由上式得到 T 在所给基下的矩阵为

$$\mathbf{A} = \begin{bmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ c & 0 & d & 0 \\ 0 & c & 0 & d \end{bmatrix}$$

满足 $T(\mathbf{E}_{11}, \mathbf{E}_{12}, \mathbf{E}_{21}, \mathbf{E}_{22}) = (\mathbf{E}_{11}, \mathbf{E}_{12}, \mathbf{E}_{21}, \mathbf{E}_{22})\mathbf{A}$

提示:

1. 零变换的矩阵为零矩阵: \mathbf{O}
2. 单位变换的矩阵为单位矩阵: \mathbf{I}
3. 数乘变换的矩阵为数量矩阵: $m\mathbf{I}$

2.5.1.3.3.2 性质

1. $(T_1 + T_2)(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)(\mathbf{A} + \mathbf{B})$
2. $(kT_1)(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)(k\mathbf{A})$
3. $(T_1T_2)(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{AB}$
4. $(T_1^{-1})(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{A}^{-1}$

2.5.1.3.3.3 像与原像的坐标转换

设有线性空间 \mathbb{V}^n 中的元素 \mathbf{x} , 线性变换 T , 一组基 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$; \mathbf{x} 在该基下的坐标为 $(\xi_1, \xi_2, \dots, \xi_n)^T$, T 在该基下的矩阵为 \mathbf{A} ; 则 $T\mathbf{x}$ 在该基下的坐标为

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} = \mathbf{A} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix}$$

提示:

$$T\mathbf{x} = T(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{A} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix}$$

2.5.1.3.3.4 线性变换在不同基下的矩阵转换

设有线性空间 \mathbb{V}^n 中的线性变换 T , 在基 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 和基 $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$ 下的矩阵分别为 \mathbf{A}, \mathbf{B} , 且由基 1 到基 2 的过度矩阵为 \mathbf{C} , 则有

$$\mathbf{B} = \mathbf{C}^{-1}\mathbf{AC}$$

提示:

$$\begin{aligned} T\mathbf{x} &= T(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{A} \\ T\mathbf{y} &= T(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)\mathbf{B} \\ T\mathbf{y} &= T(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) = T(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{C} \\ &= (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\mathbf{AC} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)\mathbf{C}^{-1}\mathbf{AC} \end{aligned}$$

2.5.1.3.3.5 相似矩阵

2.5.1.3.3.6 什么是相似矩阵

由线性变换在不同基下的矩阵关系引出矩阵相似定义:

Definition 12 (相似矩阵) 设有矩阵 A, B , 若存在 非奇异矩阵 P 使得 $B = P^{-1}AP$, 则称 A 相似于 B , 记作 $A \sim B$.

警告: 矩阵合同是指 $B = P^TAP$

若 $B = P^{-1}AP$, 且 $f(t)$ 是数域 K 上的多项式, 则有

$$f(B) = P^{-1}f(A)P$$

2.5.1.3.3.7 性质

1. 反身性: $A \sim A$
2. 对称性: 若 $A \sim B$, 则 $B \sim A$
3. 传递性: 若 $A \sim B, B \sim C$, 则 $A \sim C$

2.5.1.3.3.8 线性变换多项式的矩阵

数量 t 的 m 次多项式可以定义为

$$f(t) = a_m t^m + a_{m-1} t^{m-1} + \cdots + a_1 t + a_0$$

类似地, 线性空间 \mathbb{V}^n 中, 线性变换 T 的 m 次多项式可以定义为

$$f(T) = a_m T^m + a_{m-1} T^{m-1} + \cdots + a_1 T + a_0$$

设线性空间 \mathbb{V}^n 中, 线性变换 T 在基 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 下的矩阵为 A , 即

$$T(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)A$$

则 $f(T)$ 在该基下的矩阵为

$$f(A) = a_m A^m + a_{m-1} A^{m-1} + \cdots + a_1 A + a_0 I$$

上式也称为方阵 A 的多项式.

2.5.1.3.3.9 常见线性变换

伸缩, 旋转, 对称变换举例

- 伸缩: $\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} kx_1 \\ kx_2 \end{bmatrix}$, $\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} k_1 x_1 \\ k_2 x_2 \end{bmatrix}$
- 逆时针旋转: $\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta - x_2 \sin \theta \\ x_1 \sin \theta + x_2 \cos \theta \end{bmatrix}$
- 顺时针旋转: $\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta + x_2 \sin \theta \\ -x_1 \sin \theta + x_2 \cos \theta \end{bmatrix}$
- 轴对称: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -x_2 \end{bmatrix}$, $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -x_1 \\ x_2 \end{bmatrix}$

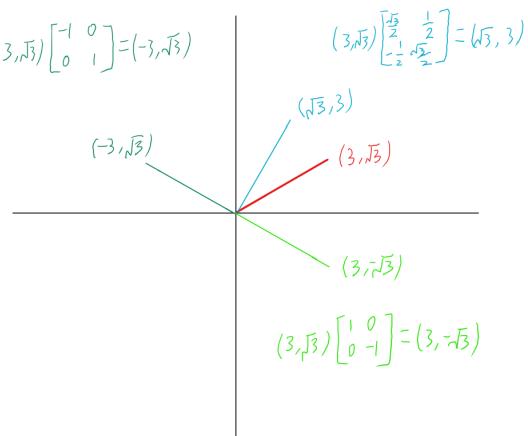


图 2.3: 变换举例
变换举例, 伸缩, 旋转, 对称变换

提示: 记忆旋转变换

- 逆时针旋转: 第一象限一个向量, 逆时针旋转一个小角度, x 坐标减小, y 坐标增大
- 顺时针旋转: 第一象限一个向量, 逆时针旋转一个小角度, x 坐标增大, y 坐标减小

下面借助 Python 工具实现旋转变换的验证:

1. 产生二维平面内椭圆内的随机点构成椭圆面 S ;
2. 将椭圆面 S 分别进行逆时针与顺时针旋转;
3. 绘制旋转前后的图像.

结果如下图

实现代码, 参见文件 [demo_rotation_ellipse.py](#).

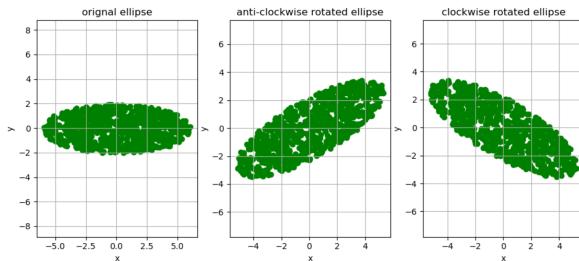


图 2.4: 旋转变换举例.

旋转变换举例: 原始图 (左), 逆时针旋转 30 度 (中), 顺时针旋转 30 度 (右).

代码 2.1: demo_rotation_ellipse.py

```

1 import pytool
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # =====generate Ellipse=====
6 a = 6 # major axis
7 b = 2 # minor axis
8 x0 = 10 # center x0
9 y0 = 10 # center y0
10 N = 1000 # number of points
11
12 # angle for rotating ellipse data
13 theta = np.pi * 30 / 180
14
15 x, y = pytool.ellipse_surface(a, b, x0, y0, N, 'rand')
16
17 x = x - np.mean(x)
18 y = y - np.mean(y)
19
20 xy = np.array([x, y])
21 print(xy.shape)
22
23 A = xy
24
25 # =====rotate=====
26 # -----rotating matrix(Anti-clockwise)-----
27 M1 = [[np.cos(theta), -np.sin(theta)],
28        [np.sin(theta), np.cos(theta)]]
29 print("M1", M1)
30
31 # -----rotating-----
32 A1 = np.dot(M1, A)
33
34 x1 = A1[0]

```

(下页继续)

(续上页)

```

35 y1 = A1[1]
36
37 # -----rotating matrix(clockwise)-----
38 M2 = [[np.cos(theta), np.sin(theta)],
39        [-np.sin(theta), np.cos(theta)]]
40 print("M2", M2)
41
42 # -----
43 A2 = np.dot(M2, A)
44
45 x2 = A2[0]
46 y2 = A2[1]
47
48 # =====show data=====
49 xmin = np.min([x, x1, x2])
50 xmax = np.max([x, x1, x2])
51 ymin = np.min([y, y1, y2])
52 ymax = np.max([y, y1, y2])
53 plt.figure()
54 plt.subplot(131)
55 plt.scatter(x, y, c='g', marker='o')
56 plt.grid()
57 plt.axis([xmin, xmax, ymin, ymax])
58 # plt.axis('equal')
59 plt.xlabel('x')
60 plt.ylabel('y')
61 plt.title('orignal ellipse')
62 plt.subplot(132)
63 plt.scatter(x1, y1, c='g', marker='o')
64 plt.grid()
65 plt.axis([xmin, xmax, ymin, ymax])
66 # plt.axis('equal')
67 plt.xlabel('x')
68 plt.ylabel('y')
69 plt.title('anti-clockwise rotated ellipse')
70 plt.subplot(133)
71 plt.scatter(x2, y2, c='g', marker='o')
72 plt.grid()
73 plt.axis([xmin, xmax, ymin, ymax])
74 # plt.axis('equal')
75 plt.xlabel('x')
76 plt.ylabel('y')
77 plt.title('clockwise rotated ellipse')
78 plt.show()

```

2.5.1.3.4 总结

- 线性变换对向量组相关性的影响 - 线性相关向量组 \rightarrow 线性变换 \rightarrow 线性相关 - 线性无关向量组 \rightarrow 线性变换 \rightarrow 线性无关或线性相关 (如零变换)
- 线性变换 T 与其对应矩阵 A - 它们的值域、核、秩、亏相同

2.5.1.4 特征值与特征向量

如何选择线性空间的基,使得线性变换在该基下的矩阵表示最为简单. 涉及特征值 (eigenvalue) 与特征向量 (eigenvector)

2.5.1.4.1 特征值与特征向量

2.5.1.4.1.1 线性变换的特征值与特征向量

Definition 13 (线性变换的特征值与特征向量) 设 T 是数域 \mathbb{K} 上的线性空间 \mathbb{V}^n 上的线性变换, 且对于数域 \mathbb{K} 中的数 λ , 若存在非零向量 $x \in \mathbb{V}^n$ 满足如下条件

$$Tx = \lambda x$$

则称 λ 为 T 的 **特征值**, x 为 T 的属于特征值 λ 的特征向量.

注解: 由上述定义知:

- 线性变换不改变特征向量的方向
 - 特征值被特征向量唯一确定, 特征向量不被特征值唯一确定 ($T(kx) = k\lambda x$)
-

2.5.1.4.1.2 矩阵的特征值与特征向量

Definition 14 (矩阵的特征值与特征向量) 设有矩阵 A , 且对于数域 \mathbb{K} 中的数 λ , 若存在非零向量 $x \in \mathbb{V}^n$ 满足如下条件

$$Ax = \lambda x$$

则称 λ 为矩阵 A 的 **特征值**, x 为 A 的属于特征值 λ 的特征向量.

2.5.1.4.1.3 特征值与特征向量的关系

注解:

- 特征值被特征向量唯一确定, 特征向量不被特征值唯一确定 ($T(kx) = k\lambda x$)
 - 互不相同的特征值对应的特征向量线性无关
-

2.5.1.4.1.4 实例

编程生成二维平面 (x, y) 中的点, 这些点落在一个椭圆内, 长轴为 6, 短轴为 2, 如 图 2.5 所示, 其主要的方向有两个, 长轴方向 (红色), 短轴方向 (蓝).

对这些二维数据做主成分分析 (*Principal Component Analysis, PCA*), 可得两个主要方向:

1. 零均值化数据
2. 计算数据的协方差 (即两个维度间的)
3. 计算协方差矩阵的特征值与特征向量

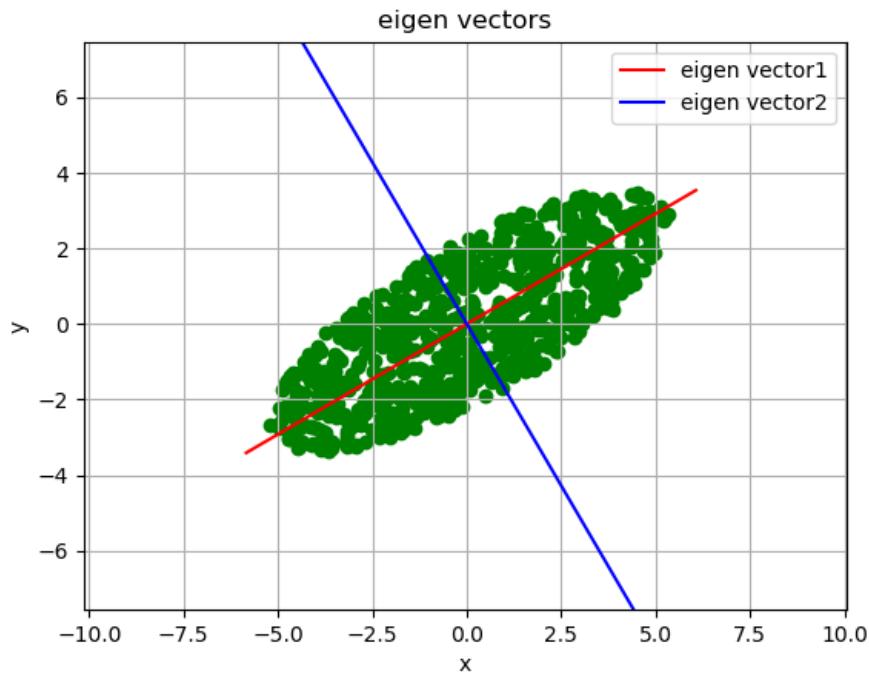


图 2.5: 椭圆内点的特征向量可视化

对椭圆内的点进行特征分解, 得到两个特征方向, 并进行特征向量可视化, 长轴 (红), 短轴 (蓝).

警告: 注意这里是对数据的协方差矩阵作特征值分解, 是否可以不求协方差矩阵直接对原始数据矩阵进行分析得到主方向. 特征值特征向量是针对变换矩阵而言的, 对数据矩阵是否有相关理论.

实现代码, 参见文件 [demo_eigen_ellipse.py](#).

代码 2.2: [demo_eigen_ellipse.py](#)

```

1 import pytool
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # =====generate Ellipse=====

```

(下页继续)

(续上页)

```

6  a = 6  # major axis
7  b = 2  # minor axis
8  x0 = 10 # center x0
9  y0 = 10 # center y0
10 N = 1000 # number of points
11
12 # angle for rotating ellipse data
13 theta = np.pi * 30 / 180
14
15 x, y = pytool.ellipse_surface(a, b, x0, y0, N, 'rand')
16 # x, y = pytool.ellipse_surface(a, b, x0, y0, N, 'order')
17
18
19 x = x - np.mean(x)
20 y = y - np.mean(y)
21
22 xse = [min(x), max(x)]
23 yse = [min(y), max(y)]
24
25
26 xy = np.array([x, y])
27 print(xy.shape)
28
29 A = xy
30
31 # ======rotate=====
32 # -----rotating matrix(Anti-clockwise)-----
33 M = [[np.cos(theta), np.sin(theta)],
34       [-np.sin(theta), np.cos(theta)]]
35 M = [[np.cos(theta), -np.sin(theta)],
36       [np.sin(theta), np.cos(theta)]]
37 print("rotating matrix M: ", M)
38
39 # -----rotating-----
40 A = np.dot(M, A)
41
42 x = A[0]
43 y = A[1]
44
45 # ======show data=====
46 plt.figure()
47 plt.scatter(x, y, c='g', marker='o')
48 plt.grid()
49
50 # ======Eigenvalue decomposition=====
51
52 AAT = np.dot(A, A.transpose())

```

(下页继续)

(续上页)

```

53 ATA = np.dot(A.transpose(), A)
54
55 values, evectors = np.linalg.eig(AAT)
56 # values, evectors = np.linalg.eig(ATA)
57 print(A.shape, AAT.shape, values.shape, evectors.shape)
58 print(AAT)
59 print("values: ", values, "evectors: ", evectors)
60 angle1 = np.arctan(evectors[1, 0] / evectors[0, 0]) * 180 / np.pi
61 angle2 = np.arctan(evectors[1, 1] / evectors[0, 1]) * 180 / np.pi
62 print(angle1, angle2)
63
64
65 # =====plot eigen vector=====
66 colorlines = ['-r', '-b']
67 pytool.plot_vectors2d(evectors.transpose(), xse=xse, yse=yse,
68                      nPoints=N, title='eigen vectors', colorlines=colorlines)
69 plt.legend(['eigen vector1', 'eigen vector2'])
70 plt.show()
71
72
73 plt.figure()
74 plt.imshow(AAT)
75 plt.show()

```

2.5.1.4.2 广义特征向量

2.5.1.4.2.1 概念与内涵

一般, 若 λ^* 是矩阵 A 的 k 重特征值, 则对应的线性无关的特征向量 x_1, x_2, \dots, x_k 可由下列方程组求出

$$(\lambda^* I - A)x_i = -x_{i-1}$$

其中, $i = 1, 2, \dots, k$, 且规定 $x_0 = \mathbf{0}$, 将求得的 k 个向量 x_1, x_2, \dots, x_k 称为矩阵 A 的广义特征向量.

2.5.1.4.2.2 计算步骤

1. 写出特征多项式矩阵 $\lambda I - A$
2. 令特征多项式矩阵对应的行列式的值为 $|\lambda I - A| = 0$
3. 求解得特征值 λ
4. 对于单重特征值: 代入方程组 $(\lambda I - A)x = \mathbf{0}$, 求解特征向量.
5. 对于多重特征值: 代入方程组 $(\lambda^* I - A)x_i = -x_{i-1}$, 求解广义特征向量.

2.5.1.4.3 例子 (广义特征向量)

注解: 求如下矩阵的特征值与特征向量/广义特征向量

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

解:

1. 特征多项式矩阵为

$$\lambda\mathbf{I} - \mathbf{A} = \begin{bmatrix} \lambda & -2 & -2 \\ -2 & \lambda - 1 & -2 \\ 0 & -2 & \lambda - 1 \end{bmatrix}$$

可使用初等变换稍微化简, 方便求行列式的值 $|\lambda\mathbf{I} - \mathbf{A}| = (\lambda + 1)^2(\lambda - 4)$

2. 求解 $|\lambda\mathbf{I} - \mathbf{A}| = (\lambda + 1)^2(\lambda - 4) = 0$ 的解为 $\lambda_1 = 4, \lambda_2 = \lambda_3 = -1$
3. 由 $(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{x}_1 = \mathbf{0}$ 求解出对应于 λ_1 的特征向量 $\mathbf{x}_1 = (5, 6, 4)^T$, 由 $(\lambda_2\mathbf{I} - \mathbf{A})\mathbf{x}_2 = \mathbf{0}, (\lambda_3\mathbf{I} - \mathbf{A})\mathbf{x}_3 = -\mathbf{x}_2$ 求解广义特征向量 $\mathbf{x}_2 = (0, 1, -1)^T, \mathbf{x}_3 = (1, -1, 1/2)^T$

2.5.1.4.4 特征多项式与最小多项式

2.5.1.4.4.1 特征多项式

设 T 是 \mathbb{V}^n 中的线性变换, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 是线性空间 \mathbb{V}^n 中的基, T 在该基下的矩阵为 \mathbf{A} , 再设 $\mathbf{x} = \xi_1\mathbf{x}_1 + \xi_2\mathbf{x}_2 + \dots + \xi_n\mathbf{x}_n$ 为 T 的属于特征值 λ 的特征向量, 则由 $T\mathbf{x} = \lambda\mathbf{x}$ 知

$$\mathbf{A} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = \lambda \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix}$$

从而有

$$(\lambda\mathbf{I} - \mathbf{A}) \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = \mathbf{0}$$

由 $\mathbf{x} \neq \mathbf{0}$ 知, $\xi_1, \xi_2, \dots, \xi_n$ 不全为零, 从而方程组有非零解, 则

$$\varphi(\lambda) = \det(\lambda\mathbf{I} - \mathbf{A}) = |\lambda\mathbf{I} - \mathbf{A}| = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n)$$

Definition 15 (特征多项式) 设有数域 \mathbb{K} 上的 n 阶方阵 \mathbf{A} , 变量 λ , 矩阵 \mathbf{A} 的特征矩阵 $\lambda\mathbf{I} - \mathbf{A}$ 的行列式 $\det(\lambda\mathbf{I} - \mathbf{A})$ 称为矩阵 \mathbf{A} 的特征多项式. 记为 $\varphi(\lambda)$, 其根 λ_i 为 \mathbf{A} 的特征值, 非零解向量: $(\xi_1, \xi_2, \dots, \xi_n)^T$ 为 \mathbf{A} 的属于特征值 λ_i 的特征向量.

注解: 线性变换 T 的特征值与特征向量, 与 T 的矩阵 A 的特征值与特征向量一一对应:

- T 与 A 的特征值一致
- T 的特征向量在基下的坐标与 A 的特征向量一致
- 线性变换 T 的矩阵 A 的特征多项式与基的选择无关

即, 若 A 为 T 在基 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 下的矩阵, $(\xi_1, \xi_2, \dots, \xi_n)^T$ 为矩阵 A 的属于特征值 λ 的特征向量, 则

- λ 是 T 的特征值
- $\mathbf{x} = \xi_1 \mathbf{x}_1 + \xi_2 \mathbf{x}_2 + \dots + \xi_n \mathbf{x}_n$ 是 T 的属于特征值 λ 的特征向量
- 线性变换 T 的矩阵 A 的特征多项式与基的选择无关

Theorem 16 (Hamilton-Cayley) n 阶矩阵 A 是其特征多项式 $\varphi(\lambda)$ 的矩阵根. 即若有

$$\varphi(\lambda) = \det(\lambda I - A) = \lambda^n + a_1 \lambda^{n-1} + \dots + a_{n-1} \lambda + a_n$$

则

$$\varphi(A) = A^n + a_1 A^{n-1} + \dots + a_{n-1} A + a_n I = O.$$

2.5.1.4.4.2 最小多项式

Definition 17 (最小多项式) 定义: 首项系数是 1, 次数最小, 且以矩阵 A 为根的 λ 的多项式, 称为 A 的 **最小多项式** (*Minimal Polynomial*), 常用 $m(\lambda)$ 表示.

2.5.1.4.5 特征子空间与不变子空间

2.5.1.4.5.1 什么是特征子空间

Definition 18 (特征子空间) 定义: 设 T 是线性空间 V^n 的线性变换, λ 是 T 的一个特征值, 称 V^n 的子空间 V_λ^n 是 T 的属于 λ 的 **特征子空间** (*Characteristic Subspace*), 其中

$$V_\lambda^n = \{x | Tx = \lambda x, x \in V^n\}.$$

提示:

1. 特征子空间是线性子空间
2. 特征子空间是由特征向量加上零向量构成的子空间.

2.5.1.4.5.2 什么是不变子空间

Definition 19 (不变子空间) 若 T 是线性空间 \mathbb{V} 的线性变换, \mathbb{V}_1 是 \mathbb{V} 的子空间, 且 $\forall \mathbf{x} \in \mathbb{V}_1$, 有 $T\mathbf{x} \in \mathbb{V}_1$, 则称 \mathbb{V}_1 是 T 的 不变子空间 (Invariant Subspace).

2.5.1.4.6 信号子空间与噪声子空间

在数字信号处理领域经常定义 信号子空间 (Signal Subspace) 与 噪声子空间 (Noise Subspace)

2.5.1.5 矩阵对角化

2.5.1.5.1 多项式矩阵及其标准形

2.5.1.5.1.1 多项式矩阵

Definition 20 (多项式矩阵) 称如下矩阵为 多项式矩阵 :

$$\mathbf{A}(\lambda) = \begin{bmatrix} a_{11}(\lambda) & a_{12}(\lambda) & \cdots & a_{1n}(\lambda) \\ a_{21}(\lambda) & a_{22}(\lambda) & \cdots & a_{2n}(\lambda) \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}(\lambda) & a_{n2}(\lambda) & \cdots & a_{nn}(\lambda) \end{bmatrix}$$

其中, $a_{ij}(\lambda)$ 为数域 K 上的 λ 的多项式. 易知数域上的 n 阶矩阵 \mathbf{A} 的特征矩阵是一个 多项式矩阵 (Polynomial Matrix).

2.5.1.5.1.2 首一多项式

定义: 最高次数项为首项, 且首项系数是 1 的多项式为 首 1 多项式.

如: $\lambda^2 + 2\lambda$ 是首 1 多项式, $2\lambda^2 + \lambda$ 不是首一多项式.

注解: 定理: 矩阵 \mathbf{A} 的最小多项式 $m(\lambda)$ 可以整除以 \mathbf{A} 为根的任意首 1 多项式 $\phi(\lambda)$, 即 $m(\lambda)|\phi(\lambda)$ 定理: 矩阵 \mathbf{A} 的最小多项式 $m(\lambda)$ 与其特征多项式 $\varphi(\lambda)$ 具有相同的零点 (不计重数).

2.5.1.5.1.3 多项式矩阵的标准形

多项式矩阵的标准形是指具有如下形式的多项式矩阵:

$$\mathbf{A}(\lambda) = \begin{bmatrix} d_1(\lambda) & & & & \\ & d_2(\lambda) & & & \\ & & \ddots & & \\ & & & d_s(\lambda) & \\ & & & & 0 \\ & & & & & \ddots \\ & & & & & & 0 \end{bmatrix}$$

其中, $d_i(\lambda)$ 是首一多项式, $d_{i-1}(\lambda)$ 可以整除 $d_i(\lambda)$, 记为 $d_{i-1}(\lambda)|d_i(\lambda)$, $i = 1, 2, \dots, s, s \leq n$.

2.5.1.5.2 不变因子与初等因子

对于多项式矩阵 $\mathbf{A}(\lambda)$, 采用初等变换 (行列均可) 可以化为标准形.

2.5.1.5.2.1 不变因子

多项式矩阵 $\mathbf{A}(\lambda)$ 的标准形式的对角线上的非零元素 $d_i(\lambda), (i = 1, 2, \dots, s)$ 不随初等变换而改变, 称之为多项式矩阵 $\mathbf{A}(\lambda)$ 的 **不变因子**或**不变因式**, 可由下式计算:

$$d_i(\lambda) = \frac{D_i(\lambda)}{D_{i-1}(\lambda)}, D_0(\lambda) = 1 (i = 1, 2, \dots, s)$$

其中, $D_i(\lambda)$ 为多项式矩阵 $\mathbf{A}(\lambda)$ 的所有 i 阶子式 (行列式) 的最大公因式, 也称为多项式矩阵 $\mathbf{A}(\lambda)$ 的 i 阶行列式因子, 不随初等变换而改变.

2.5.1.5.2.2 初等因子

把多项式矩阵 $\mathbf{A}(\lambda)$ 的每个次数大于零的不变因子 $d_i(\lambda)$ 分解为不可约因式的乘积, 这些不可约因式及它们的幂指数称为多项式矩阵 $\mathbf{A}(\lambda)$ 的一个**初等因子**, 初等因子的全体称为多项式矩阵 $\mathbf{A}(\lambda)$ 的**初等因子组**.

2.5.1.5.2.3 举例

举例如下, 用以说明以上概念, 通过初等变换将多项式矩阵 $\mathbf{A}(\lambda)$ 化为标准形:

$$\begin{aligned} \mathbf{A}(\lambda) &= \begin{bmatrix} -\lambda + 1 & 2\lambda - 1 & \lambda \\ \lambda & \lambda^2 & -\lambda \\ \lambda^2 + 1 & \lambda^2 + \lambda - 1 & -\lambda^2 \end{bmatrix} \xrightarrow{c_1+c_3} \begin{bmatrix} 1 & 2\lambda - 1 & \lambda \\ 0 & \lambda^2 & -\lambda \\ 1 & \lambda^2 + \lambda - 1 & -\lambda^2 \end{bmatrix} \\ &\xrightarrow{r_3-r_1} \begin{bmatrix} 1 & 2\lambda - 1 & \lambda \\ 0 & \lambda^2 & -\lambda \\ 0 & \lambda^2 - \lambda & -\lambda^2 - \lambda \end{bmatrix} \xrightarrow{\substack{c_2+(1-2\lambda)c_1 \\ c_3+(-\lambda)c_1}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda^2 & -\lambda \\ 0 & \lambda^2 - \lambda & -\lambda^2 - \lambda \end{bmatrix} \\ &\xrightarrow{\substack{c_2+\lambda c_3 \\ c_2 \leftrightarrow c_3}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\lambda & 0 \\ 0 & -\lambda^2 - \lambda & -\lambda^3 - \lambda \end{bmatrix} \xrightarrow{r_3+(-\lambda-1)r_2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\lambda & 0 \\ 0 & 0 & -\lambda^3 - \lambda \end{bmatrix} \\ &\xrightarrow{-r_2,-r_3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda^3 + \lambda \end{bmatrix} \end{aligned}$$

易知, $d_1(\lambda) = 1$, $d_2(\lambda) = \lambda$, $d_3(\lambda) = \lambda^3 + \lambda$ 为 $\mathbf{A}(\lambda)$ 的不变因子.

大于零的不变因子有 $d_2(\lambda) = \lambda$, $d_3(\lambda) = \lambda^3 + \lambda$;

对于实数域, 它们分别可以分解为 λ , $\lambda(\lambda^2 + 1)$, 所以初等因子组为 $\lambda, \lambda, \lambda^2 + 1$; 对于复数域, 它们分别可以分解为 $\lambda, \lambda(\lambda + j)(\lambda - j)$, 所以初等因子组为 $\lambda, \lambda, \lambda + j, \lambda - j$.

2.5.1.5.3 Jordan 标准型

对于 n 阶矩阵 \mathbf{A} , 存在可逆矩阵 \mathbf{P} , 使得

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{J}.$$

2.5.1.5.3.1 概念与内涵

Jordan 标准型是指具有如下形式的矩阵:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1(\lambda_1) & & & \\ & \mathbf{J}_2(\lambda_2) & & \\ & & \ddots & \\ & & & \mathbf{J}_s(\lambda_s) \end{bmatrix}_{n \times n},$$

其中, 每个 Jordan 标准形具有如下形式:

$$\mathbf{J}_i(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}_{m_i \times m_i}.$$

Jordan 标准形的对角元素为矩阵 \mathbf{A} 的特征值.

2.5.1.5.3.2 求解方法

在复数域 \mathbb{C} 上求解 n 阶矩阵 \mathbf{A} 的 Jordan 标准形的步骤如下:

1. 求特征矩阵的初等因子组, 记为 $(\lambda - \lambda_1)^{m_1}, (\lambda - \lambda_2)^{m_2}, \dots, (\lambda - \lambda_s)^{m_s}$, 其中 λ_i 为矩阵 \mathbf{A} 的特征值, 且可以相同, m_i 为对于特征值 λ_i 的特征值的重数, 也可相同;
2. 写出每个初等因子 $(\lambda - \lambda_i)^{m_i}$, $i = 1, 2, \dots, s$ 对应的 Jordan 块;
3. 写出以这些 Jordan 块构成的 Jordan 标准形.

2.5.1.5.3.3 举例

求如下矩阵的 Jordan 标准形:

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

1. 写出特征多项式矩阵，并化简为标准形：

$$\begin{aligned}
 & \left[\begin{array}{ccc} \lambda & -2 & -2 \\ -2 & \lambda - 1 & -2 \\ 0 & -2 & \lambda - 1 \end{array} \right] \xrightarrow{2r_1} \left[\begin{array}{ccc} 2\lambda & -4 & -4 \\ -2 & \lambda - 1 & -2 \\ 0 & -2 & \lambda - 1 \end{array} \right] \\
 & \xrightarrow{r_1 + \lambda r_2} \left[\begin{array}{ccc} 0 & \lambda(\lambda - 1) - 4 & -2\lambda - 4 \\ -2 & \lambda - 1 & -2 \\ 0 & -2 & \lambda - 1 \end{array} \right] \\
 & \rightarrow \left[\begin{array}{ccc} 0 & \lambda(\lambda - 1) - 4 & -2\lambda - 4 \\ 1 & 0 & 0 \\ 0 & -2 & \lambda - 1 \end{array} \right] \\
 & \xrightarrow{2c_3} \left[\begin{array}{ccc} 0 & \lambda(\lambda - 1) - 4 & -4\lambda - 8 \\ 1 & 0 & 0 \\ 0 & -2 & 2(\lambda - 1) \end{array} \right] \\
 & c_3 + (\lambda - 1)c_2 \xrightarrow{-\frac{1}{2}r_3} \left[\begin{array}{ccc} 0 & \lambda(\lambda - 1) - 4 & \lambda^3 - 2\lambda^2 - 7\lambda - 4 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right] \\
 & \xrightarrow{r_1 - [\lambda(\lambda - 1) - 4]r_3} \left[\begin{array}{ccc} 0 & 0 & (\lambda + 1)^2(\lambda - 4) \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right] \\
 & = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (\lambda + 1)^2(\lambda - 4) \end{array} \right]
 \end{aligned}$$

2. 初等因子组为 $(\lambda + 1)^2, \lambda - 4$ ，它们对于的 Jordan 标准形为

$$\left[\begin{array}{cc} 1 & 1 \\ & 1 \end{array} \right], [4]$$

3. 所求矩阵的 Jordan 标准形为

$$\left[\begin{array}{ccc} 4 & & \\ & -1 & 1 \\ & & -1 \end{array} \right]$$

提示：在进行因式分解时，可以使用赋值数分解法，假设有 m 次多项式 $f(\lambda)$ ，代入 $\lambda = a$ ，有 $f(a) = X$ ，若 X 可以分解为 m 个数的乘积，可以据此分解因式。

如对 $f(\lambda) = \lambda^3 - 2\lambda^2 - 7\lambda - 4$ ，有 $f(1) = -12$ ，而 $-3 \times 2 \times 2 = -12$ ，可猜想 $f(\lambda) = (\lambda + a)(\lambda + b)(\lambda + c)$ ， $a = -4, b = 1, c = 1$ 。

在 Matlab 中进行因式分解很简单

```
>> syms x
>> y = x^3 - 2*x^2 - 7*x - 4
>> factor(y)
```

(下页继续)

(续上页)

```
ans =
[ x - 4, x + 1, x + 1]
```

在 Matlab 中可以很容易的实现 Jordan 标准形分解

```
>> A = [0 2 2; 2 1 2; 0 2 1]
>> jordan(A)

ans =
4     0     0
0    -1     1
0     0    -1
```

2.5.1.5.3.4 非奇异矩阵 P 的求法

求矩阵 A 的特征值与特征向量/广义特征向量, 将特征向量和广义特征向量组成矩阵 P , 即可.

以例子(广义特征向量)(页 36)为例, 求矩阵 P , 使得矩阵 A 与对角矩阵 J 相似.

解: 依据例子(广义特征向量)(页 36)的求解步骤求出矩阵 A 的特征值为 $\lambda_1 = 4, \lambda_2 = \lambda_3 = -1$, 对应于 λ_1 的特征向量 $x_1 = (5, 6, 4)^T$, 对应于 $\lambda_2 = \lambda_3 = -1$ 的特征向量和广义特征向量 $x_2 = (0, 1, -1)^T, x_3 = (1, -1, 1/2)^T$.

所以取

$$P = \begin{bmatrix} 5 & 0 & 1 \\ 6 & 1 & -1 \\ 4 & -1 & 1/2 \end{bmatrix}$$

可以验证 $P^{-1}AP = J$.

matlab 代码验证如下:

```
>> A=[0 2 2;2 1 2;0 2 1]

A =
0     2     2
2     1     2
0     2     1

>> P=[5 6 4;0 1 -1;1 -1 1/2]'

P =
```

(下页继续)

(续上页)

```

5.0000      0      1.0000
6.0000      1.0000   -1.0000
4.0000     -1.0000    0.5000

>> inv(P)*A*P

ans =

4.0000   -0.0000    0.0000
-0.0000   -1.0000    1.0000
0.0000      0     -1.0000

>> % [1 -1 1/2] --> [2 -2 1]
>> P=[5 6 4;0 1 -1;2 -2 1]'

P =

5.0000      0      2.0000
6.0000      1.0000   -2.0000
4.0000     -1.0000    1.0000

>> inv(P)*A*P

ans =

4.0000   -0.0000    0.0000
-0.0000   -1.0000    2.0000
0.0000      0     -1.0000

```

2.5.1.5.4 Jordan 标准型求解微分线性方程组

2.5.1.5.4.1 计算步骤

1. 将微分线性方程组写成矩阵形式 $\frac{dx}{dt} = Ax$ ；
2. 求解矩阵 A 的特征值和广义特征向量，组成矩阵 P 和 Jordan 标准形；
3. 进行非奇异线性变换 $x = Py \Rightarrow \frac{dx}{dt} = P \frac{dy}{dt}$ ，从而 $\frac{dy}{dt} = P^{-1} \frac{dx}{dt} = P^{-1} APy$ ；
4. 由 $\frac{dy}{dt} = Jy$ 求解出一般解；
5. 再由 $x = Py$ 求解出原微分方程的一般解。

2.5.1.5.4.2 举例

求解如下微分线性方程组的通解

$$\begin{cases} \frac{dx_1}{dt} = 2x_2 + 2x_3 \\ \frac{dx_2}{dt} = 2x_1 + x_2 + 2x_3 \\ \frac{dx_3}{dt} = 2x_2 + x_3 \end{cases}$$

解:

1. 写成矩阵形式 $\frac{dx}{dt} = Ax$, 其中

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

2. 求解矩阵 A 的特征值与广义特征向量与 P

由例子(广义特征向量)(页 36)知其特征值为 $\lambda_1 = 4, \lambda_2 = \lambda_3 = -1$, 对应于 λ_1 的特征向量 $x_1 = (5, 6, 4)^T$, 对应于 $\lambda_2 = \lambda_3 = -1$ 的广义特征向量 $x_2 = (0, 1, -1)^T, x_3 = (2, -2, 1)^T$.

所以取

$$P = \begin{bmatrix} 5 & 0 & 1 \\ 6 & 1 & -1 \\ 4 & -1 & 1/2 \end{bmatrix}$$

3. 由 $P^{-1}AP$ 或者初等因子组求解出 Jordan 标准形

$$J = \begin{bmatrix} 4 & & \\ & -1 & 1 \\ & & -1 \end{bmatrix}$$

4. 求解 $\frac{dy}{dt} = Jy$ 一般解

$$\begin{cases} \frac{dy_1}{dt} = 4y_1 \\ \frac{dy_2}{dt} = -y_2 + y_3 \\ \frac{dy_3}{dt} = -y_3 \end{cases} \Rightarrow \begin{cases} y_1 = c_1 e^{4t} \\ \frac{dy_2}{dt} = -y_2 + c_3 e^{-t} \\ y_3 = c_3 e^{-t} \end{cases} \Rightarrow \begin{cases} y_1 = c_1 e^{4t} \\ y_2 = c_2 e^{-t} + c_3 t e^{-t} \\ y_3 = c_3 e^{-t} \end{cases}$$

5. 由 $x = Py$ 知

$$\begin{aligned} x = Py &= \begin{bmatrix} 5 & 0 & 1 \\ 6 & 1 & -1 \\ 4 & -1 & 1/2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \\ \Rightarrow \begin{cases} x_1 = 5y_1 + y_3 \\ x_2 = 6y_1 + y_2 - y_3 \\ x_3 = 4y_1 - y_2 + y_3/2 \end{cases} &\Rightarrow \begin{cases} x_1 = 5c_1 e^{4t} + c_3 e^{-t} \\ x_2 = 6c_1 e^{4t} + c_2 e^{-t} + c_3 t e^{-t} - c_3 e^{-t} \\ x_3 = 4c_1 e^{4t} - c_2 e^{-t} - c_3 t e^{-t} + c_3 e^{-t}/2 \end{cases} \end{aligned}$$

提示: 在 Matlab 中可以使用 `dsolve` 函数求解微分方程组, 如对于 $\frac{dx}{dt} = -x$

```
>> syms x(t)
Dx = diff(x);
dsolve(diff(Dx) == -x, Dx(0) == 1)

ans =
sin(t) + C3*cos(t)
```

2.5.1.5.5 总结

2.5.1.5.5.1 可对角化

1. 互不相同的特征值对应的特征向量线性无关
2. n 阶矩阵 A 有 n 个线性无关的特征向量 $\Leftrightarrow A$ 与对角阵相似
3. n 阶矩阵 A 有 n 个互不相同的特征值 $\Leftrightarrow A$ 与对角阵相似
4. n 阶矩阵 A 的秩为 $n \Leftrightarrow A$ 与对角阵相似

2.5.1.6 广义特征值与特征向量

2.5.1.6.1 广义特征值与特征向量

2.5.1.6.1.1 矩阵的广义特征值与特征向量

Definition 21 (矩阵的广义特征值与特征向量) 设有矩阵 A, B , 且对于数域 \mathbb{K} 中的数 λ , 若存在非零向量 $x \in \mathbb{V}^n$ 满足如下条件

$$Ax = \lambda Bx$$

则称 λ 为矩阵 A 相对于矩阵 B 的 **广义特征值** (*Generalized Eigenvalue*), x 为 A 相对于矩阵 B 的属于特征值 λ 的 **广义特征向量** (*Generalized Eigenvector*). 上述问题称为 **广义特征值问题** (*Generalized eigenvalue problem*).

2.5.1.6.1.2 特征值与特征向量的关系

2.5.1.6.1.3 广义特征值问题的等价形式

2.5.1.6.1.4 第一种形式

当 B 正定即可逆时, 有第一种等价形式

$$B^{-1}Ax = \lambda x,$$

此时, 广义特征值问题退化为矩阵 $B^{-1}A$ 的普通特征值问题.

2.5.1.6.1.5 第二种形式

当 \mathbf{B} 正定即可逆时, 对其进行 Cholesky 分解 $\mathbf{B} = \mathbf{G}\mathbf{G}^T$, 其中 \mathbf{G} 是下三角矩阵, 则有

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{G}\mathbf{G}^T\mathbf{x},$$

令 $\mathbf{y} = \mathbf{G}^T\mathbf{x}$, 则 $\mathbf{x} = (\mathbf{G}^{-1})^T\mathbf{y}$, 则广义特征值问题变为

$$\mathbf{S}\mathbf{y} = \lambda\mathbf{y}$$

此时, 广义特征值问题退化为矩阵 $\mathbf{S} = \mathbf{G}^{-1}\mathbf{A}(\mathbf{G}^{-1})^T$ 的普通特征值问题.

2.5.1.6.1.6 实例

2.5.1.7 特殊线性空间

2.5.1.7.1 内积空间

2.5.1.7.1.1 什么是内积空间

内积空间 (*Inner product space*): 在线性代数中, 内积空间是具有称为内积的附加结构的向量空间. 这个附加结构将空间中的每对向量与称为向量内积的标量量相关联. 内积允许严格引入直观的几何概念, 如向量的长度或两个向量之间的角度. 它们还提供了定义向量之间的正交性(零内积)的方法. 内积空间将欧氏空间(其中内积是点积, 也称为标量积)推广到任意(可能是无限)维的向量空间, 并在泛函分析中加以研究. 带内积的向量空间概念的首次使用是由于 Peano 在 1898 年提出的.

Definition 22 (内积空间) 设 \mathbb{V} 是数域 \mathbb{K} 上的线性空间 (线性空间 (页 14)), 对于 \mathbb{V} 中的任意两个元素, 按某一规则定义一个实数, 记为 $\langle \mathbf{x}, \mathbf{y} \rangle$, 若它满足

1. 交换律: $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{x}, \mathbf{y} \rangle}$
2. 分配率: $\langle \mathbf{x}, \mathbf{y} \oplus \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle$
3. 齐次性: $\langle k \odot \mathbf{x}, \mathbf{y} \rangle = k \langle \mathbf{x}, \mathbf{y} \rangle, (\forall k \in \mathbb{K})$
4. 非负性: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$, 当且仅当 $\mathbf{x} = \mathbf{0}$, $\langle \mathbf{x}, \mathbf{x} \rangle = 0$

则称 $\langle \mathbf{x}, \mathbf{y} \rangle$ 为 \mathbf{x} 与 \mathbf{y} 的内积, 称 \mathbb{V} 为 内积空间 (*Inner product space*).

警告: 为什么不定义一个复数?

2.5.1.7.1.2 欧式空间 (实内积空间)

当上述内积空间中的数域 \mathbb{K} 为实数域 \mathbb{R} 时, 称 \mathbb{V} 为 **欧式空间** (Euclid space) 或 **实内积空间**.

2.5.1.7.1.3 酉空间 (复内积空间)

当上述内积空间中的数域 \mathbb{K} 为实数域 \mathbb{C} 时, 称 \mathbb{V} 为 **酉空间** (Euclid space) 或 **复内积空间**.

2.5.1.7.1.4 常见内积及其性质

2.5.1.7.1.5 定义

在复 n 维向量空间 \mathbb{C}^n 中, 对于任意两个元素

$$\mathbf{x} = (\xi_1, \xi_2, \dots, \xi_n), \mathbf{y} = (\eta_1, \eta_2, \dots, \eta_n)$$

定义它们的内积为

$$\langle \mathbf{x}, \mathbf{y} \rangle = \xi_1 \bar{\eta}_1 + \xi_2 \bar{\eta}_2 + \dots + \xi_n \bar{\eta}_n = \mathbf{x} \mathbf{y}^H$$

上述定义对实 n 维向量空间 \mathbb{R}^n 也成立.

2.5.1.7.1.6 性质

- $\langle \mathbf{x}, \mathbf{x} \rangle = \sum_{i=1}^n |\xi_i|^2$
- $\langle \mathbf{x}, k\mathbf{y} \rangle = \bar{k} \langle \mathbf{x}, \mathbf{y} \rangle$
- $\langle \mathbf{x}, \mathbf{0} \rangle = \langle \mathbf{0}, \mathbf{x} \rangle = 0$
- $\left\langle \sum_{i=1}^n \xi_i \mathbf{x}_i, \sum_{j=1}^n \eta_j \mathbf{y}_j \right\rangle = \sum_{i,j=1}^n \xi_i \bar{\eta}_j \langle \mathbf{x}_i, \mathbf{y}_j \rangle$
- 长度: $|\mathbf{x}| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
- 夹角: $\cos(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{|\mathbf{x}| |\mathbf{y}|}$, 当 $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ 时, 称 \mathbf{x}, \mathbf{y} 正交或 垂直
- Cauchy-不等式: $\langle \mathbf{x}, \mathbf{y} \rangle \langle \mathbf{y}, \mathbf{x} \rangle \leq \langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle$
- 任意线性无关的向量组可以用 Schmidt 正交化
- 任一非零酉空间都存在正交基和标准正交基
- 任一 n 维酉空间 \mathbb{V}^n 均为其子空间 \mathbb{V}_1 与 \mathbb{V}_1^\perp 的和

2.5.1.7.1.7 度量矩阵

2.5.1.7.1.8 Schmidt 正交化

任给线性空间 V 中的一组向量, 可以采用 **Schmidt 正交化** (*Gram–Schmidt process*) 方法正交化之, 其原理如图 2.7 所示, 动态示意图如 图 2.7 所示.

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{v}_1 \\ \mathbf{u}_2 &= \mathbf{v}_2 - \frac{\langle \mathbf{v}_2, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 \\ \mathbf{u}_3 &= \mathbf{v}_3 - \frac{\langle \mathbf{v}_3, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 - \frac{\langle \mathbf{v}_3, \mathbf{u}_2 \rangle}{\langle \mathbf{u}_2, \mathbf{u}_2 \rangle} \mathbf{u}_2\end{aligned}$$

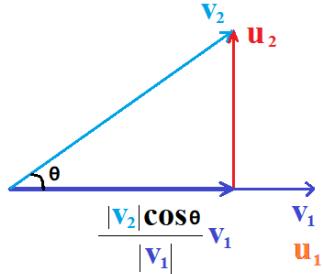


图 2.6: Schmidt 正交化图解
Schmidt 正交化图解 (二维空间域三维空间)

$$\|\mathbf{e}_1\| = \|\mathbf{e}_2\| = \|\mathbf{e}_3\| = 1$$

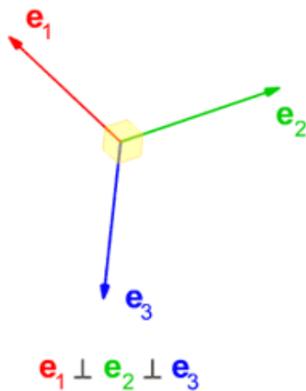


图 2.7: Schmidt 正交化动图示意
Schmidt 正交化动图示意 (来自维基百科)

一般地, 设有内积空间 V^n 中的 m 个元素 $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$ $1 \leq m \leq n, m \in \mathbb{Z}^+$, 可以使用如下 Schmidt 正交

化步骤正交化它们:

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{v}_1 \\ \mathbf{u}_2 &= \mathbf{v}_2 - \frac{\langle \mathbf{v}_2, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 \\ \mathbf{u}_3 &= \mathbf{v}_3 - \frac{\langle \mathbf{v}_3, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 - \frac{\langle \mathbf{v}_3, \mathbf{u}_2 \rangle}{\langle \mathbf{u}_2, \mathbf{u}_2 \rangle} \mathbf{u}_2 \\ &\vdots \\ \mathbf{u}_m &= \mathbf{v}_m - \frac{\langle \mathbf{v}_m, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 - \frac{\langle \mathbf{v}_m, \mathbf{u}_2 \rangle}{\langle \mathbf{u}_2, \mathbf{u}_2 \rangle} \mathbf{u}_2 \cdots - \frac{\langle \mathbf{v}_m, \mathbf{u}_{m-1} \rangle}{\langle \mathbf{u}_{m-1}, \mathbf{u}_{m-1} \rangle} \mathbf{u}_{m-1}\end{aligned}$$

2.5.1.7.1.9 内积空间中线性变换

设 $\mathbf{e}_1, \mathbf{e}_2$ 为二维欧氏空间 \mathbb{R}^2 的一组基, \mathbb{R}^2 中的两个元素 \mathbf{x}, \mathbf{y} 在该基下的坐标表示为 $\mathbf{x} = (\xi_1, \xi_2), \mathbf{y} = (\eta_1, \eta_2)$, 线性变换 T 在该基下的矩阵为 $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, 则有

$$\begin{aligned}T\mathbf{x} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} a\xi_1 + b\xi_2 \\ c\xi_1 + d\xi_2 \end{bmatrix} \\ T\mathbf{y} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} a\eta_1 + b\eta_2 \\ c\eta_1 + d\eta_2 \end{bmatrix}\end{aligned}$$

从而有

$$\begin{aligned}\langle \mathbf{x}, \mathbf{y} \rangle &= \left\langle (\xi_1, \xi_2)^T, (\eta_1, \eta_2)^T \right\rangle = \xi_1\eta_1 + \xi_2\eta_2 \\ \langle T\mathbf{x}, \mathbf{y} \rangle &= \left\langle (a\xi_1 + b\xi_2, c\xi_1 + d\xi_2)^T, (\eta_1, \eta_2)^T \right\rangle \\ &= (a\xi_1 + b\xi_2)\eta_1 + (c\xi_1 + d\xi_2)\eta_2 \\ &= a\xi_1\eta_1 + b\xi_2\eta_1 + c\xi_1\eta_2 + d\xi_2\eta_2 \\ \langle \mathbf{x}, T\mathbf{y} \rangle &= \left\langle (\xi_1, \xi_2)^T, (a\eta_1 + b\eta_2, c\eta_1 + d\eta_2)^T \right\rangle \\ &= \xi_1(a\eta_1 + b\eta_2) + \xi_2(c\eta_1 + d\eta_2) \\ &= a\xi_1\eta_1 + b\xi_1\eta_2 + c\xi_2\eta_1 + d\xi_2\eta_2 \\ \langle T\mathbf{x}, T\mathbf{y} \rangle &= \left\langle (a\xi_1 + b\xi_2, c\xi_1 + d\xi_2)^T, (a\eta_1 + b\eta_2, c\eta_1 + d\eta_2)^T \right\rangle \\ &= (a\xi_1 + b\xi_2)(a\eta_1 + b\eta_2) + (c\xi_1 + d\xi_2)(c\eta_1 + d\eta_2) \\ &= a^2\xi_1\eta_1 + b^2\xi_2\eta_2 + ab\xi_1\eta_2 + ab\xi_2\eta_1 + \\ &\quad c^2\xi_1\eta_1 + d^2\xi_2\eta_2 + cd\xi_1\eta_2 + cd\xi_2\eta_1 \\ &= (a^2 + c^2)\xi_1\eta_1 + (b^2 + d^2)\xi_2\eta_2 + (ab + cd)\xi_1\eta_2 + (ab + cd)\xi_2\eta_1\end{aligned}$$

1. 由 $\langle T\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, T\mathbf{y} \rangle$ 可以推出

$$b = c$$

2. 由 $\langle T\mathbf{x}, \mathbf{y} \rangle = -\langle \mathbf{x}, T\mathbf{y} \rangle$ 可以推出

$$b = -c$$

3. 由 $\langle \mathbf{x}, \mathbf{y} \rangle = \langle T\mathbf{x}, T\mathbf{y} \rangle$ 可以推出

$$\begin{cases} a^2 + c^2 = 1 \\ b^2 + d^2 = 1 \\ ab + cd = 0 \end{cases}$$

2.5.1.7.1.10 实对称变换

设 T 是欧氏空间 \mathbb{V} 的一个线性变换, 且对 \mathbb{V} 中的任意两个元素 \mathbf{x}, \mathbf{y} 有

$$\langle T\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, T\mathbf{y} \rangle,$$

则称 T 是 \mathbb{V} 中的 **实对称变换**, 一般简称为 **对称变换**.

对称变换在标准正交基下的矩阵满足

$$\mathbf{A}^T = \mathbf{A}$$

称为 **实对称矩阵** (*Symmetric matrix*), 其中, \mathbf{A}^T 为 \mathbf{A} 的 **转置矩阵** (也可以是共轭转置, 实数的共轭为其自身), 即有 $a_{ij} = a_{ji}$.

注解:

1. 欧式空间的线性变换 T 是实对称变换 $\leftrightarrow T$ 在标准正交基下的矩阵为实对称矩阵;
2. 实对称矩阵的特征值都是实数;
3. 实对称矩阵的不同特征值对应特征向量正交.

2.5.1.7.1.11 实反对称变换

设 T 是欧氏空间 \mathbb{V} 的一个线性变换, 且对 \mathbb{V} 中的任意两个元素 \mathbf{x}, \mathbf{y} 有

$$\langle T\mathbf{x}, \mathbf{y} \rangle = -\langle \mathbf{x}, T\mathbf{y} \rangle,$$

则称 T 是 \mathbb{V} 中的 **实反对称变换**, 一般简称为 **反对称变换**.

反对称变换在标准正交基下的矩阵满足

$$\mathbf{A}^T = -\mathbf{A}$$

称为 **实反对称矩阵** (*Skew-symmetric matrix*), 其中, \mathbf{A}^T 为 \mathbf{A} 的 **转置矩阵** (也可以是共轭转置, 实数的共轭为其自身), 即有 $a_{ij} = -a_{ji}$.

2.5.1.7.1.12 正交变换

设 \mathbb{V} 是欧氏空间, T 是 \mathbb{V} 的一个线性变换, 若 T 对 \mathbb{V} 中的任意元素 $\mathbf{x} \neq \mathbf{y}$ 满足

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle T\mathbf{x}, T\mathbf{y} \rangle,$$

那么称 T 是线性空间 \mathbb{V} 的一个 **正交变换** (*Orthogonal transformation*).

正交变换在标准正交基下的矩阵满足

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{I} \quad \mathbf{Q}^{-1} = \mathbf{Q}^T$$

称为 **正交矩阵**.

提示:

1. 正交变换是保持图形形状和大小不变的几何变换, 包含旋转, 平移, 轴对称及上述变换的复合.
2. 正交变换可以保证向量的长度和两个向量之间的夹角不变.
3. 正交变换将正交基映射到正交基.
4. 二维或三维欧氏空间中的正交变换是刚性旋转、反射或旋转和反射的组合.

容易验证, 旋转变换是正交变换

- 逆时针旋转: $\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta - x_2 \sin \theta \\ x_1 \sin \theta + x_2 \cos \theta \end{bmatrix}$
- 顺时针旋转: $\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta + x_2 \sin \theta \\ -x_1 \sin \theta + x_2 \cos \theta \end{bmatrix}$

注解: 设 x, y 是欧氏空间中的元素, 则有

$$\begin{aligned} T\mathbf{x} &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta - x_2 \sin \theta \\ x_1 \sin \theta + x_2 \cos \theta \end{bmatrix} \\ T\mathbf{y} &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \cos \theta - y_2 \sin \theta \\ y_1 \sin \theta + y_2 \cos \theta \end{bmatrix} \end{aligned}$$

从而有

$$\begin{aligned} \langle T\mathbf{x}, T\mathbf{y} \rangle &= (\xi_1 \cos \theta - \xi_2 \sin \theta)(\eta_1 \cos \theta - \eta_2 \sin \theta) \\ &\quad + (\xi_1 \sin \theta + \xi_2 \cos \theta)(\eta_1 \sin \theta + \eta_2 \cos \theta) \\ &= \xi_1 \eta_1 \cos^2 \theta + \xi_2 \eta_2 \sin^2 \theta - \xi_1 \eta_2 \sin \theta \cos \theta - \xi_2 \eta_1 \sin \theta \cos \theta \\ &\quad + \xi_1 \eta_1 \sin^2 \theta + \xi_2 \eta_2 \cos^2 \theta + \xi_1 \eta_2 \sin \theta \cos \theta + \xi_2 \eta_1 \sin \theta \cos \theta \\ &= \xi_1 \eta_1 + \xi_2 \eta_2 = \langle \mathbf{x}, \mathbf{y} \rangle \end{aligned}$$

2.5.1.7.1.13 酉对称变换

设 T 是欧氏空间 \mathbb{V} 的一个线性变换, 且对 \mathbb{V} 中的任意两个元素 \mathbf{x}, \mathbf{y} 有

$$\langle T\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, T\mathbf{y} \rangle,$$

则称 T 是 \mathbb{V} 中的 **酉对称变换**.

酉对称变换在标准正交基下的矩阵满足

$$\mathbf{A}^H = \mathbf{A}$$

称为 **Hermite 矩阵** (*Hermitian matrix*) 或 **酉对称矩阵**, 其中, \mathbf{A}^H 为 \mathbf{A} 的 **共轭转置转置矩阵**, 即有 $a_{ij} = \bar{a}_{ji}$

注解:

1. 西空间的线性变换 T 是西对称变换 $\leftrightarrow T$ 在标准正交基下的矩阵为西对称矩阵, 即 hermite 矩阵;
 2. 西对称矩阵的特征值都是实数;
 3. 西对称矩阵的不同特征值对应特征向量正交.
-

2.5.1.7.1.14 反西对称变换

设 T 是欧氏空间 \mathbb{V} 的一个线性变换, 且对 \mathbb{V} 中的任意两个元素 \mathbf{x}, \mathbf{y} 有

$$\langle T\mathbf{x}, \mathbf{y} \rangle = -\langle \mathbf{x}, T\mathbf{y} \rangle,$$

则称 T 是 \mathbb{V} 中的 **反西对称变换**.

反西对称变换在标准正交基下的矩阵满足

$$\mathbf{A}^H = -\mathbf{A}$$

称为 **反 Hermite 矩阵** (*Skew-Hermitian matrix*) 或 **反西对称矩阵**, 其中, \mathbf{A}^H 为 \mathbf{A} 的 **共轭转置转置矩阵**, 即有 $a_{ij} = -\bar{a}_{ji}$.

2.5.1.7.1.15 西变换

设 \mathbb{V} 是西空间, T 是 \mathbb{V} 的一个线性变换, 若 T 对 \mathbb{V} 中的任意元素 $\mathbf{x} \otimes \mathbf{y}$ 满足

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle T\mathbf{x}, T\mathbf{y} \rangle,$$

那么称 T 是线性空间 \mathbb{V} 的一个 **西变换** (*Unitary transformation*).

西变换在标准正交基下的矩阵满足

$$\mathbf{A}^H \mathbf{A} = \mathbf{A} \mathbf{A}^H = \mathbf{I}$$

称为 **酉矩阵**.

提示:

1. 复内积空间的酉变换对于实内积空间的正交变换.
 2. 酉变换可以保证向量的长度和两个向量之间的夹角不变.
 3. 酉变换将正交基映射到正交基.
-

设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 且满足

$$\mathbf{A}^H \mathbf{A} = \mathbf{A} \mathbf{A}^H$$

则称 \mathbf{A} 为 **正规矩阵**.

2.5.1.7.1.16 Givens 旋转变换

Givens 变换 (*Givens transformation*)，也称 Givens 旋转 (Givens rotation)，是描述将某一平面内的向量进行旋转的线性变换。如下图所示为二维平面内的旋转变换：

一般地，在 n 维欧式空间 \mathbb{R}^n 中，设 e_1, e_2, \dots, e_n 是一组标准正交基，在平面 $[e_i, e_j]$ 中可定义旋转变换

$$G_{i,j,\theta} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & c & s & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \\ i & & & & & & & & j \end{bmatrix}$$

其中， $c^2 + s^2 = 1$ ， $c = \cos\theta$, $s = \sin\theta$ 称其为 **Givens 矩阵**，也称 **初等旋转矩阵**，所对应的线性变换称为 **Givens 变换**，也称 **初等旋转变换**。

性质：

- 正交： $G_{i,j,\theta}^{-1} = G_{i,j,\theta}^T = G_{i,j,-\theta}$
- $\det(G_{i,j,\theta}) = 1$

设 $x = (\xi_1, \xi_2, \dots, \xi_n)^T$, $y = G_{i,j,\theta}x = (\eta_1, \eta_2, \dots, \eta_n)^T$ 则选取如下的 c, s ，可以使得 $\eta_i = \sqrt{\xi_i^2 + \xi_j^2} > 0, \eta_j = 0$

$$c = \cos\theta = \frac{\xi_i}{\sqrt{\xi_i^2 + \xi_j^2}}, s = \sin\theta = \frac{\xi_j}{\sqrt{\xi_i^2 + \xi_j^2}}$$

定理： 设 $x = (\xi_1, \xi_2, \dots, \xi_n)^T \neq \mathbf{0}$ ，则存在有限个 Givens 矩阵的乘积 $G = G_{1,n,\theta_{n-1}} G_{1,n-1,\theta_{n-2}} \cdots G_{1,2,\theta_1}$ ，使得 $Gx = |x|e_1$ ，其中

- $G_{1,2,\theta_1}$ 满足： $c = \cos\theta_1 = \frac{\xi_1}{\sqrt{\xi_1^2 + \xi_2^2}}, s = \sin\theta_1 = \frac{\xi_2}{\sqrt{\xi_1^2 + \xi_2^2}}$
- $G_{1,3,\theta_2}$ 满足： $c = \cos\theta_2 = \frac{\sqrt{\xi_1^2 + \xi_2^2}}{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}}, s = \sin\theta_2 = \frac{\xi_3}{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}}$
- \vdots
- $G_{1,n-1,\theta_{n-2}}$ 满足： $c = \cos\theta_{n-2} = \frac{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-2}^2}}{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-2}^2 + \xi_{n-1}^2}}, s = \sin\theta_{n-2} = \frac{\xi_{n-1}}{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-2}^2 + \xi_{n-1}^2}}$
- $G_{1,n,\theta_{n-1}}$ 满足： $c = \cos\theta_{n-1} = \frac{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-1}^2}}{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-1}^2 + \xi_n^2}}, s = \sin\theta_{n-1} = \frac{\xi_n}{\sqrt{\xi_1^2 + \xi_2^2 + \dots + \xi_{n-1}^2 + \xi_n^2}}$

注解：举个例子，设 $x = (2, 4, 4)^T$ ，用 Givens 变换，将 x 化为与 e_1 同方向的向量。

解：由题意，构造

- $\mathbf{G}_{1,2,\theta_1} = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$, 使得 $\mathbf{G}_{1,2,\theta_1} \mathbf{x} = (\frac{10}{\sqrt{5}}, 0, 4)^T$, 其中 $c = \cos\theta_1 = \frac{\xi_1}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{2}{\sqrt{2^2 + 4^2}} = \frac{1}{\sqrt{5}}$,
 $s = \sin\theta_1 = \frac{\xi_2}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{4}{\sqrt{2^2 + 4^2}} = \frac{2}{\sqrt{5}}$
- $\mathbf{G}_{1,3,\theta_2} = \begin{bmatrix} \frac{\sqrt{5}}{3} & 0 & \frac{2}{3} \\ 0 & 1 & 0 \\ \frac{-2}{3} & 0 & \frac{\sqrt{5}}{3} \end{bmatrix}$ 使得 $\mathbf{G}_{1,3,\theta_2}(\mathbf{G}_{1,2,\theta_1} \mathbf{x}) = (6, 0, 0)^T$, 其中 $c = \cos\theta_2 = \frac{\sqrt{2^2 + 4^2}}{\sqrt{2^2 + 4^2 + 4^2}} = \frac{\sqrt{5}}{3}$,
 $s = \sin\theta_2 = \frac{4}{\sqrt{2^2 + 4^2 + 4^2}} = \frac{2}{3}$

提示: 由于旋转变换不改变向量的长度, 如上述例子, 变换前后长度均为 6, 所以在计算 c, s 时可以根据原始坐标 $\mathbf{x} = (\xi_1, \xi_2, \dots, \xi_n)^T$ 计算.

警告: 如果 Givens 矩阵定义为逆时针旋转的矩阵, 即将 $\mathbf{G}_{i,j,\theta}$ 中的 s 变为 $-s$, 那么上述相应的结论做对应的改变即可.

2.5.1.7.1.17 Householder 反射变换

Householder 变换 (*Householder transformation*) 也称为 Householder 反射 (Householder reflection), 是描述包含原点的平面或超平面的反射的线性变换. 反射超平面可以通过与其正交的单位向量 \mathbf{v} 来定义, 点 \mathbf{x} 关于此超平面的映射是一个线性变换, 定义如下:

$$\mathbf{y} = \mathbf{x} - 2 \langle \mathbf{v}, \mathbf{x} \rangle \mathbf{v} = \mathbf{x} - 2\mathbf{v}(\mathbf{v}^H \mathbf{x}) = (\mathbf{I} - 2\mathbf{v}\mathbf{v}^H)\mathbf{x} = \mathbf{H}\mathbf{x},$$

其中, \mathbf{v} 是单位列向量. \mathbf{H} 称为 **Householder 矩阵**, 也称 **初等反射矩阵**, 所对应的线性变换称为 **Householder 变换** (*Householder transformation*), 如 图 2.8 所示.

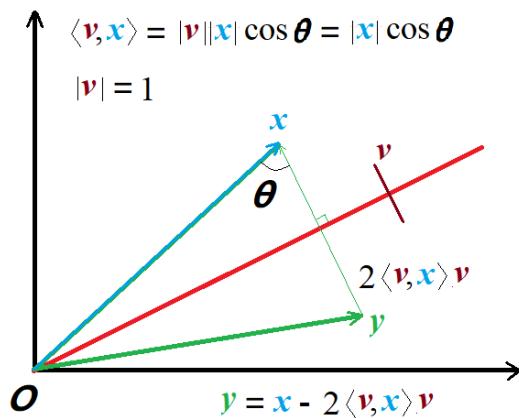


图 2.8: 二维空间中的 Householder 变换.
二维空间中的 Householder 变换. 其中, \mathbf{v} 为超平面的单位法向量.

性质:

- 对称: $\mathbf{H}^H = \mathbf{H}$
- 自逆: $\mathbf{H}^{-1} = \mathbf{H}$
- 对合: $\mathbf{H}^2 = \mathbf{I}$
- 正交: $\mathbf{H}^H \mathbf{H} = \mathbf{I}$
- $\det(\mathbf{H}) = -1$

定理: 设 $\mathbf{x} = (\xi_1, \xi_2, \dots, \xi_n)^T \neq \mathbf{0}$ 单位列向量 \mathbf{z} , 则存在 Householder 矩阵 \mathbf{H} , 使得 $\mathbf{H}\mathbf{x} = |\mathbf{x}|\mathbf{z}$, 其中

$$\mathbf{H} = \mathbf{I} - 2\mathbf{v}\mathbf{v}^H$$

且

$$\mathbf{v} = \frac{\mathbf{x} - |\mathbf{x}|\mathbf{z}}{|\mathbf{x} - |\mathbf{x}|\mathbf{z}|}$$

注解: 举个例子, 设 $\mathbf{x} = (2, 4, 4)^T$, 用 Householder 变换, 将 \mathbf{x} 化为与 \mathbf{e}_1 同方向的向量.

解: 由题意, 有

$$\mathbf{x} - |\mathbf{x}|\mathbf{e}_1 = (2, 4, 4)^T - 6(1, 0, 0)^T = (-4, 4, 4)^T$$

$$\text{且 } \mathbf{v} = \frac{\mathbf{x} - |\mathbf{x}|\mathbf{e}_1}{|\mathbf{x} - |\mathbf{x}|\mathbf{e}_1|} = \frac{(-4, 4, 4)^T}{\sqrt{4^2 + 4^2 + 4^2}} = \left(\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)^T$$

$$\text{则 } \mathbf{H} = \mathbf{I} - 2\mathbf{v}\mathbf{v}^H = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{-2}{3} \\ \frac{2}{3} & \frac{-2}{3} & \frac{1}{3} \end{bmatrix} \text{ 使得 } \mathbf{H}\mathbf{x} = (6, 0, 0)^T.$$

2.5.1.7.1.18 谱分解

对于 n 阶 Hermite 矩阵 \mathbf{A} , 存在 n 阶酉矩阵 \mathbf{P} , 使得

$$\mathbf{P}^H \mathbf{A} \mathbf{P} = \mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

即有

$$\mathbf{A} = \mathbf{P} \mathbf{\Lambda} \mathbf{P}^H = \lambda_1(\mathbf{p}_1 \mathbf{p}_1^H) + \lambda_2(\mathbf{p}_2 \mathbf{p}_2^H) + \dots + \lambda_n(\mathbf{p}_n \mathbf{p}_n^H)$$

其中, $\lambda_i (i = 1, 2, \dots, n)$ 是 \mathbf{A} 的特征值. 称上式为矩阵 \mathbf{A} 的 **谱分解** (*Spectral decomposition*)

2.5.1.7.2 总结

当复内积空间的内积定义为 $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}\mathbf{y}^H$ (行向量) 或 $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^H \mathbf{y}$ (列向量) 时,

- 欧式空间 \rightarrow 实内积空间
- 酉空间 \rightarrow 复内积空间
- 实内积空间与复内积空间一一对应

- 实正交变换对应酉变换
- 实对称变换对应酉对称变换
- 正交变换在标准正交基下的矩阵为正交矩阵
- 对称变换在标准正交基下的矩阵为实对称矩阵
- 酉变换在标准正交基下的矩阵为酉矩阵
- 酉对称变换在标准正交基下的矩阵为酉对称矩阵

2.5.2 范数理论及应用

2.5.2.1 向量范数

2.5.2.1.1 概念与内涵

Definition 23 (向量范数) 设 \mathbb{V} 是数域 \mathbb{K} 上的线性空间, 若对于 \mathbb{V} 中的任意元素 x , 存在实数 $\|x\|$ 满足:

1. 非负性: $\|x\| \geq 0$, 当且仅当 $x = \mathbf{0}$ 时, 等号成立;
2. 齐次性: $\|k \odot x\| = |k|\|x\|$, ($k \in \mathbb{K}, x \in \mathbb{V}$)
3. 三角不等式: $\|x \oplus y\| \leq \|x\| + \|y\|$, ($x, y \in \mathbb{V}$)

则称 $\|x\|$ 为向量 x 在线性空间 \mathbb{V} 上的 **向量范数**.

提示:

- 范数是实数, 为什么不定义复数呢?
 - 注意与内积的定义比较
 - 注意与矩阵范数的定义比较
-

2.5.2.1.2 向量范数的等价性

设 $\|x\|_\alpha, \|x\|_\beta$ 为有限维线性空间 \mathbb{V} 上的任意两种范数, 则存在数 c_1, c_2 与向量无关, 且使得对于 $\forall x \in \mathbb{V}$ 如下不等式成立

$$c_1\|x\|_\beta \leq \|x\|_\alpha \leq c_2\|x\|_\beta$$

称其为向量范数的等价性质.

提示: 如对于向量的 ℓ_1, ℓ_2 范数 $|x|, \|x\|$

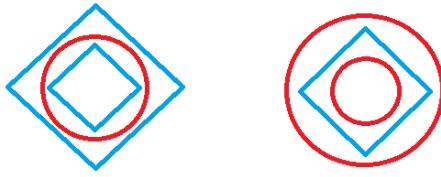


图 2.9: 向量范数的等价性
向量范数的等价性 ℓ_1 (蓝, blue), ℓ_2 (红, red)

2.5.2.1.3 常见向量范数

2.5.2.1.3.1 普通向量范数

设有线性空间 \mathbb{V}^n , $x = (\xi_1, \xi_2, \dots, \xi_n) \in \mathbb{V}^n$, 其中, ξ_i 为 x 的第 i 个坐标, $|\xi_i|$ 表示 ξ 的模, 则有以下常见范数

- **p-范数或 ℓ_p 范数:** $\|x\|_p = (\sum_{i=1}^n |\xi_i|^p)^{1/p}$, $1 \leq p < +\infty$
- **∞ -范数或 ℓ_∞ 范数:** $\|x\|_\infty = \max_i |\xi_i|$, ($p \rightarrow +\infty$)
- **1-范数或 ℓ_1 范数:** $\|x\|_1 = \sum_{i=1}^n |\xi_i|$, ($p = 1$)
- **2-范数或 ℓ_2 范数:** $\|x\|_2 = \sqrt{\sum_{i=1}^n |\xi_i|^2}$, ($p = 2$)

当 \mathbb{V}^n 为复空间 \mathbb{C}^n , 或实数空间 \mathbb{R}^n 时也有上述范数.

注解: 对区间 $[a, b]$ 上的实值连续函数集合, 和实数域 \mathbb{R} , 定义通常函数的加法, 实数与函数的数乘运算, 构成 \mathbb{R} 上的一个线性空间 \mathbb{V} , 有以下范数:

- $\|f(t)\|_p = \left[\int_a^b |f(t)|^p dt \right]^{1/p}$, ($1 \leq p < +\infty$)
- $\|f(t)\|_{+\infty} = \max_{t \in [a, b]} |f(t)|$

2.5.2.1.3.2 加权范数

Definition 24 (加权范数) 设 A 为 n 阶对称正定阵, $x \in \mathbb{R}^n$, 则称

$$\|x\|_A = (x^T A x)^{1/2}$$

为 **加权范数 (Weighted Norm)** 或 **椭圆范数**.

同理, 设 A 为 n 阶 *Hermite* 对称正定阵, $x \in \mathbb{C}^n$, 则称

$$\|x\|_A = (x^H A x)^{1/2}$$

为 **加权范数 (Weighted Norm)** 或 **椭圆范数**.

2.5.2.2 矩阵范数

2.5.2.2.1 概念与内涵

Definition 25 (矩阵范数) 若对于 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 定义实值函数 $\|\mathbf{A}\|$, 满足以下条件

1. 非负性: $\|\mathbf{A}\| \geq 0$, 当且仅当 $\mathbf{A} = \mathbf{0}$ 时, 等号成立;
2. 齐次性: $\|k\mathbf{A}\| = |k|\|\mathbf{A}\|$, ($k \in \mathbb{C}$)
3. 三角不等式: $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$, ($\mathbf{B} \in \mathbb{C}^{m \times n}$)
4. 相容性: $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ ($\mathbf{B} \in \mathbb{C}^{n \times l}$)

若满足 1, 2, 3 则称 $\|\mathbf{A}\|$ 为矩阵 \mathbf{A} 的 广义矩阵范数. 若满足 1, 2, 3, 4 则称 $\|\mathbf{A}\|$ 为矩阵 \mathbf{A} 的 矩阵范数.

提示:

- 范数是实数, 为什么不定义复数呢?
- 注意与内积的定义比较
- 注意与向量范数的定义比较

2.5.2.2.2 矩阵范数与向量范数的相容性

设有 $\mathbb{C}^{m \times n}$ 上的矩阵范数 $\|\cdot\|_M$ 和 \mathbb{C}^m 与 \mathbb{C}^n 上的同类向量范数 $\|\cdot\|_V$, 若

$$\|\mathbf{Ax}\|_V \leq \|\mathbf{A}\|_M \|\mathbf{x}\|_V, (\forall \mathbf{A} \in \mathbb{C}^{m \times n}, \forall \mathbf{x} \in \mathbb{C}^n)$$

则称矩阵范数 $\|\cdot\|_M$ 与向量范数 $\|\cdot\|_V$ 相容.

2.5.2.2.3 常见矩阵范数

Definition 26 (Frobenius 矩阵范数) 设有 $\mathbf{A} = (a_{ij})_{m \times n} \in \mathbb{C}^{m \times n}$, 则

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{1/2} = (\text{tr}(\mathbf{A}^H \mathbf{A}))^{1/2}$$

是 $\mathbb{C}^{m \times n}$ 上的矩阵范数, 且与向量范数 $\|\cdot\|_2$ 相容. 也称为 **Frobenius 范数**, 或简称 **F-范数**.

定理: 给矩阵左乘或右乘一个酉矩阵其 F-范数不变, 即设 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 且 $\mathbf{P} \in \mathbb{C}^{m \times m}$, $\mathbf{Q} \in \mathbb{C}^{n \times n}$ 为酉矩阵, 则

$$\|\mathbf{PA}\|_F = \|\mathbf{A}\|_F = \|\mathbf{AQ}\|_F$$

2.5.2.2.3.1 从属范数

矩阵范数与向量范数密切相关, 一一对应.

Definition 27 (从属范数) 设有 $\mathbb{C}^m, \mathbb{C}^n$ 上的同类向量范数 $\|\cdot\|$, $A \in \mathbb{C}^{m \times n}$, 则函数

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

是 $\mathbb{C}^{m \times n}$ 上的矩阵范数, 且与向量范数 $\|\cdot\|$ 相容. 由上式给出的矩阵范数称为 **由向量导出的矩阵范数**, 简称 **从属范数 (Subordinate Norm)**.

据此可以定义以下常见矩阵范数:

- **∞ -范数或 行和范数:** $\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$
- **1-范数或 列和范数:** $\|A\|_1 = \max_j \sum_{i=1}^m |a_{ij}|$
- **2-范数或 谱范数:** $\|A\|_2 = \sqrt{\lambda_1} = \sqrt{\max \lambda(A^H A)} = \max \sigma_i$, (λ_1 为 $A^H A$ 的最大特征值)

提示: 设 $A \in \mathbb{C}_r^{m \times n}, r > 0$, 则 $A^H A$ 为正定阵, 其特征值均为非负实数, 奇异值 $\sigma_i = \sqrt{\lambda_i}$, ($i = 1, 2, \dots, n$)

注解: [谱范数正则\(Spectral Norm Regularization\)的理解](https://blog.csdn.net/StreamRock/article/details/83539937) (<https://blog.csdn.net/StreamRock/article/details/83539937>)

谱范数正则 (Spectral Norm Regularization, 简称为 SNR) 最早来自于 2017 年 5 月日本国立信息研究所 Yoshida 的一篇论文 [2], 他们后续又于 2018 年 2 月再在 arXiv 发了一篇 SNR 用于 Gan 的论文 [3], 以阐明 SNR 的有效性。因为当 SGD (统计梯度下降) 的批次 (Batch size) 一大的时候, 其泛化性能却会降低, SNR 能有效地解决这一问题。

SNR 的讨论是从网络的泛化 ((Generalizability)) 开始的。对于 Deep Learning 而言, 泛化是一个重要的性能指标, 直觉上它与扰动 (Perturbation) 的影响有关。我们可以这样理解: 局部最小点附近如果是平坦 (flatness) 的话, 那么其泛化的性能将较好, 反之, 若是不平坦 (sharpness) 的话, 稍微一点变动, 将产生较大变化, 则其泛化性能就不好。因此, 我们可以从网络对抗扰动的性能入手来提升网络的泛化能力。

2.5.2.3 范数的应用

2.5.2.3.1 矩阵的谱半径

2.5.2.3.1.1 什么是谱半径

Definition 28 (矩阵的谱半径) 设 $A \in \mathbb{C}^{n \times n}$ 的 n 个特征值为 $\lambda_1, \lambda_2, \dots, \lambda_n$, 称

$$\rho(A) = \max_i |\lambda_i|$$

为 **谱半径 (Spectral Radius)**.

提示:

1. 矩阵 \mathbf{A} 的谱半径是正实数
 2. 矩阵 \mathbf{A} 的谱半径是矩阵的特征值的模值, 且为最大的那个
 3. 矩阵 \mathbf{A} 的 2-范数是 $\mathbf{A}^H \mathbf{A}$ 的最大特征值的平方根
 4. 矩阵 \mathbf{A} 的谱半径不超过矩阵的范数 $\|\mathbf{A}\|$
-

2.5.2.3.1.2 性质

1. 设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 则对于 $\mathbb{C}^{n \times n}$ 上的任意矩阵范数 $\|\cdot\|_M$, 都有

$$\rho(\mathbf{A}) \leq \|\mathbf{A}\|_M$$

提示: 取与矩阵范数 $\|\cdot\|_M$ 相容的向量范数 $\|\cdot\|_V$, 设矩阵 \mathbf{A} 的特征值 λ 对应的特征向量为 \mathbf{x} , 则

$$|\lambda| \|\mathbf{x}\|_V = \|\lambda \mathbf{x}\|_V = \|\mathbf{A} \mathbf{x}\|_V \leq \|\mathbf{A}\|_M \|\mathbf{x}\|_V$$

由 $\mathbf{x} \neq \mathbf{0}$ 得, $\rho(\mathbf{A}) \leq \|\mathbf{A}\|_M$.

2. 设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 则 $\rho(\mathbf{A}^k) = [\rho(\mathbf{A})]^k$
3. 设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 则 $\|\mathbf{A}\|_2 = \rho^{1/2}(\mathbf{A}^H \mathbf{A}) = \rho^{1/2}(\mathbf{A} \mathbf{A}^H)$, 当 \mathbf{A} 为 Hermite 正定阵时, $\|\mathbf{A}\|_2 = \rho(\mathbf{A})$
4. 设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 则 $\forall \epsilon \in \mathbb{Z}^+$, 存在某种矩阵范数 $\|\cdot\|_M$ 使得 $\|\mathbf{A}\|_M \leq \rho(\mathbf{A}) + \epsilon$

2.5.2.3.2 矩阵非奇异性条件

设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 若对 $\mathbb{C}^{n \times n}$ 上的某种矩阵范数 $\|\cdot\|_M$ 有 $\|\mathbf{A}\| < 1$, 则矩阵 $\mathbf{I} - \mathbf{A}$ 非奇异, 且

$$\|(\mathbf{I} - \mathbf{A})^{-1}\|_M \leq \frac{\|\mathbf{I}\|_M}{1 - \|\mathbf{A}\|_M}$$

$$\|\mathbf{I} - (\mathbf{I} - \mathbf{A})^{-1}\|_M \leq \frac{\|\mathbf{A}\|_M}{1 - \|\mathbf{A}\|_M}$$

2.5.2.3.3 逆矩阵的摄动与条件数

设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 且非奇异, $\mathbf{B} \in \mathbb{C}^{n \times n}$ 若对 $\mathbb{C}^{n \times n}$ 上的某种矩阵范数 $\|\cdot\|_M$, 有 $\|\mathbf{A}^{-1} \mathbf{B}\|_M < 1$, 则

1. $\mathbf{A} + \mathbf{B}$ 非奇异
2. $\|\mathbf{I} - (\mathbf{I} + \mathbf{A}^{-1} \mathbf{B})^{-1}\|_M \leq \frac{\|\mathbf{A}^{-1} \mathbf{B}\|_M}{1 - \|\mathbf{A}^{-1} \mathbf{B}\|_M}$
3. $\frac{\|\mathbf{A}^{-1} - (\mathbf{A} + \mathbf{B})^{-1}\|_M}{\|\mathbf{A}\|_M} \leq \frac{\|\mathbf{A}^{-1} \mathbf{B}\|_M}{1 - \|\mathbf{A}^{-1} \mathbf{B}\|_M}$

2.5.2.3.3.1 矩阵的摄动

设 $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{n \times n}$, $\delta\mathbf{A} = (\delta a_{ij}) \in \mathbb{C}^{n \times n}$, 且 δa_{ij} 表示元素 a_{ij} 的误差, 称矩阵 $\delta\mathbf{A}$ 为 \mathbf{A} 的 **摄动矩阵**.

2.5.2.3.3.2 条件数

设 $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{n \times n}$ 非奇异, $\|\mathbf{A}\|_M$ 为 $\mathbb{C}^{n \times n}$ 上的某种矩阵范数, 则称

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\|_M \|\mathbf{A}^{-1}\|_M$$

为矩阵 \mathbf{A} 的 **条件数**.

对应地, 当 $\|\mathbf{A}^{-1}\delta\mathbf{A}\|_M < 1$ 时有

1. $\|\mathbf{I} - (\mathbf{I} + \mathbf{A}^{-1}\delta\mathbf{A})^{-1}\|_M \leq \frac{\text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|_M}{\|\mathbf{A}\|_M}}{1 - \text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|_M}{\|\mathbf{A}\|_M}}$
2. $\frac{\|\mathbf{A}^{-1} - (\mathbf{A} + \delta\mathbf{A})^{-1}\|_M}{\|\mathbf{A}\|_M} \leq \frac{\text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|_M}{\|\mathbf{A}\|_M}}{1 - \text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|_M}{\|\mathbf{A}\|_M}}$

2.5.3 矩阵分析及应用

2.5.3.1 矩阵级数

2.5.3.1.1 概念

2.5.3.2 矩阵序列

2.5.3.2.1 概念理解

2.5.3.2.2 矩阵序列收敛与矩阵收敛

2.5.3.3 矩阵函数

2.5.3.3.1 概念

2.5.3.3.2 求解方法

- 方法 1: 待定系数法;
- 方法 2: 数项级数求和法, 写成矩阵级数求和形式, 计算 \mathbf{A}^k , 代入求解;
- 方法 3: 对角矩阵法, 根据 $\mathbf{A} = \mathbf{P}\Lambda\mathbf{P}^{-1}$, 有 $\mathbf{A}^k = \mathbf{P}\Lambda^k\mathbf{P}^{-1}$ 可知 $f(\mathbf{A}) = \mathbf{P}f(\Lambda)\mathbf{P}^{-1}$, 其中 $f(\Lambda) = (f(\Lambda_{ii})) = (f(\lambda_i))$ 据此可求;
- 方法 4: Jordan 标准型法, 对于不能化为对角阵的矩阵, 采用此方法, 注意此时 $f(\Lambda) \neq (f(\Lambda_{ii}))$

2.5.3.3.2.1 待定系数法

2.5.3.3.2.2 数项级数求和法

2.5.3.3.2.3 对角矩阵法

根据 $A = P\Lambda P^{-1}$, 有 $A^k = P\Lambda^k P^{-1}$ 可知 $f(A) = Pf(\Lambda)P^{-1}$, 其中 $f(\Lambda) = (f(\Lambda_{ii})) = (f(\lambda_i))$

2.5.3.3.2.4 Jordan 标准型法

对于不能化为对角阵的, 采用 Jordan 标准形法来求

$$f(A) = \sum_{k=0}^{\infty} c_k A^k = \sum_{k=0}^{\infty} c_k P J^k P^{-1} = P \left(\sum_{k=0}^{\infty} c_k J^k \right) P^{-1}$$

即

$$f(A) = P \begin{bmatrix} f(J_1) & & & \\ & \ddots & & \\ & & f(J_s) & \end{bmatrix} P^{-1}$$

其中

$$f(J_i) = \begin{bmatrix} f(\lambda_i) & \frac{f^{(1)}(\lambda_i)}{1!} & \frac{f^{(2)}(\lambda_i)}{2!} & \cdots & \frac{f^{(m_i-1)}(\lambda_i)}{(m_i-1)!} \\ f(\lambda_i) & \frac{f^{(1)}(\lambda_i)}{1!} & \ddots & & \vdots \\ & \ddots & \ddots & \frac{f^{(2)}(\lambda_i)}{2!} & \\ & & f(\lambda_i) & \frac{f^{(1)}(\lambda_i)}{1!} & \\ & & & f(\lambda_i) & \end{bmatrix}$$

其中, m_i 为 Jordan 块 J_i 的阶数, 也是特征值 λ_i 的重数.

提示: 只需要求出第一行, 斜向右下顺着写即可.

2.5.3.3 总结

- $e^z = 1 + \frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots$
- $\cos z = 1 - \frac{z^2}{2!} + \frac{z^4}{4!} - \cdots$
- $\sin z = \frac{z}{1!} - \frac{z^3}{3!} + \frac{z^5}{5!} - \cdots$
- $\frac{1}{1-z} = \sum_{n=0}^{\infty} z^n, |z| < 1$
- $e^{jz} = \cos z + j \sin z$
- $\cos z = \frac{e^{jz} + e^{-jz}}{2}$
- $\sin z = \frac{e^{jz} - e^{-jz}}{2}$
- $\cos(-z) = \cos z$

- $\sin(-z) = -\sin z$
- $e^A = 1 + \frac{A}{1!} + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$
- $\cos A = 1 - \frac{A^2}{2!} + \frac{A^4}{4!} - \dots$
- $\sin A = \frac{A}{1!} - \frac{A^3}{3!} + \frac{A^5}{5!} - \dots$
- $(I - A)^{-1} = \sum_{n=0}^{\infty} A^n$
- $e^{jA} = \cos A + j \sin A$
- $\cos A = \frac{e^{jA} + e^{-jA}}{2}$
- $\sin A = \frac{e^{jA} - e^{-jA}}{2}$
- $\cos(-A) = \cos A$
- $\sin(-A) = -\sin A$
- 当 $AB \neq BA$ 时, $e^A e^B \neq e^B e^A \neq e^{AB}$
- 当 $AB = BA$ 时, $e^A e^B = e^B e^A = e^{AB}$

2.5.3.4 矩阵微分与积分

2.5.3.4.1 概念

2.5.3.5 总结

2.5.3.5.1 概念对比

- 函数矩阵: 矩阵 $A^{(k)}$ 的元素是关于变量 k 的函数;
- 矩阵函数: 以矩阵 A 为自变量的函数;
- 矩阵序列: $A^{(k)} = (a_{ij}^{(k)})_{m \times n} \in \mathbb{C}^{m \times n}$, 其中 $a_{ij}^{(k)}$ 为关于 k 的函数;
- 矩阵级数: $\sum_{k=0}^{+\infty} A^{(k)} = A^{(0)} + A^{(1)} + \dots$
- 矩阵序列收敛: $A^{(k)} \rightarrow A (k \rightarrow +\infty)$, 其中 $A^{(k)}$ 为关于 k 的函数矩阵, $A = (a_{ij})_{m \times s}$ 为常数矩阵;
- 收敛矩阵: $A^k \rightarrow O (k \rightarrow +\infty)$, 其中 $A^k = AA \cdots A$ 表示 A 的 k 次幂, O 为零矩阵;

2.5.3.5.2 收敛定理

- 矩阵序列收敛: $A^{(k)} \rightarrow O$ 的充要条件是 $\|A^{(k)}\| \rightarrow 0$, 其中, $\|\cdot\|$ 为 $\mathbb{C}^{m \times n}$ 上的任意范数.
- 矩阵序列收敛: $A^{(k)} \rightarrow A$ 的充要条件是 $\|A^{(k)} - A\| \rightarrow 0$, 其中, $\|\cdot\|$ 为 $\mathbb{C}^{m \times n}$ 上的任意范数.
- 矩阵收敛: $A^k \rightarrow O$ 的充要条件是 $\rho(A) < 1$, 其中, $\rho(A)$ 为 A 的谱半径.
- 矩阵级数收敛: $\sum_{k=0}^{+\infty} A^{(k)} = A^{(0)} + A^{(1)} + A^{(2)} + \dots$ 绝对收敛的充要条件是正项级数 $\sum_{k=0}^{\infty} \|A^{(k)}\|$ 收敛.
- 矩阵幂级数收敛: $\sum_{k=0}^{+\infty} A^k = I + A^1 + A^2 + \dots$ 收敛的充要条件是 A 收敛, 且幂级数收敛于 $(I - A)^{-1}$
- .

2.5.4 矩阵分解

2.5.4.1 简介

- 三角分解
- 满秩分解
- 正交三角分解
- 奇异值分解

有关低秩分解参见

矩阵的满秩分解和奇异值分解及 QR 分解, 在广义逆理论中有广泛的应用, 近年来成为求解各类最小二乘方问题和最优化问题的主要数学工具.

2.5.4.2 三角分解

三角分解 (*Triangular Decomposition*) 是把一个矩阵 $A_{n \times n}$ 分解为一个下三角矩阵 (*Lower Triangular Matrix*) L 与一个上三角矩阵 (*Upper Triangular Matrix*) U 乘积的分解, 也可分解为一个单位下三角矩阵 (*Unit Lower Triangular Matrix*) L 与一个对角阵 (*Diagonal Matrix*) D 及一个单位上三角矩阵 (*Unit Upper Triangular Matrix*) U 乘积.

Definition 29 (三角分解) 称满足如下形式的矩阵分解为 **三角分解**:

$$A_{n \times n} = L_{n \times n} U_{n \times n}$$

$$A_{n \times n} = L_{n \times n} D_{n \times n} U_{n \times n}$$

2.5.4.2.1 引言

考虑方程组 $Ax = b$, 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

若 A 为下三角矩阵, 则有

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 \\ a_{21}x_1 + a_{22}x_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

若 A 为上三角矩阵, 则有

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{nn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

可见当 A 为下三角矩阵或上三角矩阵时, 方程组可以很容易求解. 下面设法将 A 化为上/下三角矩阵.

2.5.4.2.2 高斯消元过程

2.5.4.2.2.1 一般消元过程

以上三角矩阵为例, 记 $A^{(1)} = A$, 且 $A_{ij}^{(k)} = a_{ij}^{(k)}$ 采用初等行变换:

1. 将 $A^{(1)}$ 的第 1 列元素除第一个元素 $a_{11}^{(1)}$ 外, 其余元素都化为 0, 显然若 $a_{11}^{(1)} \neq 0$, 可以分别以 $-\frac{a_{21}^{(1)}}{a_{11}^{(1)}}, -\frac{a_{31}^{(1)}}{a_{11}^{(1)}}, \dots, -\frac{a_{n1}^{(1)}}{a_{11}^{(1)}}$ 乘以第 1 行元素, 分别加到第 $2, 3, \dots, n$ 行得

$$A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix} \xrightarrow[i=2, \dots, n]{r_i - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} r_1}$$

$$A^{(2)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} - \frac{a_{21}^{(1)} a_{12}^{(1)}}{a_{11}^{(1)}} & a_{23}^{(1)} - \frac{a_{21}^{(1)} a_{13}^{(1)}}{a_{11}^{(1)}} & \cdots & a_{2n}^{(1)} - \frac{a_{21}^{(1)} a_{1n}^{(1)}}{a_{11}^{(1)}} \\ 0 & a_{32}^{(1)} - \frac{a_{31}^{(1)} a_{12}^{(1)}}{a_{11}^{(1)}} & a_{33}^{(1)} - \frac{a_{31}^{(1)} a_{13}^{(1)}}{a_{11}^{(1)}} & \cdots & a_{3n}^{(1)} - \frac{a_{31}^{(1)} a_{1n}^{(1)}}{a_{11}^{(1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} - \frac{a_{n1}^{(1)} a_{12}^{(1)}}{a_{11}^{(1)}} & a_{n3}^{(1)} - \frac{a_{n1}^{(1)} a_{13}^{(1)}}{a_{11}^{(1)}} & \cdots & a_{nn}^{(1)} - \frac{a_{n1}^{(1)} a_{1n}^{(1)}}{a_{11}^{(1)}} \end{bmatrix}$$

若记 $(L_{i1}^{(1)})^{-1} = l_{i1}^{(1)} = -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, $L_{i1}^{(1)} = l_{i1}^{(1)} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, 即

$$(L^{(1)})^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} & 1 & 0 & \cdots & 0 \\ -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}^{(1)}}{a_{11}^{(1)}} & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad L^{(1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{a_{21}^{(1)}}{a_{11}^{(1)}} & 1 & 0 & \cdots & 0 \\ \frac{a_{31}^{(1)}}{a_{11}^{(1)}} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}^{(1)}}{a_{11}^{(1)}} & 0 & 0 & \cdots & 1 \end{bmatrix}$$

则有

$$A^{(2)} = (L^{(1)})^{-1} A^{(1)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

2. 将 $A^{(2)}$ 的第 2 列元素除前 2 个元素 $a_{12}^{(2)}, a_{22}^{(2)}$ 外, 其余元素都化为 0, 显然若 $a_{22}^{(2)} \neq 0$, 可以分别以 $-\frac{a_{32}^{(2)}}{a_{22}^{(2)}}, -\frac{a_{42}^{(2)}}{a_{22}^{(2)}}, \dots, -\frac{a_{n2}^{(2)}}{a_{22}^{(2)}}$ 乘以第 2 行元素, 分别加到第 $3, \dots, n$ 行得

$$A^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \xrightarrow[i=3, \dots, n]{r_i - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} r_2}$$

$$\mathbf{A}^{(3)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(2)} - \frac{a_{32}^{(2)}a_{23}^{(2)}}{a_{22}^{(2)}} & \cdots & a_{3n}^{(2)} - \frac{a_{32}^{(2)}a_{2n}^{(2)}}{a_{22}^{(2)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(2)} - \frac{a_{n2}^{(2)}a_{23}^{(2)}}{a_{22}^{(2)}} & \cdots & a_{nn}^{(2)} - \frac{a_{n2}^{(2)}a_{2n}^{(2)}}{a_{22}^{(2)}} \end{bmatrix}$$

若记 $(\mathbf{L}_{i2}^{(2)})^{-1} = l_{i2}^{(2)} = -\frac{a_{i2}^{(2)}}{a_{22}^{(2)}}$, $\mathbf{L}_{i2}^{(2)} = l_{i2}^{(2)} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}$, 即

$$(\mathbf{L}^{(2)})^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -\frac{a_{32}^{(2)}}{a_{22}^{(2)}} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\frac{a_{n2}^{(2)}}{a_{22}^{(2)}} & 0 & \cdots & 1 \end{bmatrix}, \quad \mathbf{L}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \frac{a_{32}^{(2)}}{a_{22}^{(2)}} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{a_{n2}^{(2)}}{a_{22}^{(2)}} & 0 & \cdots & 1 \end{bmatrix}$$

则有

$$\mathbf{A}^{(3)} = (\mathbf{L}^{(2)})^{-1} \mathbf{A}^{(2)} = \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} & \cdots & a_{1n}^{(3)} \\ 0 & a_{22}^{(3)} & a_{23}^{(3)} & \cdots & a_{2n}^{(3)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{bmatrix}$$

3. 重复如上步骤, 最终可将矩阵 \mathbf{A} 化为上三角矩阵.

$$\mathbf{A}^{(n)} = (\mathbf{L}^{(n-1)})^{-1} \mathbf{A}^{(n-1)} = \begin{bmatrix} a_{11}^{(n)} & a_{12}^{(n)} & a_{13}^{(n)} & \cdots & a_{1n}^{(n)} \\ 0 & a_{22}^{(n)} & a_{23}^{(n)} & \cdots & a_{2n}^{(n)} \\ 0 & 0 & a_{33}^{(n)} & \cdots & a_{3n}^{(n)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix}$$

警告: 上述消元过程可以执行的充要条件是矩阵 \mathbf{A} 的各阶顺序主子式不为零, 即

$$\Delta_k = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{kk}^{(k)} \neq 0, (k = 1, 2, \dots, n).$$

2.5.4.2.2.2 主元素选取

若矩阵 \mathbf{A} 的各阶顺序主子式不全为零, 则可以通过选取主元素进行消元, 核心思想是通过初等变换, 使得矩阵的顺序主子式不为零, 主元素选取包含:

- 列主元素法: 在矩阵的某列中选取模值最大的作为新的对角元素, 选取范围为对角线元素以下的各元素.
- 行主元素法: 在矩阵的某行中选取模值最大的作为新的对角元素, 选取范围为对角线元素以后的各元素.
- 全主元素法: 若某列元素均较小或某行元素均较小时, 可在各行各列中选取模值最大者作为对角元素.

此时, 可以通过初等变换 P 将矩阵 A 变为各阶顺序主子式不为零等矩阵 PA (行) AP (列).

注解: 如对于矩阵

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

易知 $\Delta_1 = 0$

1. 采用列主元素法, 选取 $a_{21} = 2$ 作为主元素, 即交换前两行, 记对应初等变换为 P_1 , 则

$$P_1 A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

1. 采用行主元素法, 选取 $a_{12} = 2$ 作为主元素, 即交换前两列, 记对应初等变换为 P_2 , 则

$$AP_2 = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 2 \\ 1 & 2 & 2 \\ 2 & 0 & 1 \end{bmatrix}$$

2.5.4.2.3 变元求解

假设矩阵 A 可以分解为下三角矩阵 L 与上三角矩阵 U 的乘积, 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{22} & \cdots & u_{2n} \\ \ddots & \ddots & \vdots \\ u_{nn} \end{bmatrix}$$

则可得以下方程组

$$\begin{cases} a_{11} = l_{11}u_{11} \\ a_{12} = l_{11}u_{12} \\ \vdots \\ a_{1n} = l_{11}u_{1n} \end{cases}, \begin{cases} a_{21} = l_{21}u_{11} \\ a_{22} = l_{21}u_{12} + l_{22}u_{22} \\ \vdots \\ a_{2n} = l_{21}u_{1n} + l_{22}u_{2n} \end{cases}, \dots, \begin{cases} a_{n1} = l_{n1}u_{11} \\ a_{n2} = l_{n1}u_{12} + l_{n2}u_{22} \\ \vdots \\ a_{nn} = l_{n1}u_{1n} + l_{n2}u_{2n} + \cdots + l_{nn}u_{nn} \end{cases}$$

观察易知上述方程组没有唯一解, 而当 L 为单位下三角矩阵时 ($l_{11} = l_{22} = \cdots = l_{nn} = 1$) 或当 U 为单位上三角矩阵时 ($u_{11} = u_{22} = \cdots = u_{nn} = 1$), 方程组有唯一解.

提示: 如对于二阶矩阵, 显然有

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ & u_{22} \end{bmatrix} = \begin{bmatrix} l_{11}u_{11} & l_{11}u_{12} \\ l_{21}u_{11} & l_{22}u_{22} \end{bmatrix}$$

得方程组

$$\begin{cases} a_{11} = l_{11}u_{11} \\ a_{12} = l_{11}u_{12} \\ a_{21} = l_{21}u_{11} \\ a_{22} = l_{21}u_{12} + l_{22}u_{22} \end{cases}$$

上述方程组显然没有唯一解, 故分解不唯一, 而当 \mathbf{L} 为单位下三角矩阵时 ($l_{11} = l_{22} = 1$) 或当 \mathbf{U} 为单位上三角矩阵时 ($u_{11} = u_{22} = 1$), 方程组有唯一解. 即

$$\mathbf{L} = \begin{bmatrix} 1 & \\ \frac{a_{21}}{a_{11}} & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} a_{11} & a_{12} \\ & a_{22} - \frac{a_{21}a_{12}}{a_{11}} \end{bmatrix}$$

或者

$$\mathbf{L} = \begin{bmatrix} a_{11} & \\ a_{21} & a_{22} - \frac{a_{21}a_{12}}{a_{11}} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} \\ & 1 \end{bmatrix}$$

2.5.4.2.4 LU 分解与 LDU 分解

2.5.4.2.4.1 概念与性质

如上所述, 易知

$$\mathbf{A} = \mathbf{A}^{(1)} = \mathbf{L}^{(1)}\mathbf{A}^{(2)} = \mathbf{L}^{(1)}\mathbf{L}^{(2)}\mathbf{A}^{(3)} = \cdots = \mathbf{L}^{(1)}\mathbf{L}^{(2)}\cdots\mathbf{L}^{(n-1)}\mathbf{A}^{(n)}$$

容易求得

$$\mathbf{L} = \mathbf{L}^{(1)}\mathbf{L}^{(2)}\cdots\mathbf{L}^{(n-1)} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix}$$

令 $\mathbf{U} = \mathbf{A}^{(n)}$, 则有 **LU 分解**

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

其中:

- $\mathbf{L}^{(1)}, \mathbf{L}^{(2)}, \dots, \mathbf{L}^{(n-1)}$ 称为 **Frobenius 矩阵**
- $l_{ij} = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}, j < i$
- \mathbf{L} 为下三角矩阵
- \mathbf{U} 为上三角矩阵
- **LU 分解一般不唯一**, 当 \mathbf{L} 为单位下三角矩阵, 或者 \mathbf{U} 为单位上三角矩阵时, **LU 分解唯一**

若矩阵 \mathbf{A} 可以分解为单位下三角矩阵 \mathbf{L} 和单位上三角矩阵 \mathbf{U} 和对角矩阵 \mathbf{D} 的乘积, 即

$$\mathbf{A}_{n \times n} = \mathbf{L}_{n \times n} \mathbf{D}_{n \times n} \mathbf{U}_{n \times n}$$

则称上述分解为矩阵 \mathbf{A} 的 **LDU 分解**.

其中 $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n)$ 为对角元素不为零的 n 阶对角阵, 且 $d_k = \frac{\Delta_k}{\Delta_{k-1}}, k = 1, 2, \dots, n, \Delta_k$ 为 \mathbf{A} 的 k 阶顺序主子式, 且规定 $\Delta_0 = 1$.

提示:

1. n 阶非奇异矩阵 A 有三角分解 LU 或 LDU 的充要条件是 A 的顺序主子式 $\Delta_k \neq 0, (k = 1, 2, \dots, n)$;
 2. 对于任意可逆矩阵 A , 存在置换矩阵 P 使得 PA 的所有顺序主子式全不为零.
-

2.5.4.2.4.2 计算方法

- 方法 1: 高斯消元法
- 方法 2: 变元求解法

2.5.4.2.4.3 高斯消元法

高斯消元法: 采用上述消元过程求解即可, 若矩阵 A 的各阶顺序主子式不全为零, 则可以通过选取主元素进行消元.

注解: 举个例子: 求如下矩阵的 LU 和 LDU 分解

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

解: 采用列主元素法, 选取 $a_{21} = 2$ 作为主元素, 即交换前两行, 记对应初等变换为 P_1 , 则

$$A^{(2)} = P_1 A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

对上述矩阵进行消元, 可见 $A^{(2)}$ 的第一列除对角元素外, 其它均为零, 无需进行消元, 下面对 $A^{(2)}$ 的第 2 列进行消元, 有

$$\begin{aligned} \left(L^{(2)}\right)^{-1} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, L_{32}^{(2)} = l_{32}^{(2)} = -\frac{a_{32}^{(2)}}{a_{22}^{(2)}} = -1 \\ \left(L^{(2)}\right)^{-1} A^{(2)} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & -1 \end{bmatrix} = A^{(3)} \end{aligned}$$

至此已将矩阵 A 转为上三角矩阵, 且 $A = P_1^{-1} A^{(2)} = P_1^{-1} L^{(2)} A^{(3)}$

对于 $A = LU$ 分解, 有

$$\begin{aligned} L &= P_1^{-1} L^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \\ U &= A^{(3)} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & -1 \end{bmatrix} \end{aligned}$$

对于 $A = LDU$ 分解, 有

$$\begin{aligned} \mathbf{L} &= \mathbf{P}_1^{-1} \mathbf{L}^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \\ \mathbf{D} &= \begin{bmatrix} 2 & & \\ & 2 & \\ & & -1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 1/2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

则

$$A = LDU = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & & \\ & 2 & \\ & & -1 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

提示: 上述求解过程可通过如下方法简化:

1. 将矩阵 A 与单位矩阵 I 组成新矩阵 $[A|I]$
2. 采用高斯消元法, 每次仅对待消元的子矩阵进行初等行变换, 如第二次消元时, 不再对第一列的元素进行初等变换
3. 如此循环往复, 将矩阵 A 的主对角元素下的元素全部化为 0, 矩阵 A 化为上三角矩阵 U , 矩阵 I 变为单位下三角矩阵的逆 $(L^{(n-1)} \dots L^{(2)} L^{(1)})^{-1} = L^{-1}$.

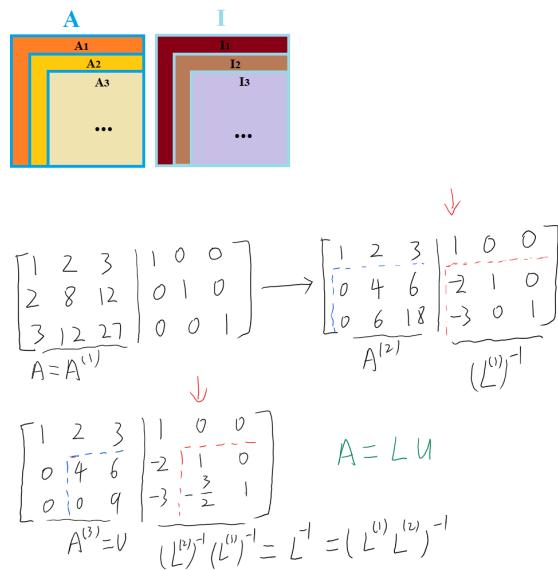


图 2.10: LU 分解高斯消元简化
LU 分解高斯消元简化, 三阶矩阵为例.

2.5.4.2.4.4 变元求解法

根据所求矩阵阶数, 设 L 为单位下三角矩阵, U 为上三角矩阵, 求解方程组 $A = LU$ 即可. 对于四阶矩阵, 求解方法如图 2.11 所示.

$$\begin{aligned} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \\ &= \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} & l_{21}u_{14} + u_{24} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} & l_{31}u_{14} + l_{32}u_{24} + u_{34} \\ l_{41}u_{11} & l_{41}u_{12} + l_{42}u_{22} & l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} \end{bmatrix} \end{aligned}$$

● → ● → ● → ● → ● → ● → ●

按颜色顺序依次计算

图 2.11: LU 分解变元求解法
LU 分解变元求解法, 四阶矩阵为例.

2.5.4.2.4.5 代码实现

在 Matlab 中可以通过函数 `lu` 求解.

上述例子的 matlab 代码求解如下:

```
>> A = [0 2 2; 2 1 2; 0 2 1]
>> [L,U] = lu(A)

L =
0     1     0
1     0     0
0     1     1

U =
2     1     2
0     2     2
0     0    -1
>> [L,U, P] = lu(A)

L =
1     0     0
```

(下页继续)

(续上页)

$$\begin{matrix} 0 & 1 & 0 \\ 0 & 1 & 1 \end{matrix}$$

 $\mathbf{U} =$

$$\begin{matrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & -1 \end{matrix}$$

 $\mathbf{P} =$

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$$

2.5.4.2.5 其它三角分解

根据上述介绍, 对 LDU 分解的形式进行一下组合和限定, 可以得到 Doolittle 分解, Crout 分解, 和 Cholesky 分解.

2.5.4.2.5.1 Crout 分解

设矩阵 A 有唯一的 LDU 分解, 将 LD 结合起来, 并用 \hat{L} 表示, 则称

$$A = (LD)U = \hat{L}U$$

为矩阵 A 的 **Crout 分解**. 可知 Crout 分解中, U 为单位上三角矩阵.

2.5.4.2.5.2 求解方法

上述变元求解法与高斯消元法即可求解.

2.5.4.2.5.3 Doolittle 分解

设矩阵 A 有唯一的 LDU 分解, 将 LD 结合起来, 并用 \hat{L} 表示, 则称

$$A = L(DU) = L\hat{U}$$

为矩阵 A 的 **Doolittle 分解**. 可知 Doolittle 分解中, L 为单位下三角矩阵.

2.5.4.2.5.4 求解方法

上述变元求解法与高斯消元法即可求解.

2.5.4.2.5.5 Cholesky 分解

设矩阵 A 为实对称正定矩阵, 有唯一的 LDU 分解, 将 L 与 D 的平方根结合起来, 记为 G 为下三角矩阵, 则称

$$A = GG^T$$

为矩阵 A 的 **Cholesky 分解** (或 **平方根分解, 对称三角分解**).

易知, $G = L\bar{D}$ 为下三角矩阵, 其中 $D = \text{diag}(d_1, d_2, \dots, d_n)$, $\bar{D} = \text{diag}(\sqrt{d_1}, \sqrt{d_2}, \dots, \sqrt{d_n})$, $d_i > 0$.

提示: 设矩阵 A 为实对称正定矩阵, 则有 $\Delta_k > 0$, 于是 A 有唯一的 LDU 分解, 其中, $D = \text{diag}(d_1, d_2, \dots, d_n)$, 记 $\bar{D} = \text{diag}(\sqrt{d_1}, \sqrt{d_2}, \dots, \sqrt{d_n})$, 则有

$$A = LDU = L\bar{D}^2U$$

由 $A^T = A$ 知

$$U^T D L^T = LDU$$

由分解的唯一性知 $L = U^T$, $U = L^T$

所以有

$$A = LDU = LDL^T$$

进一步地, 有

$$A = LDL^T = L\bar{D}\bar{D}L^T = GG^T$$

其中, $G = L\bar{D}$.

2.5.4.2.5.6 求解方法

上述变元求解法与高斯消元法即可求解,

- 可以先求 Doolittle 分解;
- Doolittle 分解中的 L 即为 Cholesky 分解中的 L ;
- Doolittle 分解中的 \hat{U} 对角线元素构成 D ;
- 取 L 与 D 的平方根的乘积 $G = L\bar{D}$ 即为所求.

注解: 举个例子, 求以下矩阵的 Cholesky 分解

$$\begin{bmatrix} 5 & 2 & -4 \\ 2 & 1 & 2 \\ -4 & -2 & 5 \end{bmatrix}.$$

解: 先求矩阵的 Doolittle 分解, 设 L 和 \hat{U} 如下

$$\begin{bmatrix} 5 & 2 & -4 \\ 2 & 1 & 2 \\ -4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 1 & & \\ l_{21} & 1 & \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ & u_{22} & u_{23} \\ & & u_{33} \end{bmatrix}$$

求解得到 $l_{21} = \frac{2}{5}, l_{31} = -\frac{4}{5}, l_{32} = -2, u_{11} = 5, u_{12} = 2, u_{13} = -4, u_{22} = \frac{1}{5}, u_{23} = -\frac{2}{5}, u_{33} = 1$, 则有

$$L = \begin{bmatrix} 1 & & \\ \frac{2}{5} & 1 & \\ -\frac{4}{5} & -2 & 1 \end{bmatrix}, \hat{U} = \begin{bmatrix} 5 & 2 & -4 \\ \frac{1}{5} & -\frac{2}{5} & \\ & 1 \end{bmatrix}$$

易知 $D = \text{diag}(5, 1/5, 1)$, $\bar{D} = \text{diag}(\sqrt{5}, 1/\sqrt{5}, 1)$, 从而有

$$G = L\bar{D} = \begin{bmatrix} 1 & & \\ \frac{2}{5} & 1 & \\ -\frac{4}{5} & -2 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{5} & & \\ & \frac{1}{\sqrt{5}} & \\ & & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{5} & 0 & 0 \\ \frac{2\sqrt{5}}{5} & \frac{1}{\sqrt{5}} & \\ -\frac{4\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 1 \end{bmatrix}$$

2.5.4.2.5.7 代码实现

在 Matlab 中可以通过函数 `chol` 求解.

```
>> A = [5 2 -4; 2 1 -2; -4 -2 5]

A =
5     2    -4
2     1    -2
-4    -2     5

>> G = chol(A, 'lower')

G =
2.2361      0      0
0.8944  0.4472      0
-1.7889 -0.8944  1.0000

>> G*G'
```

(下页继续)

(续上页)

```
ans =
5.0000    2.0000   -4.0000
2.0000    1.0000   -2.0000
-4.0000   -2.0000    5.0000
```

2.5.4.3 满秩分解

2.5.4.3.1 什么是满秩分解

满秩分解 (*Full Rank Decomposition*) 是指将一矩阵分解为行满秩与列满秩的两个矩阵的乘积的分解.

Definition 30 (满秩分解) 设 $A \in \mathbb{C}_r^{m \times n} (r > 0)$, 若存在矩阵 $F \in \mathbb{C}_r^{m \times r} (r > 0)$ 和 $G \in \mathbb{C}_r^{r \times n} (r > 0)$ 使得

$$A_r^{m \times n} = F_r^{m \times r} G_r^{r \times n}$$

则称上述分解为 **满秩分解**. 其中, r 为矩阵 A, F, G 的秩.

2.5.4.3.2 求解方法

2.5.4.3.2.1 初等行变换法

对矩阵 $A \in \mathbb{C}_r^{m \times n} (r > 0)$ 进行满秩分解的步骤如下:

1. 仅使用初等行变换, 将 $A \in \mathbb{C}_r^{m \times n} (r > 0)$ 化为行阶梯标准型 $B \in \mathbb{C}_r^{m \times n} (r > 0)$;
2. 列满秩矩阵 $F_r^{m \times r}$ 取为矩阵 A 的前 r 列构成的 $m \times r$ 的矩阵;
3. 行满秩矩阵 $G_r^{r \times n}$ 取为矩阵 B 的前 r 行构成的 $r \times n$ 的矩阵.

提示:

行阶梯标准型又称 **Hermite 标准型**, 是指满足如下条件的矩阵 $B \in \mathbb{C}_r^{m \times n} (r > 0)$

1. B 的前 r 行中每一行至少含一个非零元素, 且第一个非零元素是 1, 后 $m - r$ 行元素均为零;
2. B 的前 r 列为单位阵 I_m 的前 r 列.

可见, Hermite 标准型矩阵具备如下形式:

$$B = \begin{bmatrix} I_{r \times r} & K_{r \times (n-r)} \\ O_{(m-r) \times r} & O_{(m-r) \times (n-r)} \end{bmatrix}$$

其中, K 为任意 $r \times (n - r)$ 矩阵.

如矩阵

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

注解: 举个例子, 求如下矩阵的满秩分解

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

1. 仅使用初等行变换将其化为 Hermite 标准形:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{r_2 \rightarrow r_1} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{r_3 \rightarrow r_2} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{r_4 \rightarrow r_3} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \mathbf{B}$$

可见 $r = 3$.

2. 列满秩矩阵 $\mathbf{F}_r^{m \times r}$ 取为矩阵 \mathbf{A} 的前 r 列构成的 $m \times r$ 的矩阵

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

3. 行满秩矩阵 $\mathbf{G}_r^{r \times n}$ 取为矩阵 \mathbf{B} 的前 r 行构成的 $r \times n$ 的矩阵

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

则有如下满秩分解:

$$\mathbf{A}_{4 \times 4} = \mathbf{F}_{4 \times 3} \mathbf{G}_{3 \times 4} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

2.5.4.4 正交三角分解

2.5.4.4.1 什么是正交三角分解

正交三角分解 (*Orthogonal Triangular Decomposition*), 是指将一矩阵分解为正交矩阵 \mathbf{Q} 与非奇异上三角矩阵 \mathbf{R} 两个矩阵的乘积的分解.

Definition 31 (正交三角分解) 设 $\mathbf{A} \in \mathbb{C}_{m \times n}$, 若存在矩阵 $\mathbf{Q} \in \mathbb{C}$ 和 $\mathbf{R} \in \mathbb{C}$ 使得

$$\mathbf{A}_{m \times n} = \mathbf{Q}_{m \times n} \mathbf{R}_{n \times n}$$

则称上述分解为 **正交三角分解**, 简称 **QR 分解**.

2.5.4.4.2 求解方法

2.5.4.4.2.1 Schmidt 正交化方法

对矩阵 $A \in \mathbb{C}^{m \times n}$ 进行 QR 分解的步骤如下:

1. 记矩阵 $A = (a_1, a_2, \dots, a_n)$, 其中 a_j 为矩阵 A 的第 j 列;
2. 正交化向量组 a_1, a_2, \dots, a_n , 得正交化后的向量组 b_1, b_2, \dots, b_n ;
3. 单位化向量组 b_1, b_2, \dots, b_n , 得单位化后的向量组 q_1, q_2, \dots, q_n , 其中 $q_j = \frac{1}{\|b_j\|} b_j$, $j = 1, 2, \dots, n$;
4. 将正交化的向量组按顺序组成矩阵 $Q = (q_1, q_2, \dots, q_n)$;
5. 计算上三角矩阵 $R = \text{diag}(|b_1|, |b_2|, \dots, |b_n|) \cdot C$;

其中,

$$C = \begin{bmatrix} 1 & k_{21} & \cdots & k_{n1} \\ & 1 & \cdots & k_{n2} \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

且 $k_{ij} = \frac{\langle a_i, b_j \rangle}{\langle b_j, b_j \rangle}$, $j < i$.

注解: 举个例子, 求如下矩阵的 QR 分解,

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

解: 由题意有,

令 $a_1 = (0, 2, 0)^T$, $a_2 = (2, 1, 2)^T$, $a_3 = (2, 2, 1)^T$

正交化得

$$b_1 = a_1 = (0, 2, 0)^T, k_{21} = \frac{\langle a_2, b_1 \rangle}{\langle b_1, b_1 \rangle} = \frac{1}{2}$$

$$b_2 = a_2 - k_{21}b_1 = (2, 1, 2)^T - \frac{1}{2}(0, 2, 0)^T = (2, 0, 2)^T, k_{32} = \frac{\langle a_3, b_2 \rangle}{\langle b_2, b_2 \rangle} = \frac{3}{4}, k_{31} = \frac{\langle a_3, b_1 \rangle}{\langle b_1, b_1 \rangle} = 1$$

$$b_3 = a_3 - k_{32}b_2 - k_{31}b_1 = (2, 2, 1)^T - \frac{3}{4}(2, 0, 2)^T - 1(0, 2, 0)^T = (\frac{1}{2}, 0, \frac{-1}{2})^T$$

单位化得

$$q_1 = \frac{(0, 2, 0)^T}{2} = (0, 1, 0)^T, q_2 = \frac{(2, 0, 2)^T}{2\sqrt{2}} = (\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})^T, q_3 = \frac{(\frac{1}{2}, 0, \frac{-1}{2})^T}{\sqrt{2}} = (\frac{\sqrt{2}}{2}, 0, \frac{-\sqrt{2}}{2})^T$$

从而有

$$C = \begin{bmatrix} 1 & k_{21} & k_{31} \\ 0 & 1 & k_{32} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} & 1 \\ 0 & 1 & \frac{3}{4} \\ 0 & 0 & 1 \end{bmatrix}$$

由 $\mathbf{R} = \text{diag}(|\mathbf{b}_1|, |\mathbf{b}_2|, |\mathbf{b}_3|) \cdot \mathbf{C}$ 知

$$\mathbf{R} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2\sqrt{2} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & 1 \\ 0 & 1 & \frac{3}{4} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2\sqrt{2} & \frac{3}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{\sqrt{2}}{2} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

2.5.4.4.2.2 Givens 变换方法

注解: 举个例子, 求如下矩阵的 QR 分解,

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

解: 由题意有,

记 $\mathbf{A}^{(1)} = \mathbf{A}$, 取 $\mathbf{A}^{(1)}$ 的第一列 $\mathbf{a}_1^{(1)} = (0, 2, 0)^T$, 构造 Givens 矩阵, 使得 $\mathbf{G}^{(1)} \mathbf{a}_1^{(1)} = |\mathbf{a}_1^{(1)}| \mathbf{e}_1$:

$$\text{取 } \mathbf{G}_{1,2}^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ 其中, } c = \frac{\xi_1}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{0}{\sqrt{0^2 + 2^2}} = 0, s = \frac{\xi_2}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{2}{\sqrt{0^2 + 2^2}} = 1.$$

从而 $\mathbf{G}_{1,2}^{(1)} \mathbf{a}_1^{(1)} = (2, 0, 0)^T = 2\mathbf{e}_1$ 的第二坐标分量变为 0, 又第三坐标分量为 0, 所以无需再构造 $\mathbf{G}_{1,3}^{(1)}$ 使第三坐标分量为 0, 当然也可以对 $(2, 0, 0)^T$ 继续构造

$$\mathbf{G}_{1,3}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ 其中, } c = \frac{\sqrt{\xi_1^2 + \xi_2^2}}{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}} = \frac{\sqrt{2^2 + 0^2}}{\sqrt{2^2 + 0^2 + 0^2}} = 1, s = \frac{\xi_3}{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}} = \frac{0}{\sqrt{2^2 + 0^2 + 0^2}} = 0, \text{ 可见 } \mathbf{G}_{1,3}^{(1)} \text{ 为单位阵.}$$

从而有

$$\begin{aligned} \mathbf{G}^{(1)} \mathbf{A}^{(1)} &= \mathbf{G}_{1,3}^{(1)} \mathbf{G}_{1,2}^{(1)} \mathbf{A}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 1 & 2 \\ 0 & -2 & -2 \\ 0 & 2 & 1 \end{bmatrix} \end{aligned}$$

对 $\mathbf{A}^{(2)} = \begin{bmatrix} -2 & -2 \\ 2 & 1 \end{bmatrix}$, 取其第一列 $\mathbf{a}_1^{(2)} = (-2, 2)^T$, 构造 Givens 矩阵, 使得 $\mathbf{G}^{(2)} \mathbf{a}_1^{(2)} = |\mathbf{a}_1^{(2)}| \mathbf{e}_1$:

$$\text{取 } \mathbf{G}_{1,2}^{(2)} = \begin{bmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}, \text{ 其中, } c = \frac{\xi_1}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{-2}{\sqrt{-2^2 + 2^2}} = \frac{-1}{\sqrt{2}}, s = \frac{\xi_2}{\sqrt{\xi_1^2 + \xi_2^2}} = \frac{2}{\sqrt{-2^2 + 2^2}} = \frac{1}{\sqrt{2}}.$$

从而有

$$\mathbf{G}^{(2)} \mathbf{A}^{(2)} = \mathbf{G}_{1,2}^{(2)} \mathbf{A}^{(2)} = \begin{bmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -2 & -2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2\sqrt{2} & \frac{3}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

最后, 令

$$\mathbf{G} = \begin{bmatrix} 1 & \\ & \mathbf{G}^{(2)} \end{bmatrix} \mathbf{G}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

则有

$$\mathbf{Q} = \mathbf{G}^T = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2\sqrt{2} & \frac{3\sqrt{2}}{2} \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2\sqrt{2} & \frac{3}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

2.5.4.4.2.3 Householder 变换方法

注解: 举个例子, 求如下矩阵的 QR 分解,

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

解: 由题意有,

记 $\mathbf{A}^{(1)} = \mathbf{A}$, 取 $\mathbf{A}^{(1)}$ 的第一列 $\mathbf{a}_1^{(1)} = (0, 2, 0)^T$, 构造 Householder 矩阵, 使得 $\mathbf{H}^{(1)} \mathbf{a}_1^{(1)} = |\mathbf{a}_1^{(1)}| \mathbf{e}_1$:

$$\mathbf{b}_1^{(1)} = \mathbf{a}_1^{(1)} - |\mathbf{a}_1^{(1)}| \mathbf{e}_1 = (0, 2, 0)^T - 2(1, 0, 0)^T = (-2, 2, 0)^T, \mathbf{u}_1^{(1)} = \frac{\mathbf{b}_1^{(1)}}{|\mathbf{b}_1^{(1)}|} = \frac{(-2, 2, 0)^T}{2\sqrt{2}} = \left(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)^T$$

$$\mathbf{u}_1^{(1)} \mathbf{u}_1^{(1)T} = \begin{bmatrix} \frac{1}{2} & \frac{-1}{2} & 0 \\ \frac{-1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H}^{(1)} = \mathbf{I} - 2\mathbf{u}_1^{(1)} \mathbf{u}_1^{(1)T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}^{(1)} \mathbf{A}^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

对 $\mathbf{A}^{(2)} = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$, 取其第一列 $\mathbf{a}_1^{(2)} = (2, 2)^T$, 构造 Householder 矩阵, 使得 $\mathbf{H}^{(2)} \mathbf{a}_1^{(2)} = |\mathbf{a}_1^{(2)}| \mathbf{e}_1$:

$$\mathbf{b}_1^{(2)} = \mathbf{a}_1^{(2)} - |\mathbf{a}_1^{(2)}| \mathbf{e}_1 = (2, 2)^T - 2\sqrt{2}(1, 0)^T = (2 - 2\sqrt{2}, 2)^T, \mathbf{u}_1^{(2)} = \frac{\mathbf{b}_1^{(2)}}{|\mathbf{b}_1^{(2)}|} = \frac{(2-2\sqrt{2}, 2)^T}{2\sqrt{4-2\sqrt{2}}} = \frac{(1-\sqrt{2}, 1)^T}{\sqrt{4-2\sqrt{2}}}$$

$$\mathbf{u}_1^{(2)} \mathbf{u}_1^{(2)T} = \frac{1}{4-2\sqrt{2}} \begin{bmatrix} 1-\sqrt{2} \\ 1 \end{bmatrix} \begin{bmatrix} 1-\sqrt{2} & 1 \end{bmatrix} = \frac{1}{4-2\sqrt{2}} \begin{bmatrix} 3-2\sqrt{2} & 1-\sqrt{2} \\ 1-\sqrt{2} & 1 \end{bmatrix}$$

$$\mathbf{H}^{(2)} = \mathbf{I} - 2\mathbf{u}_1^{(2)} \mathbf{u}_1^{(2)T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \frac{1}{2-\sqrt{2}} \begin{bmatrix} 3-2\sqrt{2} & 1-\sqrt{2} \\ 1-\sqrt{2} & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix}$$

$$\mathbf{H}^{(2)} \mathbf{A}^{(2)} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2\sqrt{2} & \frac{3\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$$

再令

$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{H}^{(2)} \end{bmatrix} \mathbf{H}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

从而

$$\mathbf{Q} = \mathbf{S}^T = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

且

$$\mathbf{R} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2\sqrt{2} & \frac{3}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

2.5.4.4.2.4 代码实现

matlab 代码

```
>> A = [0 2 2; 2 1 2; 0 2 1]
```

```
A =
```

```
0      2      2
2      1      2
0      2      1
```

```
>> [Q, R] = qr(A)
```

```
Q =
```

```
0      0.7071   -0.7071
-1.0000      0      0
0      0.7071    0.7071
```

(下页继续)

(续上页)

```
R =
-2.0000 -1.0000 -2.0000
  0     2.8284  2.1213
  0       0   -0.7071
```

2.5.4.5 奇异值分解

2.5.4.5.1 概念

2.5.4.5.1.1 矩阵的奇异值

Definition 32 (奇异值) 设 $A \in \mathbb{C}_r^{m \times n}$ ($r > 0$), $A^H A$ 的特征值为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq \lambda_{r+1} = \cdots = \lambda_n = 0$$

则称 $\sigma_i = \sqrt{\lambda_i}$, ($i = 1, 2, \dots, n$) 为 A 的 奇异值 (Singular Value); 当矩阵 A 为零矩阵时, 奇异值全为零.

- A 的奇异值的个数等于 A 的列数;
- A 的非零奇异值的个数等于 A 的秩;

Theorem 33 (奇异值对角化定理) 设 $A \in \mathbb{C}_r^{m \times n}$ ($r > 0$), 则存在 m 阶酉矩阵 U 和 n 阶酉矩阵 V , 使得

$$U^H A V = \begin{bmatrix} \Sigma & O \\ O & O \end{bmatrix}$$

其中, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, σ_i ($i = 1, 2, \dots, r$) 为矩阵 A 的全部非零特征值.

2.5.4.5.1.2 奇异值分解

奇异值分解 (Singular Value Decomposition) 是把一个矩阵 $A_{m \times n}$ 分解为一个酉矩阵 (unitary matrix) $U_{m \times m}$, 矩形对角矩阵 $S_{m \times n}$ 与一个酉矩阵 $V_{n \times n}$ 共轭转置乘积的分解.

Definition 34 (奇异值分解) 称满足如下形式的矩阵分解为 奇异值分解:

$$A_{m \times n} = U_{m \times m} S_{m \times n} V_{n \times n}^H$$

其中, $S_{m \times n} = \begin{bmatrix} \Sigma_{r \times r} & O_{r \times (n-r)} \\ O_{(m-r) \times r} & O_{(m-r) \times (n-r)} \end{bmatrix}$.

图 2.12 所示为奇异值分解示意图.

提示:

- 矩阵 A 的奇异值由 A 唯一确定;
- 矩阵 A 的奇异值分解不唯一, 因为 U, V 一般不唯一;

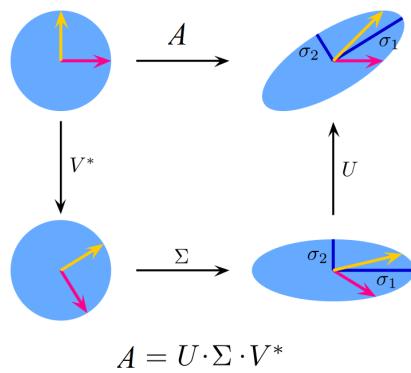


图 2.12: 奇异值分解示意图
奇异值分解示意图

- 矩阵 U 是 AA^H 的特征向量
- 矩阵 V 是 A^HA 的特征向量

注解: 奇异值分解证明

2.5.4.5.2 求解方法

设有矩阵 $A \in \mathbb{C}_r^{m \times n} (r > 0)$, 求其奇异值分解的步骤如下:

- 计算矩阵 $A^H A$ 及其特征值和特征向量, 秩 r ;
- 根据特征值 λ_i , 求奇异值 σ_i , 并写出矩阵 $\Sigma_r \times r$ 及矩形对角阵 $S_{m \times n}$;
- 根据特征向量, 标准化特征向量并将特征向量按列依奇异值顺序排成矩阵 V , 取前 r 列构成 V_1 ;
- 根据 $U_1 = AV_1\Sigma^{-1}$ 计算 U_1 ;
- 取与 U_1 正交的矩阵 U_2 , 合并得到 $U = [U_1|U_2]$;
- 最终求得矩阵 A 的奇异值分解 $A = USV^H$.

2.5.4.5.2.1 实例

求如下矩阵的 SVD 分解

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 2 & 2 \end{bmatrix}$$

解: 由题意有

$$A^H A = \begin{bmatrix} -1 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

1. 求得 $\mathbf{A}^H \mathbf{A}$ 特征值与特征向量分别为 $\lambda_1 = 9, \lambda_2 = 1$, 对应特征向量 $\mathbf{x}_1 = (1, -1)^T, \mathbf{x}_2 = (1, 1)^T$
2. 从而有奇异值 $\sigma_1 = 3, \sigma_2 = 1$, 于是有

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \end{bmatrix}, \quad S = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

3. 根据特征向量, 标准化特征向量并将特征向量按列依奇异值顺序排成矩阵, 由于秩 $r = 2$, 所以

$$\mathbf{V}_1 = \mathbf{V} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

4. 根据 $\mathbf{U}_1 = \mathbf{A}\mathbf{V}_1\Sigma^{-1}$ 计算 \mathbf{U}_1

$$\mathbf{U}_1 = \mathbf{A}\mathbf{V}\Sigma^{-1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{-1}{3\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{3\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{4}{3\sqrt{2}} & 0 \end{bmatrix}$$

5. 取与 \mathbf{U}_1 正交的矩阵 \mathbf{U}_2 , 合并得到 $\mathbf{U} = [\mathbf{U}_1 | \mathbf{U}_2]$

$$\mathbf{U}_2 = \begin{bmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \frac{-1}{3\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{2}{3} \\ \frac{1}{3\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{-2}{3} \\ \frac{4}{3\sqrt{2}} & 0 & \frac{1}{3} \end{bmatrix}$$

6. 最终求得矩阵 \mathbf{A} 的奇异值分解 $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^H$.

2.5.4.5.2.2 代码实现

matlab 代码

代码 2.3: demo_svd.m

```

1 >> A=[-1 0;0 1;2 2]
2
3 A =
4
5 -1      0
6 0      1
7 2      2
8
9 >> [U, S, V]=svd(A)
10
11 U =
12
13 -0.2357 -0.7071  0.6667
14  0.2357 -0.7071 -0.6667
15  0.9428 -0.0000  0.3333

```

(下页继续)

(续上页)

```

16
17
18 S =
19
20     3.0000      0
21     0      1.0000
22     0      0
23
24
25 V =
26
27     0.7071    0.7071
28     0.7071   -0.7071

```

例子 2

```

1 >> A=[1 0 1;0 1 1; 0 0 0]
2
3 A =
4
5     1      0      1
6     0      1      1
7     0      0      0
8
9 >> [U, S, V]=svd(A)
10
11 U =
12
13     0.7071   -0.7071      0
14     0.7071    0.7071      0
15     0          0      1.0000
16
17
18 S =
19
20     1.7321      0      0
21     0      1.0000      0
22     0          0      0
23
24
25 V =
26
27     0.4082   -0.7071    0.5774
28     0.4082    0.7071    0.5774
29     0.8165    0.0000   -0.5774
30
31 >> U*S*V

```

(下页继续)

(续上页)

```

32
33 ans =
34
35 0.2113 -1.3660 0.2989
36 0.7887 -0.3660 1.1154
37 0 0 0
38
39 >> U*S*V'
40
41 ans =
42
43 1.0000 -0.0000 1.0000
44 0.0000 1.0000 1.0000
45 0 0 0

```

2.5.4.5.3 奇异值分解应用

http://en.volupedia.org/wiki/Singular_value_decomposition

2.5.4.5.3.1 SVD 与谱分解

2.5.5 特征值估计

2.5.5.1 特征值估计

2.5.5.1.1 特征值的界

定理 1: 设 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{R}^{n \times n}$, 令 $M = \max_{1 \leq i, j \leq n} \frac{1}{2}|a_{ij} - a_{ji}|$, 若 λ 表示 \mathbf{A} 的任一特征值, 则 λ 的虚部 $\text{Im}(\lambda)$ 满足不等式

$$|\text{Im}(\lambda)| \leq M \sqrt{\frac{n(n-1)}{2}}.$$

定理 2: 设 $\mathbf{A} \in \mathbb{C}^{n \times n}$, 则 \mathbf{A} 的任一特征值 λ 满足

$$\begin{aligned} |\lambda| &\leq \|\mathbf{A}\|_{m_\infty}, \\ |\text{Re}(\lambda)| &\leq \frac{1}{2} \|\mathbf{A} + \mathbf{A}^H\|_{m_\infty}, \\ |\text{Im}(\lambda)| &\leq \frac{1}{2} \|\mathbf{A} - \mathbf{A}^H\|_{m_\infty}. \end{aligned}$$

定理 3: 设 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{C}^{n \times n}$ 的特征值为 $\lambda_1, \lambda_2, \dots, \lambda_n$, 则

$$\sum_{i=1}^n |\lambda_i|^2 \leq \sum_{i,j=1}^n |a_{ij}|^2 = \|\mathbf{A}\|_F^2,$$

当且仅当 $\mathbf{A}^H \mathbf{A} = \mathbf{A} \mathbf{A}^H$ 时, 等号成立.

2.5.5.1.2 特征值的包含区域

2.5.5.1.2.1 盖尔圆

2.5.5.1.2.2 什么是盖尔圆

Definition 35 (盖尔圆 (一般教材定义)) 设 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{C}^{n \times n}$, 称不等式

$$|z - a_{ii}| \leq R_i, R_i = R_i(\mathbf{A}) = \sum_{j=1, j \neq i}^n |a_{ij}|$$

在复平面上所确定的区域为矩阵 \mathbf{A} 的第 i 个 盖尔圆 (*Gerschgorin*), 记为 G_i , R_i 为不包括 a_{ii} 的行和, 称为盖尔圆 G_i , ($i = 1, 2, \dots, n$) 的半径.

Definition 36 (盖尔圆 (我的定义)) 实际上, 在上述定义中, 若令半径为

$$|z - a_{ii}| \leq R_i, R_i = R_i(\mathbf{A}) = \min \left(\sum_{j=1, j \neq i}^n |a_{ij}|, \sum_{i=1, i \neq j}^n |a_{ij}| \right)$$

显然确定特征值范围会变得更为简单, 不知道为什么不这样定义.

注解: 半径 R_i 也可以按列求和得到, 这是因为 \mathbf{A} 与 \mathbf{A}^T 的特征值相同.

2.5.5.1.2.3 定理与结论

- 盖尔圆定理 1: 矩阵 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{C}^{n \times n}$ 的一切特征值都在它的 n 个盖尔圆的并集中.
- 盖尔圆定理 2: 矩阵 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{C}^{n \times n}$ 的联通部分由几个盖尔圆组成, 该部分就包含几个特征值 (盖尔圆相重时重复计数, 特征值相同时重复计数).

注解: 结论:

- 若矩阵 \mathbf{A} 的盖尔圆 G_i 关于实轴对称, 则特征值 λ_i 为实数.
 - 若矩阵 \mathbf{A} 的盖尔圆 G_i 各自孤立不连通, 则 \mathbf{A} 有 n 个互不相同的特征值.
-

2.5.5.1.2.4 特征值的隔离问题

设矩阵 $\mathbf{A} = (a_{ij})_{n \times n} \in \mathbb{C}^{n \times n}$, 构造对角矩阵 $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n)$, 其中 $d_i > 0, i = 1, 2, \dots, n$, 由于矩阵

$$\mathbf{B} = \mathbf{DAD}^{-1} = \left(\frac{d_i}{d_j} a_{ij} \right)_{n \times n}$$

相似于 \mathbf{A} , 所以矩阵 \mathbf{B}, \mathbf{A} 的特征值集合相同.

提示:

- 取 $d_i < 1, d_k = 1, k \neq i$, 可使第 i 个盖尔圆的半径减小
- 取 $d_i > 1, d_k = 1, k \neq i$, 可使第 i 个盖尔圆的半径增大

举个例子

应用盖尔圆定理隔离如下矩阵的特征值

$$\mathbf{A} = \begin{bmatrix} 20 & 3 & 1 \\ 2 & 10 & 2 \\ 8 & 1 & 0 \end{bmatrix}$$

解: 矩阵 \mathbf{A} 的三个盖尔圆为 $G_1 : |z - 20| \leq 4$, $G_2 : |z - 10| \leq 4$, $G_3 : |z - 0| \leq 9$, 如图所示

易知盖尔圆 G_2, G_3 相交, G_1 孤立, 构造矩阵 $D = \text{diag}(1, 1, 1/2)$ 使得 G_3 的半径减小, 则有

$$\mathbf{B} = \mathbf{DAD}^{-1} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1/2 \end{bmatrix} \begin{bmatrix} 20 & 3 & 1 \\ 2 & 10 & 2 \\ 8 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 2 \end{bmatrix} = \begin{bmatrix} 20 & 3 & 1 \\ 2 & 10 & 4 \\ 4 & 1/2 & 0 \end{bmatrix}$$

矩阵 \mathbf{B}^T 的三个盖尔圆为 $G_1 : |z - 20| \leq 4$, $G_2 : |z - 10| \leq 4$, $G_3 : |z - 0| \leq 9$, 如 fig-demo_Gerschgorin_exp1 所示, 它们相互孤立,

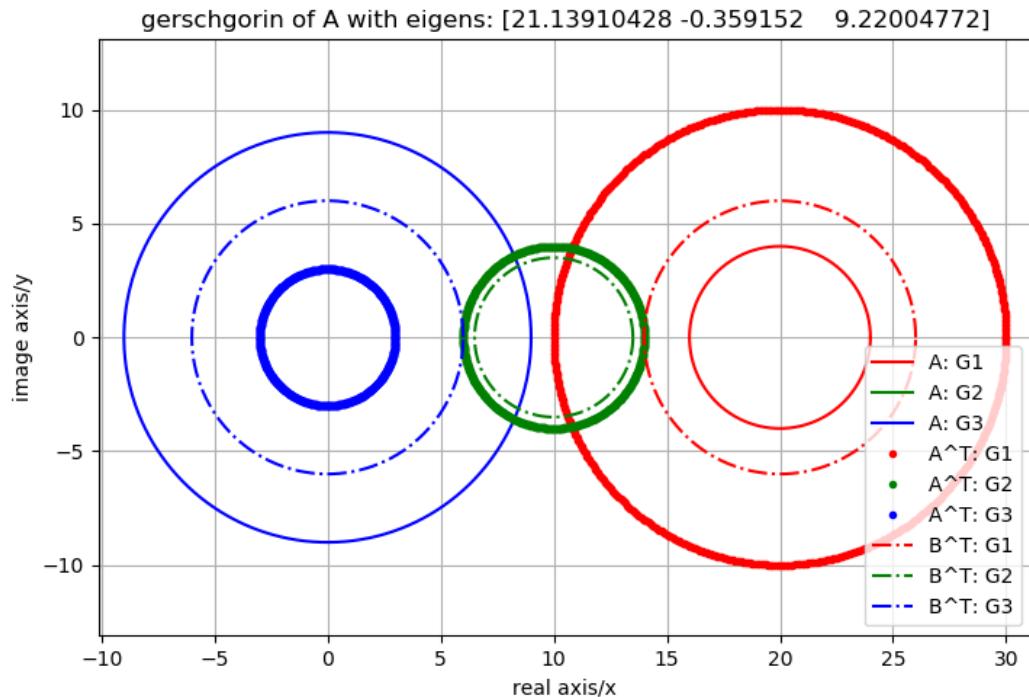


图 2.13: 矩阵 $\mathbf{A}, \mathbf{A}^T, \mathbf{B}^T$ 的盖尔圆
矩阵 $\mathbf{A}, \mathbf{A}^T, \mathbf{B}^T$ 的盖尔圆示意图.

实际上若采用盖尔圆定义 2, 可以很容易得到 fig-demo_Gerschgorin 所示结果

上述结果实验代码为

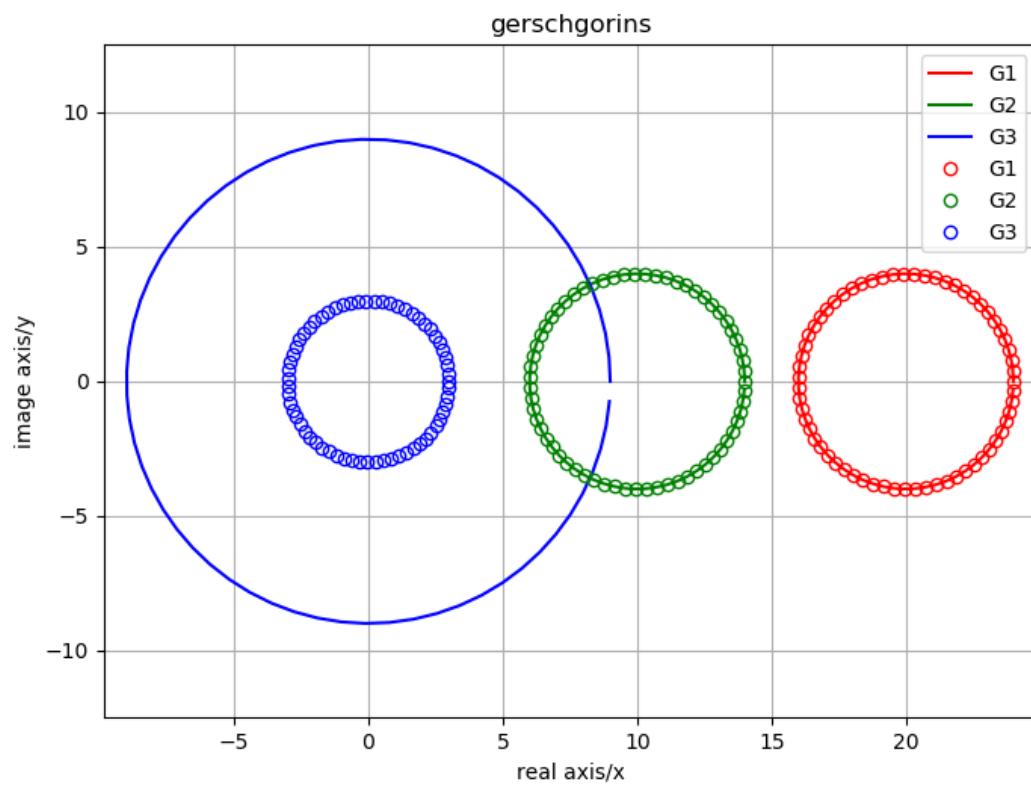


图 2.14: 矩阵 A 的盖尔圆

矩阵 A 的盖尔圆示意图, 每个盖尔圆的半径取为元素所在行元素和, 与元素所在列元素和中最小的那个.

代码 2.4: demo_Gerschgorin.py

```

1 import numpy as np
2 import pytool
3
4 A = [[20, 3, 1], [2, 10, 2], [8, 1, 0]]
5
6 A = np.array(A)
7 D = np.diag((1, 1, 1 / 2))
8 B = np.dot(D, A)
9 B = np.dot(B, np.linalg.inv(D))
10
11 Cis = []
12 Ris = []
13 Cis1, Ris1 = pytool.gerschgorin(A)
14 Cis2, Ris2 = pytool.gerschgorin(A.transpose())
15 Cis3, Ris3 = pytool.gerschgorin(B.transpose())
16
17 Cis = Cis + Cis1
18 Ris = Ris + Ris1
19
20 Cis = Cis + Cis2
21 Ris = Ris + Ris2
22
23 Cis = Cis + Cis3
24 Ris = Ris + Ris3
25
26 colorlines = ['-r', '-g', '-b', '.r', '.g', '.b', '-.r', '-.g', '-.b']
27
28 evals, eigvectors = np.linalg.eig(A)
29
30 Title = "gerschgorin of A with eigens: " + str(evals)
31 Legend = ['A: G1', 'A: G2', 'A: G3', 'A^T: G1',
32           'A^T: G2', 'A^T: G3', 'B^T: G1', 'B^T: G2', 'B^T: G3']
33
34 pytool.plot_circles(
35     Cis=Cis, Ris=Ris, colorlines=colorlines, dTheta=0.01, title=Title, ↴
36     legend=Legend)

```

2.5.5.1.2.5 应用

- 判定矩阵是否可逆: 矩阵的行列式的值等于矩阵特征值的乘积, 由于行列式不为零的矩阵可逆, 所以可逆矩阵的特征值不等于零, 借助盖尔圆判定特征值的取值

2.5.5.2 特征值的极性

2.5.5.2.1 概念与内涵

2.5.6 广义逆矩阵

2.5.6.1 投影矩阵

2.5.6.1.1 投影算子与投影矩阵

2.5.6.1.1.1 什么是投影

Definition 37 (投影) 设 \mathbb{L} 和 \mathbb{M} 均为 \mathbb{C}^n 的子空间, 且 $\mathbb{L} \oplus \mathbb{M} = \mathbb{C}^n$, $\forall \mathbf{x} \in \mathbb{C}^n$ 可分解为

$$\mathbf{x} = \mathbf{y} + \mathbf{z}, \mathbf{y} \in \mathbb{L}, \mathbf{z} \in \mathbb{M}$$

称 \mathbf{y} 是 \mathbf{x} 沿着 \mathbb{M} 到 \mathbb{L} 的 **投影**.

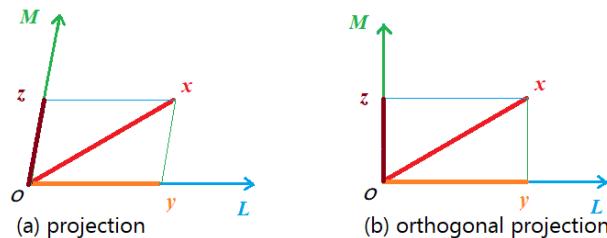


图 2.15: 二维空间中的投影示意
二维空间中的投影示意

2.5.6.1.1.2 投影算子与投影矩阵

Definition 38 (投影算子与投影矩阵) 将 $\forall \mathbf{x} \in \mathbb{C}^n$ 沿着 \mathbb{M} 到 \mathbb{L} 的投影变换称为沿着 \mathbb{M} 到 \mathbb{L} 的 **投影算子**, 记为 $P_{\mathbb{L}, \mathbb{M}}$, 即

$$P_{\mathbb{L}, \mathbb{M}} \mathbf{x} = \mathbf{y}.$$

投影算子 $P_{\mathbb{L}, \mathbb{M}}$ 在 \mathbb{C}^n 中的基 e_1, e_2, \dots, e_n 下的矩阵称为 **投影矩阵**.

2.5.6.1.1.3 线性投影算子

Definition 39 (线性投影算子) 若对于 $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{C}^n$ 和 $\lambda, \mu \in \mathbb{C}$, 恒有

$$P_{\mathbb{L}, \mathbb{M}}(\lambda \mathbf{x}_1 + \mu \mathbf{x}_2) = \lambda P_{\mathbb{L}, \mathbb{M}} \mathbf{x}_1 + \mu P_{\mathbb{L}, \mathbb{M}} \mathbf{x}_2$$

则称 $P_{\mathbb{L}, \mathbb{M}}$ 为 **线性算子**.

2.5.6.1.1.4 正交投影算子与正交投影矩阵

Definition 40 (正交投影算子与正交投影矩阵) 设 L 是 \mathbb{C}^n 的子空间, 称沿着 \mathbb{L}^\perp 到 \mathbb{L} 的投影算子 $P_{\mathbb{L}, \mathbb{L}^\perp}$ 为 **正交投影算子**, 简记为 $P_{\mathbb{L}}$.

正交投影算子 $P_{\mathbb{L}, \mathbb{L}^\perp}$ 在 \mathbb{C}^n 中的基 $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n$ 下的矩阵称为 **正交投影矩阵**.

2.5.6.1.1.5 性质

- 矩阵 \mathbf{P} 为投影矩阵 $\leftrightarrow \mathbf{P}$ 为幂等矩阵.
- 矩阵 \mathbf{P} 为正交投影矩阵 $\leftrightarrow \mathbf{P}$ 为幂等 Hermite 矩阵.

提示: 若 $\mathbf{P}^2 = \mathbf{P}\mathbf{P} = \mathbf{P}$, 则称 \mathbf{P} 为 **幂等矩阵**.

2.5.6.1.1.6 正交投影阵的求解

2.5.6.2 广义逆的定义与性质

2.5.6.2.1 Moore-Penrose 逆

2.5.6.2.1.1 概念

设有矩阵 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 矩阵 $\mathbf{X} \in \mathbb{C}^{n \times m}$ 和以下 Penrose 方程:

- (1) $\mathbf{AXA} = \mathbf{A}$
- (2) $\mathbf{XAX} = \mathbf{X}$
- (3) $(\mathbf{AX})^H = \mathbf{AX}$
- (4) $(\mathbf{XA})^H = \mathbf{XA}$

若矩阵 \mathbf{X} 满足 Penrose 方程组中的 $(i), (j), \dots, (k)$ 方程, 则称 \mathbf{X} 为 \mathbf{A} 的 $\{i, j, \dots, k\}$ -逆, 记为 $\mathbf{A}^{(i, j, \dots, k)}$, 全体记为 $\mathbf{A}\{i, j, \dots, k\}$.

若矩阵 \mathbf{X} 满足 Penrose 方程组中的 (1), (2), (3), (4) 方程, 则称 \mathbf{X} 为 \mathbf{A} 的 Moore-Penrose 逆, 记为 \mathbf{A}^+ .

易知:

- $\mathbf{A}^+ = \mathbf{A}^{(1, 2, 3, 4)}$

- 满足 Penrose 方程的广义逆共有 $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$ 类

2.5.6.2.1.2 结论与性质

1. 一般地, $\mathbf{A}\mathbf{A}^+ \neq \mathbf{A}^+\mathbf{A} \neq \mathbf{I}$
2. 若矩阵 \mathbf{A} 非奇异, 则 $\mathbf{A}^+ = \mathbf{A}^{-1}$
3. 若矩阵 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 则 \mathbf{A}^+ 存在且唯一
4. 若矩阵 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 则 $\mathbf{Y} = (\mathbf{A}^H \mathbf{A})^{(1)} \mathbf{A}^H \in \mathbf{A}\{1, 2, 3\}$
5. 若矩阵 $\mathbf{A} \in \mathbb{C}^{m \times n}$, 则 $\mathbf{Y} = \mathbf{A}^H (\mathbf{A}\mathbf{A}^H)^{(1)} \in \mathbf{A}\{1, 2, 4\}$
6. 若矩阵 $\mathbf{A} \in \mathbb{C}_n^{m \times n}$, 则 $\mathbf{A}^+ = (\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H$
7. 若矩阵 $\mathbf{A} \in \mathbb{C}_m^{m \times n}$, 则 $\mathbf{A}^+ = \mathbf{A}^H (\mathbf{A}\mathbf{A}^H)^{-1}$
8. 若矩阵 $\mathbf{A} \in \mathbb{C}^{m \times n}$, $\mathbf{B} \in \mathbb{C}^{n \times l}$, 不一定有 $(\mathbf{AB})^+ = \mathbf{B}^+ \mathbf{A}^+$

提示: 如设 $\mathbf{A} = (1, 0)$, $\mathbf{B} = (1, 1)^T$, 则 $\mathbf{AB} = [1]$, $(\mathbf{AB})^+ = [1]$, $\mathbf{A}^+ = (1, 0)^T$, $\mathbf{B}^+ = (1/2, 1/2)$, $\mathbf{A}^+ \mathbf{B}^+ = [1/2]$

2.5.6.3 广义逆计算

2.5.6.3.1 Hermite 标准形计算 {1}-逆和 {1,2}-逆

2.5.6.3.2 满秩分解法

设 $\mathbf{A} \in \mathbb{C}_r^{m \times n}$ ($r > 0$), 若满秩分解为 $\mathbf{A}_r^{m \times n} = \mathbf{F}_r^{m \times r} \mathbf{G}_r^{r \times n}$, 则有

1. $\mathbf{G}^{(1)} \mathbf{F}^{(i)} \in \mathbf{A}\{i\}$ ($i = 1, 2, 3$)
2. $\mathbf{G}^{(i)} \mathbf{F}^{(1)} \in \mathbf{A}\{i\}$ ($i = 1, 2, 4$)
3. $\mathbf{G}^{(1)} \mathbf{F}^+ \in \mathbf{A}\{1, 2, 3\}$
4. $\mathbf{G}^+ \mathbf{F}^{(1)} \in \mathbf{A}\{1, 2, 4\}$
5. $\mathbf{A}^+ = \mathbf{G}^+ \mathbf{F}^{(1,3)} = \mathbf{G}^{(1,4)} \mathbf{F}^+$
6. $\mathbf{A}^+ = \mathbf{G}^+ \mathbf{F}^+ = \mathbf{G}^H (\mathbf{G}\mathbf{G}^H)^{-1} (\mathbf{F}^H \mathbf{F})^{-1} \mathbf{F}^H = \mathbf{G}^H (\mathbf{F}^H \mathbf{F} \mathbf{G}\mathbf{G}^H)^{-1} \mathbf{F}^H = \mathbf{G}^H (\mathbf{F}^H \mathbf{A}\mathbf{G}^H)^{-1} \mathbf{F}^H$

2.5.6.4 广义逆矩阵与线性方程组

2.5.6.4.1 基本概念

考虑非齐次 (*Non-Homogeneous*) 线性方程组 (*System of linear equations*)

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

其中, $\mathbf{A} \in \mathbb{C}^{m \times n}$, $\mathbf{b} \in \mathbb{C}^m$, $\mathbf{x} \in \mathbb{C}^n$ 为待求解向量. 若存在 \mathbf{x} 满足上述方程, 则称方程组 **相容**或**有解**, 否则称方程组 **不相容**或**无解**.

提示:

- 线性(可以表示成 $\mathbf{A}\mathbf{x} = \mathbf{b}$), 非线性(不能表示成 $\mathbf{A}\mathbf{x} = \mathbf{b}$):
- 齐次($\mathbf{b} = 0$), 非齐次($\mathbf{b} \neq 0$):
- 相容(有解, 不矛盾), 不相容(无解, 矛盾):

2.5.6.4.2 几种常见解

- 线性方程组相容的充要条件: $\mathbf{A}\mathbf{A}^{(1)}\mathbf{b} = \mathbf{b}$, 不唯一
- 不相容线性方程组的最小二乘解: $\mathbf{x} = \mathbf{A}^{(1,3)}\mathbf{b}$, 不唯一
- 相容线性方程组的极小范数解: $\mathbf{x} = \mathbf{A}^{(1,4)}\mathbf{b}$, 唯一, 且在 $R(\mathbf{A}^H)$ 中
- 不相容线性方程组的极小范数最小二乘解: $\mathbf{x} = \mathbf{A}^+\mathbf{b}$, 唯一

对应地还有:

- 相容线性方程组的通解: $\mathbf{x} = \mathbf{A}^{(1)}\mathbf{b} + (\mathbf{I} - \mathbf{A}^{(1)}\mathbf{A}\mathbf{y})$, $\mathbf{y} \in \mathbb{C}^n$
- 设 $\mathbf{X} \in \mathbb{C}^{n \times m}$, 若 $\forall \mathbf{b} \in \mathbb{C}^m$, 有 $\mathbf{x} = \mathbf{X}\mathbf{b}$ 是 $\mathbf{A}\mathbf{x} = \mathbf{b}$ 的最小二乘解, 则 $\mathbf{X} \in \mathbf{A}\{1, 3\}$
- 设 $\mathbf{X} \in \mathbb{C}^{n \times m}$, 若 $\forall \mathbf{b} \in \mathbb{C}^m$, 有 $\mathbf{x} = \mathbf{X}\mathbf{b}$ 是 $\mathbf{A}\mathbf{x} = \mathbf{b}$ 的极小范数解, 则 $\mathbf{X} \in \mathbf{A}\{1, 4\}$
- 设 $\mathbf{X} \in \mathbb{C}^{n \times m}$, 若 $\forall \mathbf{b} \in \mathbb{C}^m$, 有 $\mathbf{x} = \mathbf{X}\mathbf{b}$ 是 $\mathbf{A}\mathbf{x} = \mathbf{b}$ 的极小范数最小二乘解, 则 $\mathbf{X} = \mathbf{A}^+$

2.5.7 杂项

2.5.7.1 常用总结

2.5.7.1.1 顺序主子式

设有 n 阶方阵 \mathbf{A}

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

则其 i , ($0 \leq i \leq n$) 阶顺序主子式为行列式

$$D_i = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1i} \\ a_{21} & a_{22} & \cdots & a_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} \end{vmatrix}$$

矩阵 \mathbf{A} 所有阶顺序主子式 D_1, D_2, \dots, D_n 组成 \mathbf{A} 的顺序主子式.

- 矩阵 \mathbf{A} 为正定矩阵的充要条件是 \mathbf{A} 的所有顺序主子式 D_i 大于零;
- 矩阵 \mathbf{A} 有唯一 LU 分解的充要条件是 \mathbf{A} 的所有顺序主子式 D_i 不等于零;

2.5.7.1.2 伴随矩阵

定义: 对于方阵 $\mathbf{A} = (a_{ij})_{n \times n}$, 将矩阵 \mathbf{A} 的第 i 行第 j 列去掉后, 剩下的元素按原来的顺序组成一个新的 $n-1$ 阶矩阵, 新矩阵的行列式称为元素 a_{ij} 的 **余子式**, 记为 M_{ij} ; 称 $A_{ij} = (-1)^{i+j} M_{ij}$ 为 a_{ij} 的代数余子式; 将所有元素的代数余子式按如下规律组成一个矩阵, 将此矩阵称为方阵 \mathbf{A} 的 **伴随矩阵**, 记为: 方阵 \mathbf{A}^*

$$\mathbf{A}^* = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}$$

性质:

- \mathbf{A} 可逆 $\leftrightarrow \mathbf{A}^*$ 可逆
- \mathbf{A} 可逆 $\leftrightarrow (\mathbf{A}^{-1})^* = (\mathbf{A}^*)^{-1}$
- \mathbf{A} 可逆 $\leftrightarrow \mathbf{A}^* = |\mathbf{A}| \mathbf{A}^{-1}$
- $(\mathbf{A}^*)^* = (|\mathbf{A}| \mathbf{A}^{-1})^* = |\mathbf{A}|^{n-1} (\mathbf{A}^*)^{-1}$
- $(\mathbf{A}^T)^* = (\mathbf{A}^*)^T$
- $(k\mathbf{A})^* = k^{n-1} \mathbf{A}^*$
- $(\mathbf{AB})^* = \mathbf{B}^* \mathbf{A}^*$
- $|\mathbf{A}^*| = |\mathbf{A}|^{n-1}$

2.5.7.1.3 矩阵的逆

2.5.7.1.3.1 性质

设 \mathbf{A}, \mathbf{B} 为 n 阶可逆矩阵, 数 $\lambda \neq 0$, 则

- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
- $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$
- $(\lambda \mathbf{A})^{-1} = \lambda^{-1} \mathbf{A}^{-1}$
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$

2.5.7.1.3.2 特殊矩阵的逆

2.5.7.1.3.3 二阶方阵的逆

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

提示: 上述公式源于伴随矩阵求逆原理.

2.5.7.1.3.4 三角矩阵的逆

$$C = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 1 & \ddots & & \\ \vdots & \vdots & \ddots & 1 & \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}, C^{-1} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & \ddots & & \\ & \ddots & 1 & & \\ & & -1 & 1 & \end{bmatrix}$$

2.5.7.1.3.5 求解方法

以以下矩阵为例:

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 0 & 1 \end{bmatrix}$$

2.5.7.1.3.6 伴随矩阵求逆

由 $A^* = |A|A^{-1}$ 知 $A^{-1} = \frac{1}{|A|}A^*$, 故可根据此式计算.

例如:

$$\begin{aligned} A_{11} &= (-1)^{1+1} \begin{vmatrix} 1 & 2 \\ 0 & 1 \end{vmatrix} = 1, & A_{12} &= (-1)^{1+2} \begin{vmatrix} 2 & 2 \\ 2 & 1 \end{vmatrix} = 2, & A_{13} &= (-1)^{1+3} \begin{vmatrix} 2 & 1 \\ 2 & 0 \end{vmatrix} = -2 \\ A_{21} &= (-1)^{2+1} \begin{vmatrix} 2 & 2 \\ 0 & 1 \end{vmatrix} = -2, & A_{22} &= (-1)^{2+2} \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = -3, & A_{23} &= (-1)^{2+3} \begin{vmatrix} 1 & 2 \\ 2 & 0 \end{vmatrix} = 4 \\ A_{31} &= (-1)^{3+1} \begin{vmatrix} 2 & 2 \\ 1 & 2 \end{vmatrix} = 2, & A_{32} &= (-1)^{3+2} \begin{vmatrix} 1 & 2 \\ 2 & 2 \end{vmatrix} = 2, & A_{33} &= (-1)^{3+3} \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = -3 \end{aligned}$$

从而有

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & -2 & 2 \\ 2 & -3 & 2 \\ -2 & 4 & -3 \end{bmatrix}$$

2.5.7.1.3.7 初等行变换求逆

将待求逆矩阵与单位矩阵拼成一个矩阵, 对新矩阵只进行 **初等行变换**, 使得待求逆矩阵部分变为单位矩阵, 那么对应的原始的单位阵变为待求逆矩阵的逆, 即

$$[\mathbf{A}|\mathbf{I}] \rightarrow [\mathbf{I}|\mathbf{A}^{-1}]$$

例如:

$$\begin{aligned} [\mathbf{A}|\mathbf{E}] &\rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 2 & 1 & 0 & 0 \\ 2 & 1 & 2 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \xrightarrow{r_2 \leftrightarrow r_3} \left[\begin{array}{ccc|ccc} 1 & 2 & 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & -1 \\ 2 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \\ &\xrightarrow{r_1 - 2r_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 1 & 0 & 1 & -1 \\ 2 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \xrightarrow{r_3 - 2r_1} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & 1 & -2 & 4 & -3 \end{array} \right] \\ &\xrightarrow{r_2 \leftrightarrow r_3} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 0 & 2 & -3 & 2 \\ 0 & 0 & 1 & -2 & 4 & -3 \end{array} \right] \Rightarrow \mathbf{A}^{-1} = \begin{bmatrix} 1 & -2 & 2 \\ 2 & -3 & 2 \\ -2 & 4 & -3 \end{bmatrix} \\ &\text{if } \xrightarrow{c_3 - c_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -2 & 4 \\ 0 & 1 & 0 & 0 & 1 & -2 \\ 0 & 0 & 1 & -2 & 4 & -7 \end{array} \right] \quad \begin{array}{l} \text{不能进行} \\ \text{初等列变换} \end{array} \end{aligned}$$

图 2.16: 初等变换求逆
初等变换求逆

2.5.7.1.3.8 Sherman-Morrison-Woodbury 公式

设 $\mathbf{A} \in \mathbb{R}^{n \times n}$ 非奇异, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, 则有 Sherman-Morrison 等式

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{u}(\mathbf{I} + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u})^{-1}\mathbf{v}^T\mathbf{A}^{-1}. \quad (2.3)$$

提示: Sherman-Morrison 等式可以通过求解线性方程组 $(\mathbf{A} + \mathbf{u}\mathbf{v}^T)\mathbf{x} = \mathbf{b}$ 得到.

由 Sherman-Morrison 公式.2.3, 令 $\mathbf{U} \in \mathbb{R}^{n \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$, 则有 Sherman-Morrison-Woodbury 等式

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}. \quad (2.4)$$

2.5.7.1.4 矩阵的秩

2.5.7.1.4.1 性质

- $0 \leq \text{rank}(\mathbf{A}_{m \times n}) \leq \min\{m, n\}$
- $\text{rank}(\mathbf{A}^T) = \text{rank}(\mathbf{A})$
- $\text{rank}(\mathbf{AB}) \leq \min\{\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})\}$
- $\text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$

- $\max\{\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})\} \leq \text{rank}(\mathbf{A}, \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$
- 若 $\mathbf{A} \sim \mathbf{B}$, 则 $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{B})$
- 若 \mathbf{P}, \mathbf{Q} 可逆, 则 $\text{rank}(\mathbf{PAQ}) = \text{rank}(\mathbf{A})$
- 若 $\mathbf{A}_{m \times n} \mathbf{B}_{n \times l} = \mathbf{O}$, 则 $\text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) \leq n$
- 若 \mathbf{A} 为列满秩矩阵, 且 $\mathbf{AB} = \mathbf{O}$, 则 $\mathbf{B} = \mathbf{O}$

2.5.7.1.5 矩阵的迹行列式特征值

- 方阵 \mathbf{A} 的行列式与特征值的关系: $|\mathbf{A}| = \lambda_1 \lambda_2 \cdots \lambda_n$
- 方阵 \mathbf{A} 的迹与特征值的关系: $\text{tr}(\mathbf{A}) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$

2.5.7.2 初等变换

初等变换 (*Elementary transformation*)

- 1) 某一行 (列) 乘以一个数;
- 2) 某一行 (列) 乘以一个数的结果加到另一行 (列);
- 3) 互换两行 (列).

2.5.7.2.1 初等行变换

2.5.7.2.2 初等列变换

2.5.7.2.3 特殊初等变换

$$\begin{aligned} \mathbf{P}^{-1} \mathbf{A} \mathbf{P} &= \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & \color{red}{a_{33}} & \color{green}{a_{34}} \\ a_{41} & a_{42} & \color{blue}{a_{43}} & \color{purple}{a_{44}} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{41} & a_{42} & \color{blue}{a_{43}} & \color{purple}{a_{44}} \\ a_{31} & a_{32} & \color{red}{a_{33}} & \color{green}{a_{34}} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{14} & a_{13} \\ a_{21} & a_{22} & a_{24} & a_{23} \\ a_{41} & a_{42} & \color{purple}{a_{44}} & \color{blue}{a_{43}} \\ a_{31} & a_{32} & \color{red}{a_{34}} & \color{green}{a_{33}} \end{bmatrix} \end{aligned}$$

图 2.17: 通过初等行变换和初等列变换转置
通过初等行变换和初等列变换转置

2.5.7.3 常见矩阵概念

2.5.7.3.1 奇异非奇异

奇异 (singular) 与非奇异 (non-singular) 和不可逆 (non-invertible) 与可逆 (invertible) 是相同的,

- 首先是指方阵 $A^{n \times n}$;
- 非奇异阵 \Leftrightarrow 可逆矩阵 \Leftrightarrow 满秩 \Leftrightarrow 行列式不等于零 $\|A\| \neq 0 \Leftrightarrow$ 正定阵;
- 奇异阵 \Leftrightarrow 非可逆矩阵 \Leftrightarrow 非满秩 \Leftrightarrow 行列式等于零 $\|A\| = 0 \Leftrightarrow$ 非正定阵;

2.5.7.3.2 特殊矩阵

2.5.7.3.2.1 概念汇总

1. 对称矩阵

- 实对称: $A^T = A$, $A \in \mathbb{R}^{n \times n}$
- 酉对称: $A^H = A$, $A \in \mathbb{C}^{n \times n}$, 也叫 Hermite 矩阵
- 复对称: $A^T = A$, $A \in \mathbb{C}^{n \times n}$

2. 正规矩阵 (*Normal Matrix*)

- 正规矩阵: $A^H A = A A^H$, $A \in \mathbb{C}^{n \times n}$
- 正交矩阵, 酉矩阵, 对角矩阵, 实对称矩阵及酉对称矩阵均为正规矩阵

3. 正交矩阵

- 实正交矩阵: $A^T A = A A^T = I$, $A \in \mathbb{R}^{n \times n}$
- 复正交矩阵: $A^H A = A A^H = I$, $A \in \mathbb{C}^{n \times n}$, 也叫酉矩阵

[Hermitian matrix](http://en.volupedia.org/wiki/Hermitian_matrix) (http://en.volupedia.org/wiki/Hermitian_matrix) 也叫自伴随矩阵 (self-adjoint matrix)

2.5.7.3.2.2 结论

设 $A \in \mathbb{C}_r^{m \times n}$, ($r > 0$), 则

- $A^H A$ 是 Hermite 矩阵, 即 $(A^H A)^H = A^H A$, 其特征值为非负实数;
- $\text{rank}(A^H A) = \text{rank}(A A^H) = \text{rank}(A)$;
- $A = \mathbf{0}$ 的充要条件是 $A^H A = \mathbf{0}$, 此时, $r = 0$

2.5.7.4 对角矩阵

$$\begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{nn} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} d_{11}a_{11} & d_{11}a_{12} & \cdots & d_{11}a_{1n} \\ d_{22}a_{21} & d_{22}a_{22} & \cdots & d_{22}a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{nn}a_{n1} & d_{nn}a_{n2} & \cdots & d_{nn}a_{nn} \end{bmatrix}$$

2.5.7.5 矩阵运算

2.5.7.5.1 乘法

2.5.7.5.1.1 逐元素积

对于维度相同的任意两个矩阵 $\mathbf{A} = (a_{ij})_{M \times N}$, $\mathbf{B} = (b_{ij})_{M \times N}$, 其逐元素积为

$$\mathbf{A} \odot \mathbf{B} = \mathbf{C} = (c_{ij})_{M \times N} = (a_{ij}b_{ij})_{M \times N} \quad (2.5)$$

2.5.7.5.1.2 点积/内积

定义矩阵 $\mathbf{A} = (a_{ij})_{M \times N}$ 与矩阵 $\mathbf{B} = (b_{ij})_{N \times P}$ 的点积为

$$\mathbf{AB} = \mathbf{C} = (c_{ij})_{M \times P} = \left(\sum_{n=1}^N a_{in}b_{nj} \right)_{M \times P} \quad (2.6)$$

设有矩阵 $\mathbf{A} = (a_{ij})_{M \times N}$, $\mathbf{B} = (b_{ij})_{N \times P}$, $\mathbf{C} = (c_{ij})_{P \times Q}$, 则他们的点积为

$$\mathbf{ABC} = \mathbf{D} = (d_{ij})_{M \times Q} = \left(\sum_{p=1}^P \left(\sum_{n=1}^N a_{in}b_{np} \right) c_{pj} \right)_{M \times Q} \quad (2.7)$$

2.5.7.5.1.3 Kronecker 积

对于任意维度的两个矩阵 $\mathbf{A} = (a_{ij})_{M \times N}$, $\mathbf{B} = (b_{ij})_{P \times Q}$, 其 Kronecker 积为用右矩阵乘以左矩阵的结果.

$$\mathbf{A} \times \mathbf{B} = \mathbf{C} = (c_{ij})_{MP \times NQ} = (a_{ij}\mathbf{B})_{MP \times NQ} \quad (2.8)$$

2.5.7.5.1.4 卷积

[卷积与转置卷积](https://blog.csdn.net/LoseInVain/article/details/81098502) (<https://blog.csdn.net/LoseInVain/article/details/81098502>)

[Deconvolution and Checkerboard Artifacts](https://distill.pub/2016/deconv-checkerboard/) (<https://distill.pub/2016/deconv-checkerboard/>)

2.5.7.5.1.5 示例

代码 2.5: demo_MatrixOperation.py

```
1 import numpy as np
2
3 a = np.array([[1, 2, 3], [4, 5, 6]])
4 b = np.array([[10, 20, 30], [40, 50, 60]])
5
6 print("---element-wise: ")
7 c = a * b
8 print(c)
9
10 print("---matmul: ")
11 c = np.matmul(a, b.transpose())
12 print(c)
13
14 c = np.dot(a, b.transpose())
15
16 print("---dot: ")
17 print(c)
18
19 c = np.kron(a, b)
20
21 print("---kron: ")
22 print(c)
```

结果如下:

```
---element-wise:
[[ 10  40  90]
 [160 250 360]]
---matmul:
[[140 320]
 [320 770]]
---dot:
[[140 320]
 [320 770]]
---kron:
[[ 10   20   30   20   40   60   30   60   90]
 [ 40   50   60   80  100  120  120  150  180]
 [ 40   80  120   50  100  150   60  120  180]
 [160  200  240  200  250  300  240  300  360]]
```

2.5.8 名词术语

Characteristic Polynomial 特征多项式 ([Characteristic polynomial](http://en.volupedia.org/wiki/Characteristic_polynomial) (http://en.volupedia.org/wiki/Characteristic_polynomial))
在线性代数中,

Characteristic Subspace 特征子空间 ([Characteristic subspace](http://en.volupedia.org/wiki/Minimal_polynomial) (http://en.volupedia.org/wiki/Minimal_polynomial))
在线性代数中,

Cholesky decomposition Cholesky 分解 ([Cholesky decomposition](http://en.volupedia.org/wiki/Cholesky_decomposition) (http://en.volupedia.org/wiki/Cholesky_decomposition))
三角分解的一种, 把一个对称正定的矩阵表示成一个下三角矩阵 L 和其转置 L^T 的乘积的分解. 它要求矩阵的所有特征值必须大于零, 故分解的下三角的对角元也是大于零的. Cholesky 分解法又称平方根法, 是当 A 为实对称正定矩阵时, LU 三角分解法的变形.

Conjugate transpose 共轭转置 ([Conjugate transpose](http://en.volupedia.org/wiki/Conjugate_transpose) (http://en.volupedia.org/wiki/Conjugate_transpose)), 也叫 Hermitian 转置 (Hermitian transpose), 是对矩阵中的元素取共轭复数后再转置的操作.

Diagonal Matrix 对角矩阵 ([Diagonal Matrix](http://en.volupedia.org/wiki/Diagonal_matrix) (http://en.volupedia.org/wiki/Diagonal_matrix)) 主对角线以外的所有元素都为零的矩阵称为对角矩阵.

Direct Sum 直和 ([Direct Sum](http://en.volupedia.org/wiki/Direct_sum) (http://en.volupedia.org/wiki/Direct_sum))

Doolittle decomposition Doolittle 分解 ([Doolittle Decomposition](http://en.volupedia.org/wiki/LU_decomposition) (http://en.volupedia.org/wiki/LU_decomposition))
三角分解的一种, 把一个对称正定的矩阵表示成一个下三角矩阵 L 和其转置 L^T 的乘积的分解. 它要求矩阵的所有特征值必须大于零, 故分解的下三角的对角元也是大于零的. Cholesky 分解法又称平方根法, 是当 A 为实对称正定矩阵时, LU 三角分解法的变形.

Elementary Transformation 初等变换 (Elementary transformation) 线性代数的基本概念之一, 包含初等行变换与初等列变换. 以矩阵的初等变换为例, 具体包含三种操作: 1) 某一行 (列) 乘以一个数; 2) 某一行 (列) 乘以一个数的结果加到另一行 (列); 3) 互换两行 (列).

Full Rank Decomposition 满秩分解 (Full Rank Decomposition) 是指将一矩阵分解为行满秩 F 与列满秩 G 的两个矩阵的乘积的分解.

Generalized eigenvalue problem 广义特征值问题 (Generalized eigenvalue problem (http://en.volupedia.org/wiki/Eigendecomposition_of_a_matrix#Generalized_eigenvalue_problem))

Givens transformation Givens 变换 ([Givens transformation](http://en.volupedia.org/wiki/Givens_transformation) (http://en.volupedia.org/wiki/Givens_transformation)) 在线性代数中, Givens 变换, 也称 Givens 旋转 (Givens rotation), 是描述将某一平面内的向量进行旋转的线性变换. 由 Wallace Givens 于 1950 年提出.

Gram–Schmidt process Gram–Schmidt 过程 ([Gram–Schmidt process](http://en.volupedia.org/wiki/Gram%E2%80%93Schmidt_process) (http://en.volupedia.org/wiki/Gram%E2%80%93Schmidt_process))
是指实内积空间的等度量线性变换, 保持内积不变, 对应于复内积空间的酉变换.

Hermitian matrix Hermitian 矩阵 ([Hermitian matrix](http://en.volupedia.org/wiki/Hermitian_matrix) (http://en.volupedia.org/wiki/Hermitian_matrix))

Householder transformation Householder 变换 (Householder transformation ([Householder transformation](http://en.volupedia.org/wiki/Householder_transformation) (http://en.volupedia.org/wiki/Householder_transformation))) 在线性代数中, Householder 变换, 也称为 Householder 反射 (Householder reflection), 是描述包含原点的平面或超平面的反射的线性变换. 由 Alston Scott Householder 于 1958 年提出.

Inner Product Space 内积空间 ([Inner product space](http://en.volupedia.org/wiki/Inner_product_space) (http://en.volupedia.org/wiki/Inner_product_space)) 在线性代数中, 内积空间是具有称为内积的附加结构的向量空间. 这个附加结构将空间中的每对向量与称为向量内积的标量量相关联. 内积允许严格引入直观的几何概念, 如向量的长度或两个向量之间

的角度。它们还提供了定义向量之间的正交性(零内积)的方法。内积空间将欧氏空间(其中内积是点积,也称为标量积)推广到任意(可能是无限)维的向量空间,并在泛函分析中加以研究。带内积的向量空间概念的首次使用是由于 Peano 在 1898 年提出的。

Invariant Subspace 不变子空间 ([Invariant subspace](http://en.volupedia.org/wiki/Invariant_subspace) (http://en.volupedia.org/wiki/Invariant_subspace)) 在线性代数中,

Kernel Space 核空间 ([Kernel Space](http://en.volupedia.org/wiki/Kernel_(linear_algebra)) ([http://en.volupedia.org/wiki/Kernel_\(linear_algebra\)](http://en.volupedia.org/wiki/Kernel_(linear_algebra))))

Linear Combination 线性组合 (Linear Combination) 线性代数的基本概念之一,它表示一些项与其相应系数相乘后的累加和,如 $z = ax + by$ 。

Linear Representation 线性表示 (Linear Representation)

Linear Space 线性空间 ([Linear Space](http://en.volupedia.org/wiki/Vector_space) (http://en.volupedia.org/wiki/Vector_space))

Linear Span 线性生成空间 ([Linear Span](http://en.volupedia.org/wiki/Linear_span) (http://en.volupedia.org/wiki/Linear_span))

Linear Subspace 线性子空间 ([Linear Subspace](http://en.volupedia.org/wiki/Linear_subspace) (http://en.volupedia.org/wiki/Linear_subspace))

Linearly Dependent 线性相关 (Linear Correlation, Linearly Dependent) 线性代数的基本概念之一,

Linearly Independent 线性无关 (Linear Independence, Linearly Independent) 线性代数的基本概念之一,也称线性独立

Lower Triangular Matrix 下三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上方的所有元素都为零的矩阵。

Minimal Polynomial 最小多项式 ([Minimal polynomial](http://en.volupedia.org/wiki/Minimal_polynomial) (http://en.volupedia.org/wiki/Minimal_polynomial)) 在线性代数中,

Noise Subspace 噪声子空间

Non-Homogeneous 非齐次 (Non-Homogeneous),常数项不为零。

Normal Matrix 正规矩阵 ([Normal Matrix](http://en.volupedia.org/wiki/Normal_matrix) (http://en.volupedia.org/wiki/Normal_matrix))

Null Space 零空间 ([Null Space](http://en.volupedia.org/wiki/Kernel_(linear_algebra)) ([http://en.volupedia.org/wiki/Kernel_\(linear_algebra\)](http://en.volupedia.org/wiki/Kernel_(linear_algebra))))

Orthogonal Matrix 正交矩阵 ([Orthogonal Matrix](http://en.volupedia.org/wiki/Orthogonal_matrix) (http://en.volupedia.org/wiki/Orthogonal_matrix))

Orthogonal transformation 正交变换 ([Orthogonal transformation](http://en.volupedia.org/wiki/Orthogonal_transformation) (http://en.volupedia.org/wiki/Orthogonal_transformation)) 是指实内积空间的等度量线性变换,保持内积不变,对应于复内积空间的酉变换。

Orthogonal Triangular Decomposition 正交三角分解 ([Orthogonal Triangular Decomposition](http://en.volupedia.org/wiki/QR_decomposition) (http://en.volupedia.org/wiki/QR_decomposition)) 分解是把一个矩阵分解为一个正交矩阵 Q 与一个上三角矩阵 R 乘积的分解。

Permutation matrix 置换矩阵 ([Permutation matrix](http://en.volupedia.org/wiki/Permutation_matrix) (http://en.volupedia.org/wiki/Permutation_matrix)) 指每行或每列只有 1 个元素为 1,其余均为 0 的二值矩阵,这种矩阵具有置换矩阵两行或两列的功能。

Polynomial Matrix 多项式矩阵 ([Polynomial Matrix](http://en.volupedia.org/wiki/Polynomial_matrix) (http://en.volupedia.org/wiki/Polynomial_matrix))

Positive-definite matrix 正定矩阵 ([Positive-definite matrix](http://en.volupedia.org/wiki/Positive-definite_matrix) (http://en.volupedia.org/wiki/Positive-definite_matrix))

Series 级数 ([Series \(mathematics\)](http://en.volupedia.org/wiki/Series_(mathematics)) ([http://en.volupedia.org/wiki/Series_\(mathematics\)](http://en.volupedia.org/wiki/Series_(mathematics)))),

Signal Subspace 信号子空间 ([Signal subspace](http://en.volupedia.org/wiki/Signal_subspace) (http://en.volupedia.org/wiki/Signal_subspace)) In signal processing, signal subspace methods are empirical linear methods for dimensionality reduction and noise reduction. These approaches have attracted significant interest and investigation recently in the context of speech enhancement, speech modeling, and speech classification research. The signal subspace is also used in radio direction finding using the MUSIC (algorithm).

Singular Value 奇异值 ([Singular value](http://en.volupedia.org/wiki/Singular_value) (http://en.volupedia.org/wiki/Singular_value)) .

Singular Value Decomposition 奇 异 值 分 解 ([Singular value decomposition](http://en.volupedia.org/wiki/Singular_value_decomposition) (http://en.volupedia.org/wiki/Singular_value_decomposition)) 分解是把一个矩阵分解为一个酉矩阵 (unitary matrix) U , 矩形对角矩阵 Σ 与一个酉矩阵 V 乘积的分解.

Skew-Hermitian matrix 反 Hermitian 矩阵 ([Skew-Hermitian matrix](http://en.volupedia.org/wiki/Skew-Hermitian_matrix) (http://en.volupedia.org/wiki/Skew-Hermitian_matrix))

Skew-symmetric matrix 反对称矩阵 ([Skew-symmetric matrix](http://en.volupedia.org/wiki/Skew-symmetric_matrix) (http://en.volupedia.org/wiki/Skew-symmetric_matrix))

Spectral decomposition 谱分解 ([Spectral decomposition](http://en.volupedia.org/wiki/Spectral_decomposition) (http://en.volupedia.org/wiki/Spectral_decomposition)) 分解是把一个矩阵分解为一个酉矩阵 (unitary matrix) U , 矩形对角矩阵 Σ 与一个酉矩阵 V 乘积的分解.

Spectral Radius 谱半径 ([Spectral Radius](http://en.volupedia.org/wiki/Spectral_radius) (http://en.volupedia.org/wiki/Spectral_radius))),

Subordinate Norm 从属范数 ([Subordinate norm](http://en.volupedia.org/wiki/Matrix_norm#Induced_norm) (http://en.volupedia.org/wiki/Matrix_norm#Induced_norm))),

Symmetric matrix 对称矩阵 ([Symmetric matrix](http://en.volupedia.org/wiki/Symmetric_matrix) (http://en.volupedia.org/wiki/Symmetric_matrix))

Symmetry transformation 对称变换 ([Symmetry transformation](http://en.volupedia.org/wiki/Symmetry_transformation) (http://en.volupedia.org/wiki/Symmetry_transformation)) 是指实内积空间的一种具有对称性的线性变换, 对应于复内积空间的酉对称变换.

System of linear equations 线性方程组

Triangular Decomposition 三角分解 ([Triangular Decomposition](http://en.volupedia.org/wiki/Triangular_decomposition) (http://en.volupedia.org/wiki/Triangular_decomposition)) 分解是把一个矩阵分解为一个下三角矩阵 L 与一个上三角矩阵 R 乘积的分解, 也可分解为一个下三角矩阵 L 与一个对角阵 D 及一个上三角矩阵 R 乘积.

Triangular Matrix 三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 在线性代数中, 三角矩阵是一种特殊的方阵. 如果主对角线上方的所有元素都为零, 则称矩阵为下三角矩阵. 类似地, 如果主对角线下方的所有元素都为零, 则方形矩阵称为上三角矩阵.

Unit Lower Triangular Matrix 单位下三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上所有元素均为 1 的下三角矩阵.

Unit Triangular Matrix 单位三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 在线性代数中, 单位三角矩阵是指主对角线上的元素均为 1 的三角矩阵.

Unit Upper Triangular Matrix 单位上三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上所有元素均为 1 的上三角矩阵.

Unitary Symmetry transformation 酉 对 称 变 换 ([Unitary Symmetry transformation](http://en.volupedia.org/wiki/Unitary_transformation) (http://en.volupedia.org/wiki/Unitary_transformation)) 是指酉空间的一种具有对称性的线性变换, 对应于实内积空间的对称变换.

Unitary transformation 酉变换 ([Unitary transformation](http://en.volupedia.org/wiki/Unitary_transformation) (http://en.volupedia.org/wiki/Unitary_transformation)) 是指复内积空间的等度量线性变换, 保持内积不变, 对应于实内积空间的正交变换.

Upper Triangular Matrix 上三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线下方的所有元素都为零的矩阵.

Weighted Norm 加权范数 ([Weighted Norm](http://en.volupedia.org/wiki/Weighted_norm) (http://en.volupedia.org/wiki/Weighted_norm)) ,

Weighted Subspace 加权子空间 ([Weighted Subspace](http://en.volupedia.org/wiki/Weighted_subspace) (http://en.volupedia.org/wiki/Weighted_subspace)) , 定义在加权范数下的子空间.

2.6 概率与统计

2.6.1 相关分析

2.6.1.1 相关的概念

- 互相关
 - 互相关矩阵 --> 随机向量
 - 互相关函数 --> 随机过程
- 自相关
 - 自相关矩阵 --> 随机向量
 - 自相关函数 --> 随机过程
- 自协方差
- 互协方差

符号说明:

- 随机向量 \mathbf{x}
- 随机变量 x
- 随机变量的取值 x

提示: 数学中确定性信号的相关概念参见相关算子 (页 121) 小节.

2.6.1.1.1 互相关

在概率论与统计学中, 互相关用于描述两个随机向量 \mathbf{x}, \mathbf{y} 的元素间的相关性.

2.6.1.1.1.1 互相关函数

设两个随机过程 (x_t, y_t) , t 为时间, 可离散或连续, 两个随机过程在时刻 t 的均值和方差分别为 μ_{x_t}, μ_{y_t} , $\sigma_{x_t}^2, \sigma_{y_t}^2$, 互相关函数定义为

$$\mathbf{R}_{xy}(t_1, t_2) = E[x_{t_1}, \bar{y}_{t_2}]$$

2.6.1.1.1.2 互相关矩阵

设两个随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$ 的期望与方差存在, 则它们的 **互相关矩阵** (*Cross Correlation Matrix*) 定义为

$$\mathbf{R}_{xy} = E[\mathbf{xy}^T] = \begin{bmatrix} E[x_1 y_1] & E[x_1 y_2] & \cdots & E[x_1 y_n] \\ E[x_2 y_1] & E[x_2 y_2] & \cdots & E[x_2 y_n] \\ \vdots & \vdots & \vdots & \vdots \\ E[x_m y_1] & E[x_m y_2] & \cdots & E[x_m y_n] \end{bmatrix}$$

提示: 当随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$ 为复数随机向量时, 互相关矩阵定义为 $\mathbf{R}_{xy} = E[\mathbf{xy}^H]$.

2.6.1.1.1.3 性质

- 实随机向量 \mathbf{x}, \mathbf{y} 不相关 $\Leftrightarrow E[\mathbf{xy}^T] = E[\mathbf{x}]E[\mathbf{y}]^T$
- 复随机向量 \mathbf{x}, \mathbf{y} 不相关 $\Leftrightarrow E[\mathbf{xy}^H] = E[\mathbf{x}]E[\mathbf{y}]^H$

2.6.1.1.1.4 互协方差矩阵

设两个随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$ 的期望与方差存在, 则它们的 **互协方差矩阵** (*Cross Covariance Matrix*) 定义为

$$\mathbf{K}_{xy} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{y} - E[\mathbf{y}])^T]$$

提示: 当随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$ 为复数随机向量时, 互协方差矩阵定义为

$$\mathbf{K}_{xy} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{y} - E[\mathbf{y}])^H].$$

2.6.1.1.5 互相关与互协方差矩阵间关系

- 实随机向量: $\mathbf{K}_{xy} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{y} - E[\mathbf{y}])^T] = \mathbf{R}_{xy} - E[\mathbf{x}]E[\mathbf{y}]^T$
- 复随机向量: $\mathbf{K}_{xy} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{y} - E[\mathbf{y}])^H] = \mathbf{R}_{xy} - E[\mathbf{x}]E[\mathbf{y}]^H$

2.6.1.1.2 自相关

当互相关中的随机向量为同一向量时, 对应的互相关矩阵, 互协方差矩阵分别变为自相关矩阵, 自协方差矩阵.

2.6.1.1.2.1 自相关矩阵

设随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$ 的期望与方差存在, 则它的 **自相关矩阵** (*Auto Correlation Matrix*) 定义为

$$\mathbf{R}_{xx} = E[\mathbf{xx}^T] = \begin{bmatrix} E[x_1 x_1] & E[x_1 x_2] & \cdots & E[x_1 x_m] \\ E[x_2 x_1] & E[x_2 x_2] & \cdots & E[x_2 x_m] \\ \vdots & \vdots & \vdots & \vdots \\ E[x_m x_1] & E[x_m x_2] & \cdots & E[x_m x_m] \end{bmatrix}$$

提示: 当随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$ 为复数随机向量时, 自相关矩阵定义为 $\mathbf{R}_{xx} = E[\mathbf{xx}^H]$.

2.6.1.1.2.2 自协方差矩阵

设两个随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$ 的期望与方差存在, 则它们的 **自协方差矩阵** (*Auto Covariance Matrix*) 定义为

$$\mathbf{K}_{xx} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T]$$

提示: 当随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$ 为复数随机向量时, 自相关矩阵定义为

$$\mathbf{K}_{xx} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^H].$$

2.6.1.1.2.3 自相关与自协方差矩阵间关系

- 实随机向量: $\mathbf{K}_{xx} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T] = \mathbf{R}_{xx} - E[\mathbf{x}]E[\mathbf{x}]^T$
- 复随机向量: $\mathbf{K}_{xx} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^H] = \mathbf{R}_{xx} - E[\mathbf{x}]E[\mathbf{x}]^H$

2.6.2 回归分析

2.6.2.1 回归分析简介

2.6.2.1.1 什么是回归分析

回归分析 (*Regression Analysis*) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法.

设有随机向量 $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$, 以及一组观测值数据样本对 $\mathbb{S} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=0}^K$, 其中 $\mathbf{x}_i = (x_{1i}, x_{2i}, \dots, x_{ni})^T$, $\mathbf{y}_i = (y_{1i}, y_{2i}, \dots, y_{mi})^T$, $n, m, K \in \mathbb{Z}^+$, 回归分析旨在确定随机变量 \mathbf{x}, \mathbf{y} 间的关系, 这种关系的数学化称为建模. 用 \mathcal{G} 表示模型, 则:

$$\mathbf{y} = \mathcal{G}(\mathbf{x}). \quad (2.9)$$

模型可以有参数, 也可以无参数. 若模型为有参的, 记 β 为模型参数, 从而式.2.9 可表示为:

$$\mathbf{y} = \mathcal{G}(\mathbf{x}, \beta). \quad (2.10)$$

即回归分析的目的是在给出一组数据样本对 $\mathbb{S} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=0}^K$ 的情况下, 求解一个模型 \mathcal{G} , 使得该模型可以很好地拟合数据. 记 $\hat{\mathbf{y}}$ 为模型预测输出, 为评测模型的预测结果与实际观测数据在结果上的一致性, 通常定义评测准则, 如定义评测准则为损失函数 (你完全可以定义其它函数), 假设损失函数为均方误差 (Mean Squared Error, MSE):

$$L = \frac{1}{K} \sum_{i=1}^K \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2 = \frac{1}{K} \sum_{i=1}^K \|\mathcal{G}(\mathbf{x}_i) - \mathbf{y}_i\|_2^2, \quad (2.11)$$

一般通过优化式.2.11 求解模型.

提示: 本书中的符号表示与一般概率论统计书籍中有所区别:

- 随机变量: x , 随机变量取值 x , 如 $P(x = x)$
- 随机向量: \mathbf{x} , 随机向量取值 \mathbf{x} , 如 $P(\mathbf{x} = \mathbf{x})$

2.6.2.1.2 回归分析的类型

1. 若模型为无参的, 称为无参数回归 (*Nonparametric Regression*);
2. 若模型为有参的, 称为有参数回归 (*Parametric Regression*);
3. 若 $n = 1, m = 1$, 称为单重回归 (*Simple Regression*), 一个输入, 单个输出;
4. 若 $n > 1, m = 1$, 称为多重回归 (*Multiple Regression*), 多个输入, 单个输出;
5. 若 $n \in \mathbb{Z}^+, m > 1$, 称为多元回归 (*Multivariate Regression*), 多个输出;
6. 若 $\mathcal{G}(\mathbf{x}) = \beta\mathbf{x} + \beta_0$ 即满足线性关系, 称为线性回归 (*Linear Regression*)
7. 若 $\mathcal{G}(\mathbf{x}) \neq \beta\mathbf{x} + \beta_0$ 即满足非线性关系, 称为非线性回归 (*Nonlinear Regression*)
8. 投影追踪回归 (*Projection Pursuit Regression*)
9. 贝叶斯回归 (*Bayesian Regression*)

2.6.2.2 回归建模

2.6.2.2.1 回归分析简介

2.6.2.2.1.1 什么是回归分析

回归分析 (*Regression Analysis*) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法.

假设变量 $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ 和 $\mathbf{y} = (y_1, y_2, \dots, y_m)^T$ 满足模型:

$$\mathbf{y} = \mathcal{G}(\mathbf{x})$$

记 β 为模型参数, 则

$$\mathbf{y} = \mathcal{G}(\mathbf{x}, \beta)$$

回归分析的目的是在给出一组数据样本对 $S = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=0}^K$ 的情况下, 求解一个模型 \mathcal{G} , 使得该模型可以很好地拟合数据. 求解方法一般是通过定义损失函数并优化损失来求解模型参数, 如定义损失函数为均方误差:

$$L =$$

ℓ_1 范数被广泛应用于各种领域: 统计学, 信号处理, 压缩感知等等.

记 $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K)$, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K)$, 则模型

1. 若 $\mathcal{G}(\mathbf{x}) = \mathbf{Ax}$ 即表示成线性关系, 则有

Previous

Setting :confval:‘collapse_navigation’ to False

2.6.2.2.2 线性回归

2.6.2.2.2.1 什么是非线性回归

线性回归 (*Linear Regression*)

$$y = a_n x_n + a_{n-2} x_{n-2} + \dots + a_1 x_1 + a_0 x_0$$

$$y = a_n \mathbf{x}_n + a_{n-2} \mathbf{x}_{n-2} + \dots + a_1 \mathbf{x}_1 + a_0 \mathbf{x}_0$$

矩阵表示如下:

$$\mathbf{Y} = \mathbf{AX}$$

2.6.2.2.2 多项式线性回归

$$y = a_n x_n + a_{n-2} x_{n-2} + \cdots + a_1 x_1 + a_0 x_0$$

2.6.2.2.3 非线性回归

2.6.2.2.3.1 什么是非线性回归

非线性回归 (*Nonlinear Regression*)

2.6.2.2.4 投影追踪回归

2.6.2.2.4.1 什么是投影追踪回归

Friedman 等人于 1974 年提出投影追踪 (*Projection Pursuit*, PP) [1], 并于 1981 年提出投影追踪回归 (*Projection Pursuit Regression*, PPR) [2], 这是一种单输出回归模型, 1985 年提出, 多输出回归分类模型 [3]. 有关投影追踪参见投影追踪 (页 113).

Definition 41 (投影追踪回归) 投影追踪回归模型将随机向量 \mathbf{y} 建模为随机向量 \mathbf{x} 的随机变量的加权和的不同非线性映射的和, 即:

$$\mathbf{y} = \mathbf{w}_0 + \sum_{j=1}^r f_j(\mathbf{w}_j^T \mathbf{x}) + \epsilon,$$

其中, $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ 为自变量, $\mathbf{w}_j = (w_{j1}, w_{j2}, \dots, w_{jn})^T$ 为模型参数, $f_j, j = 1, 2, \dots, r$ 为 r 个非线性映射函数 (可以相同或不同), ϵ 为模型估计误差.

给定 K 个观测样本对 $\mathbb{S} = \{\mathbf{x}_i, y_i\}_{i=0}^K$, 定义误差函数:

$$\min_{f_j, \mathbf{w}_j} L = \sum_{i=1}^K \left(y_i - \sum_{j=1}^r f_j(\mathbf{w}_j^T \mathbf{x}_i) \right)^2 \quad (2.12)$$

式 2.12 可以通过交替优化 (*Alternating Optimization*) 的方法进行优化求解, 投影追踪回归采用投影追踪交替优化算法进行求解, 具体过程参见投影追踪 (页 113).

提示: 投影追踪回归与神经网络的区别:

- 理论上都可以逼近任意连续函数;
- 投影追踪回归的非线性映射可以不同, 神经网络的每层的非线性映射一般相同;
- 投影追踪回归的非线性映射每次估计一次, 接着更新权重, 而神经网络的非线性映射一般是预先指定并同时估计的.
- 有关反向传播 (Back-Propagation Learning, BPL) 和投影追踪学习 (Projection Pursuit Learning, PPL), 可参见文献 [6][9][7].

2.6.2.2.4.2 实验与分析

2.6.2.2.4.3 回归实验

2.6.2.2.4.4 实验代码

从 这 里 [projection-pursuit benchmarks](#) (<https://github.com/pavelkomarov/projection-pursuit/tree/master/benchmarks>) 获得 `compare_to_neural_network.py`

2.6.2.2.4.5 实验结果

r 的影响.

```
hidden_layer_sizes, max_iter: 200 200
r, stage_maxiter, backfit_maxiter: 20 200 200
Average squared error per training example for MLPRegressor: 33.561744644172634
Average squared error per training example for ProjectionPursuitRegressor: 0.
↪7212472748844135
Average squared error per testing example for MLPRegressor: 22.095682612129952
Average squared error per testing example for ProjectionPursuitRegressor: 23.
↪26310438760661
Training time for MLPRegressor: 0.6172130107879639
Training time for ProjectionPursuitRegressor: 93.76978826522827

hidden_layer_sizes, max_iter: 200 200
r, stage_maxiter, backfit_maxiter: 10 200 200
Average squared error per training example for MLPRegressor: 25.894054755430098
Average squared error per training example for ProjectionPursuitRegressor: 2.
↪0112607496432418
Average squared error per testing example for MLPRegressor: 46.90504990729163
Average squared error per testing example for ProjectionPursuitRegressor: 23.
↪705301000070467
Training time for MLPRegressor: 0.6327333450317383
Training time for ProjectionPursuitRegressor: 26.458088159561157

hidden_layer_sizes, max_iter: 200 200
r, stage_maxiter, backfit_maxiter: 5 200 200
Average squared error per training example for MLPRegressor: 25.394874641020802
Average squared error per training example for ProjectionPursuitRegressor: 3.
↪9056643659649666
Average squared error per testing example for MLPRegressor: 36.865379854954945
Average squared error per testing example for ProjectionPursuitRegressor: 20.
↪50523972734486
Training time for MLPRegressor: 0.6016314029693604
Training time for ProjectionPursuitRegressor: 12.463459730148315
```

隐层神经元数的影响:

```

hidden_layer_sizes, max_iter: 200 200
r, stage_maxiter, backfit_maxiter: 5 200 200
Average squared error per training example for MLPRegressor: 25.394874641020802
Average squared error per training example for ProjectionPursuitRegressor: 3.
↪9056643659649666
Average squared error per testing example for MLPRegressor: 36.865379854954945
Average squared error per testing example for ProjectionPursuitRegressor: 20.
↪50523972734486
Training time for MLPRegressor: 0.6016314029693604
Training time for ProjectionPursuitRegressor: 12.463459730148315

hidden_layer_sizes, max_iter: 512 200
r, stage_maxiter, backfit_maxiter: 5 200 200
Average squared error per training example for MLPRegressor: 19.539311275178886
Average squared error per training example for ProjectionPursuitRegressor: 4.
↪754953589272597
Average squared error per testing example for MLPRegressor: 39.934009948538275
Average squared error per testing example for ProjectionPursuitRegressor: 20.
↪004689504319465
Training time for MLPRegressor: 1.225292682647705
Training time for ProjectionPursuitRegressor: 14.15251898765564

hidden_layer_sizes, max_iter: 1024 200
r, stage_maxiter, backfit_maxiter: 5 200 200
Average squared error per training example for MLPRegressor: 21.09513135671064
Average squared error per training example for ProjectionPursuitRegressor: 4.
↪805062370738948
Average squared error per testing example for MLPRegressor: 16.876270527353554
Average squared error per testing example for ProjectionPursuitRegressor: 18.
↪131646272884936
Training time for MLPRegressor: 3.123934745788574
Training time for ProjectionPursuitRegressor: 10.765942096710205

```

迭代次数的影响::

```

hidden_layer_sizes, max_iter: 200 1000
r, stage_maxiter, backfit_maxiter: 5 200 200
Average squared error per training example for MLPRegressor: 17.596837460636106
Average squared error per training example for ProjectionPursuitRegressor: 4.
↪219905575343612
Average squared error per testing example for MLPRegressor: 27.622677425433874
Average squared error per testing example for ProjectionPursuitRegressor: 14.
↪879565144602562
Training time for MLPRegressor: 1.9046266078948975
Training time for ProjectionPursuitRegressor: 14.236632347106934

hidden_layer_sizes, max_iter: 200 2000
r, stage_maxiter, backfit_maxiter: 5 200 200

```

(下页继续)

(续上页)

```
Average squared error per training example for MLPRegressor: 17.660446224053825
Average squared error per training example for ProjectionPursuitRegressor: 9.
→443822651593512
Average squared error per testing example for MLPRegressor: 27.116718449437933
Average squared error per testing example for ProjectionPursuitRegressor: 112.
→54856398900877
Training time for MLPRegressor: 1.456831932067871
Training time for ProjectionPursuitRegressor: 19.396251440048218
```

2.6.2.2.4.6 分类实验

2.6.2.2.4.7 实验代码

从 这 里 [projection-pursuit](#) [benchmarks](#) (<https://github.com/pavelkomarov/projection-pursuit/tree/master/benchmarks>) 获得 compare_classifier.py

2.6.2.2.4.8 实验结果

2.6.2.2.5 贝叶斯回归

2.6.2.2.5.1 什么是贝叶斯回归

贝叶斯回归 (*Bayesian Regression*)

2.6.2.3 估计方法

2.6.2.3.1 简介

2.6.2.3.2 岭回归

2.6.2.3.3 LASSO

2.6.2.3.3.1 什么是 LASSO

最小绝对收缩与选择算子 (*Least Absolute Shrinkage and Selection Operator* , LASSO)

2.6.2.3.4 基追踪

2.6.2.3.4.1 什么是基追踪

基追踪 (*Basis Pursuit*)

2.6.2.3.5 投影追踪

2.6.2.3.5.1 什么是投影追踪

投影追踪 (*Projection Pursuit*, PP) 最初由 Friedman 等人与 1974 年提出 [1], Friedman 等人继而提出基于投影追踪的回归与分类模型 [2][3][4]. 投影追踪回归分类模型与神经网络有着联系与区别, 一些研究者研究了基于投影追踪的神经网络学习 [5][6][7]. 关于投影追踪的详细清晰的介绍与推导可以参考 [8][9][10]

提示: 投影追踪可用于:

- 主成份分析
- 回归
- 分类
- 概率密度估计

2.6.2.3.5.2 实验与分析

- python implementation of projection pursuit :An implementation of multivariate projection pursuit regression and univariate projection pursuit regression
- [github code](https://github.com/pavelkomarov/projection-pursuit) (<https://github.com/pavelkomarov/projection-pursuit>)
- [api doc](https://pavelkomarov.com/projection-pursuit/) (<https://pavelkomarov.com/projection-pursuit/>)
- [math doc](https://github.com/pavelkomarov/projection-pursuit/blob/master/doc/math.pdf) (<https://github.com/pavelkomarov/projection-pursuit/blob/master/doc/math.pdf>)

2.6.3 参考文献

2.6.4 名词术语

Alternating Optimization 交替优化 ([Alternating Optimization](http://en.volupedia.org/wiki/Alternating_optimization) (http://en.volupedia.org/wiki/Alternating_optimization))

Auto Correlation 自相关 ([Auto Correlation Matrix](http://en.volupedia.org/wiki/Autocorrelation) (<http://en.volupedia.org/wiki/Autocorrelation>))

Auto Correlation Function 自相关矩阵 ([Auto Correlation Function](http://en.volupedia.org/wiki/Autocorrelation#Autocorrelation_of_stochastic_processes) (http://en.volupedia.org/wiki/Autocorrelation#Autocorrelation_of_stochastic_processes))

Auto Correlation Matrix 自相关矩阵 ([Auto Correlation Matrix](http://en.volupedia.org/wiki/Autocorrelation_matrix) (http://en.volupedia.org/wiki/Autocorrelation_matrix)),

Auto Covariance Matrix 自协方差矩阵 ([Auto Covariance Matrix](http://en.volupedia.org/wiki/Covariance_matrix) (http://en.volupedia.org/wiki/Covariance_matrix)) ,

Basis Pursuit 基追踪 ([Basis Pursuit](http://en.volupedia.org/wiki/Basis_pursuit) (http://en.volupedia.org/wiki/Basis_pursuit) , BP)

Bayesian Linear Regression 贝叶斯线性回归 (Bayesian Linear Regression (http://en.volupedia.org/wiki/Bayesian_linear_regression))

Bayesian Regression 贝叶斯回归 ([Bayesian Regression](http://en.volupedia.org/wiki/Bayesian_regression) (http://en.volupedia.org/wiki/Bayesian_regression))

Cross Correlation 互相关 ([Cross Correlation](http://en.volupedia.org/wiki/Cross-correlation) (<http://en.volupedia.org/wiki/Cross-correlation>))

Cross Correlation Function 互相关函数 (Cross Correlation Function (http://en.volupedia.org/wiki/Cross-correlation#Cross-correlation_of_stochastic_processes))

Cross Correlation Matrix 互相关矩阵 (Cross Correlation Matrix (http://en.volupedia.org/wiki/Cross-correlation_matrix)) ,

Cross Covariance Matrix 互协方差矩阵 (Cross Covariance Matrix (http://en.volupedia.org/wiki/Cross-covariance_matrix)) ,

Least Absolute Shrinkage and Selection Operator 最小绝对收缩与选择算子 ([Least Absolute Shrinkage and Selection Operator](http://en.volupedia.org/wiki/Lasso_(statistics)) ([http://en.volupedia.org/wiki/Lasso_\(statistics\)](http://en.volupedia.org/wiki/Lasso_(statistics)))) , LASSO)

Linear Regression 线性回归 ([Linear Regression](http://en.volupedia.org/wiki/Linear_regression) (http://en.volupedia.org/wiki/Linear_regression))

Matching Pursuit 匹配追踪 ([Matching Pursuit](http://en.volupedia.org/wiki/Matching_pursuit) (http://en.volupedia.org/wiki/Matching_pursuit)) , MP)

Multiple Regression 多重回归 ([Multiple Regression](http://en.volupedia.org/wiki/Multiple_regression) (http://en.volupedia.org/wiki/Multiple_regression))

Multivariate Regression 多元回归 ([Multivariate Regression](http://en.volupedia.org/wiki/Multivariate_regression) (http://en.volupedia.org/wiki/Multivariate_regression)) 也称多变量回归

Nonlinear Regression 非线性回归 ([Nonlinear Regression](http://en.volupedia.org/wiki/Nonlinear_regression) (http://en.volupedia.org/wiki/Nonlinear_regression))

Nonparametric Regression 无参回归 ([Nonparametric Regression](http://en.volupedia.org/wiki/Nonparametric_regression) (http://en.volupedia.org/wiki/Nonparametric_regression))

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

Parametric Regression 有参回归 ([Parametric Regression](http://en.volupedia.org/wiki/Parametric_regression) (http://en.volupedia.org/wiki/Parametric_regression))

Projection Pursuit 投影追踪 ([Projection Pursuit](http://en.volupedia.org/wiki/Projection_pursuit) (http://en.volupedia.org/wiki/Projection_pursuit)) , PP)

Projection Pursuit Regression 投影追踪回归 (Projection Pursuit Regression (http://en.volupedia.org/wiki/Projection_pursuit_regression)) , PPR)

Regression Analysis 回归分析 ([Regression Analysis](http://en.volupedia.org/wiki/Regression_analysis) (http://en.volupedia.org/wiki/Regression_analysis))

Simple Regression 单重回归 ([Simple Regression](http://en.volupedia.org/wiki/Simple_regression) (http://en.volupedia.org/wiki/Simple_regression))

2.7 随机过程

2.7.1 概率论基础

2.7.1.1 概率空间与随机变量

2.7.1.1.1 柯尔莫哥洛夫概率公理化体系

概率论公理化有三种学派：主观（任何命题都看作事件）、客观（频率的极限）、以测度论为基础（柯尔莫哥洛夫）

2.7.1.1.1.1 是什么

- 是一种概率论公理化方法，结合了集合论、测度论、实变函数论（与复变函数论关系？）；
- 是一个概率空间 $(\Omega, \mathcal{F}, \mathbb{P})$ 的可测描述；
- 测度为概率、可测函数为随机变量、全直线对应概率空间、点集对应事件集；

2.7.1.1.1.2 为什么

19世纪下半叶，概率论自身有了一定的发展，但诸如概率、收敛性、随机变量数学期望等没有严格定义。

- 为什么要有公理化方法：
- 为什么会想到用测度论：

2.7.1.1.1.3 怎么办

- 怎么结合集合论、测度论、实变函数论公理化概率论
- 怎么用这种公理化系统
- 1. 概率空间

概率空间可以用一个三元组表示为 $(\Omega, \mathcal{F}, \mathbb{P})$ ，即样本空间 Ω 上的事件 \mathcal{F} 对应的概率测度 \mathbb{P} ，其中：

- **事件域：**事件域 \mathcal{F} 为样本空间 Ω 的子集生成的 σ -代数，即满足：
 - 空集是任意集合的子集： $\emptyset \in \mathcal{F}$
 - 补运算封闭性：若 $A \in \mathcal{F}$ ，则 $A^c \in \mathcal{F}$
 - 并集运算封闭性：若 $A_1, A_2, \dots, A_n, \dots \subset \mathcal{F}$ ，则 $\bigcup_{n=1}^{\infty} A_n \in \mathcal{F}$
- **概率测度：**概率测度 \mathbb{P} 是定义在事件域 \mathcal{F} 上，取值为 $[0, 1]$ 的函数
 - 非负性： $P(A) \geq 0$
 - 规范性：测度空间为 1， $P(\Omega) = 1$

- 可列（完全）可加性：相互独立事件联合概率为各自概率和，若 $A_1, A_2, \dots, A_n, \dots \subset \mathcal{F}$ 且两两互不相交 ($A_m \cap A_n = \emptyset, m \neq n$)

提示： σ -代数又称 σ -域，是一个集合，是一个满足对交并补运算封闭的集合（数域是满足四则运算封闭的集合），集合 X 上的 σ -代数是集合 X 的所有子集集合（幂集）的一个子集。

2. 随机变量

设有概率空间 $(\Omega, \mathcal{F}, \mathbb{P})$ ，随机变量 $X(\omega)$ 为样本空间 Ω 上的取值在 \mathbb{R}^d 上的 \mathcal{F} 可测函数，即 $X(\omega) : \Omega \rightarrow \mathbb{R}^d$

2.7.1.1.4 还有什么

- 随机性的本质依然未解决
- 随机性与确定性界限
- 定义也是人为定义的，未必是严密的

2.7.1.1.2 傅立叶变换

2.7.2 随机过程基础

2.7.2.1 What is ?

2.7.2.1.1 dddddddd

2.7.2.1.1.1 ssssssssssss

2.7.2.2 Why ?

2.7.2.2.1 dddddddd

2.7.2.2.1.1 ssssssssssss

2.7.3 布朗运动

2.7.3.1 What is ?

2.7.3.1.1 dddddddd

2.7.3.1.1.1 ssssssssssss

2.7.3.2 Why ?

2.7.3.2.1 dddddddd

2.7.3.2.1.1 ssssssssssss

2.7.4 跳跃随机过程

2.7.4.1 What is ?

2.7.4.1.1 dddddddd

2.7.4.1.1.1 ssssssssssss

2.7.4.2 Why ?

2.7.4.2.1 dddddddd

2.7.4.2.1.1 ssssssssssss

2.7.5 平稳过程

2.7.5.1 What is ?

2.7.5.1.1 dddddddd

2.8 拓扑学

2.8.1 名词术语

Convolution 卷积 ([Convolution](http://en.volupedia.org/wiki/Convolution) (<http://en.volupedia.org/wiki/Convolution>))

Convolution 卷积 ([Circular Convolution](http://en.volupedia.org/wiki/Circular_convolution) (http://en.volupedia.org/wiki/Circular_convolution))

Functional Analysis 泛函分析 ([Functional Analysis](http://en.volupedia.org/wiki/Functional_analysis) (http://en.volupedia.org/wiki/Functional_analysis))

2.9 泛函分析

2.9.1 线性算子

2.9.1.1 卷积算子

2.9.1.1.1 普通卷积

在数学中, **卷积** ([Convolution](#)) 是一种由两个函数 (f, g) 产生第三个函数 (h) 的操作, 即将两个函数 (f, g) 中的一个进行翻转和平移不同量后与另一个分别相乘后再累加. 与相关分析类似, 不同的是在相关分析中, 无需进行翻转操作.

警告: 卷积神经网络中的卷积 (convolution) 运算实际上是互相关 (cross-correlation) 运算, 详见 [卷积单元](#) (页 343) 小节.

2.9.1.1.1.1 一维卷积

设有单变量连续函数 $f(t), g(t)$, 两者之间的卷积可以表示为

$$\begin{aligned} (f * g)(t) &= \int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau \\ &= \int_{-\infty}^{+\infty} f(t - \tau)g(\tau)d\tau \end{aligned} \tag{2.13}$$

单变量离散序列 $f[n], g[n]$, 两者之间的卷积可以表示为

$$\begin{aligned} (f * g)[n] &= \sum_{m=-\infty}^{+\infty} f[m]g[n - m] \\ &= \sum_{m=-\infty}^{+\infty} f[n - m]g[m] \end{aligned} \tag{2.14}$$

一维卷积具有如下性质 (以连续卷积为例)

- 分配律: $f_1(t) * (f_2(t) + f_3(t)) = f_1(t) * f_2(t) + f_1(t) * f_3(t) \Leftrightarrow h(t) = h_1(t) + h_2(t)$
- 结合律: $(f_1(t) * f_2(t)) * f_3(t) = f_1(t) * (f_2(t) * f_3(t)) \Leftrightarrow h(t) = h_1(t) * h_2(t)$

- 交换律: $f_1(t) * f_2(t) = f_2(t) * f_1(t)$
- 时域卷积对应频域相乘: $f_1(t) * f_2(t) \Leftrightarrow F_1(\omega)F_2(\omega)$
- 频域卷积对应时域相乘: $F_1(\omega) * F_2(\omega) \Leftrightarrow 2\pi f_1(t)f_2(t)$
- 卷积的微分: $\frac{d}{dt} [f_1(t) * f_2(t)] = f_1(t) * \frac{df_2(t)}{dt} + f_2(t) * \frac{df_1(t)}{dt}$
- 卷积的积分: $\int_{-\infty}^t f_1(\tau) * f_2(\tau) d\tau = f_1(t) * \int_{-\infty}^t f_2(\tau) d\tau = f_2(t) * \int_{-\infty}^t f_1(\tau) d\tau$

任意函数与冲击响应函数卷积的性质(以连续卷积为例)

- 任意函数与单位冲激信号卷积结果为自身: $f(t) * \delta(t) = \int_{-\infty}^{+\infty} f(\tau)\delta(t-\tau)d\tau = f(t)$
- 任意函数与时延为 t_d 的单位冲激信号卷积结果为自身时延 t_d : $f(t) * \delta(t-t_d) = \int_{-\infty}^{+\infty} f(\tau)\delta(t-t_d-\tau)d\tau = f(t-t_d)$
- 任意函数与单位冲激信号导数的卷积结果为自身的导数: $f(t) * \delta^{(k)}(t) = f^{(k)}(t)$
- 任意函数与时延为 t_d 单位冲激信号导数的卷积结果为自身时延 t_d 导数: $f(t) * \delta^{(k)}(t-t_d) = f^{(k)}(t-t_d)$

2.9.1.1.2 二维卷积

对于双变量连续函数 $f(x, y), g(x, y)$, 它们之间的二维卷积定义为

$$(f * g)(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(u, v)g(x-u, y-v) du dv \\ = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x-u, y-v)g(u, v) du dv \quad (2.15)$$

2.9.1.1.2 回旋卷积

回旋卷积 (Circular Convolution) 也称 循环卷积, 即当 g 为周期函数时的卷积, 卷积结果也是周期的.

2.9.1.1.2.1 连续形式

设有周期为 T 的函数 g , 定义如下连续形式循环卷积

$$f(t) * g_T(t) = \int_{t_0}^{t_0+T} \left(\sum_{k=-\infty}^{+\infty} f(\tau + kT) \right) g_T(t-\tau) d\tau$$

2.9.1.1.2.2 离散形式

设有周期为 T 的函数 g , 定义如下离散形式循环卷积

$$f[n] * g_N[n] = \sum_{m=0}^{N-1} \left(\sum_{k=-\infty}^{+\infty} f[m+kN] \right) g_N[n-m]$$

2.9.1.1.3 实验分析

2.9.1.1.3.1 二维卷积

2.9.1.2 相关算子

2.9.1.2.1 互相关

互相关 (*Cross Correlation*) 在不同领域有稍微不同的定义.

在数字信号处理中, 互相关两个序列相似性的度量, 是一个序列相对于另一序列的位移函数. 通常被称为滑动点积或滑动内积, 且用于搜索较长信号的较短的已知特征, 在模式识别, 单粒子分析, 电子断层扫描, 密码分析和神经生理学中有广泛应用. 互相关与卷积本质上相似, 自相关是信号自己的互相关.

2.9.1.2.1.1 确定信号的互相关

对于连续函数 f, g , 互相关定义为

$$(f \star g)(\tau) = \int_{-\infty}^{+\infty} \overline{f(t)}g(t + \tau)dt = \int_{-\infty}^{+\infty} \overline{f(t - \tau)}g(t)dt \quad (2.16)$$

其中, \bar{f} 表示 f 的共轭转置, τ 表示移位.

类似的, 对于离散函数 f, g , 互相关定义为

$$(f \star g)[n] = \sum_{m=-\infty}^{+\infty} \overline{f[m]}g[m + n] = \sum_{m=-\infty}^{+\infty} \overline{f[m - n]}g[m] \quad (2.17)$$

2.9.1.2.1.2 不确定信号的互相关

参见章节 [相关分析](#) (页 104) 的 [相关的概念](#) (页 104) 小节.

2.9.1.2.1.3 周期信号的互相关

2.9.1.2.2 自相关

2.9.1.3 卷积与相关

2.9.1.3.1 卷积与相关关系

卷积与相关关系示意图

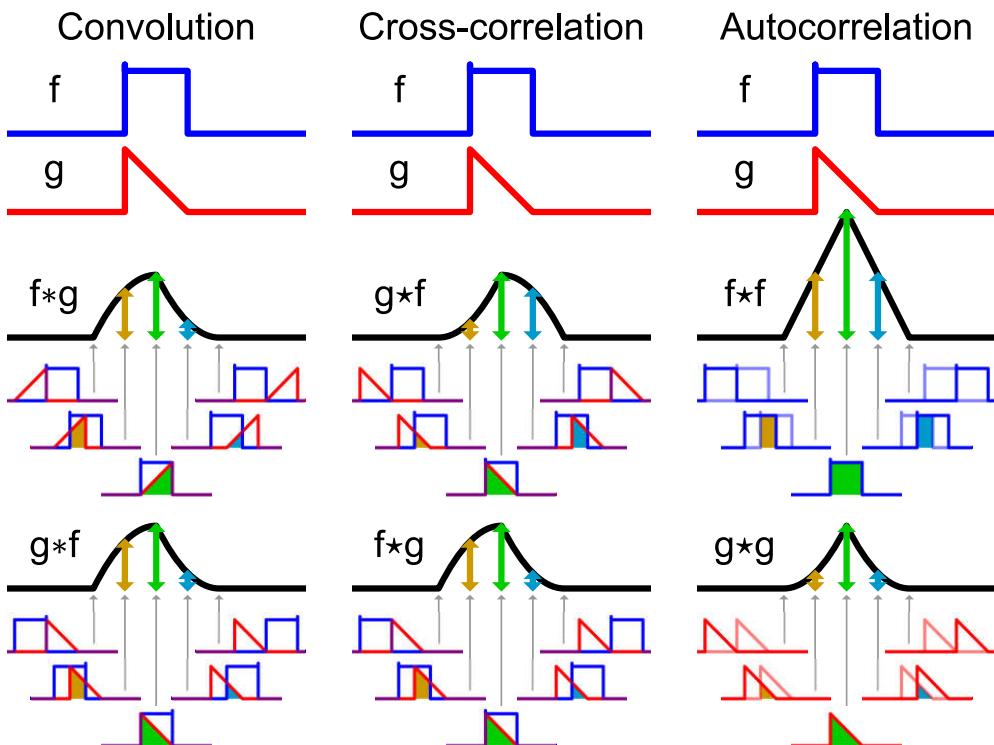


图 2.18: Visual comparison of convolution, cross-correlation and autocorrelation (from wiki).

For the operations involving function f , and assuming the height of f is 1.0, the value of the result at 5 different points is indicated by the shaded area below each point. Also, the vertical symmetry of f is the reason $f * g$, $f * g$ are identical in this example.

2.9.2 名词术语

Convolution 卷积 ([Convolution](http://en.volupedia.org/wiki/Convolution) (<http://en.volupedia.org/wiki/Convolution>))

Convolution 卷积 ([Circular Convolution](http://en.volupedia.org/wiki/Circular_convolution) (http://en.volupedia.org/wiki/Circular_convolution))

Functional Analysis 泛函分析 ([Functional Analysis](http://en.volupedia.org/wiki/Functional_analysis) (http://en.volupedia.org/wiki/Functional_analysis))

2.10 模糊数学

2.10.1 引言

由于模糊性概念已经找到了模糊集的描述方式，人们运用概念进行判断、评价、推理、决策和控制的过程也可以用模糊性数学的方法来描述。例如模糊聚类分析、模糊模式识别、模糊综合评判、模糊决策与模糊预测、模糊控制、模糊信息处理等。这些方法构成了一种模糊性系统理论，构成了一种思辨数学的雏形，它已经在医学、气象、心理、经济管理、石油、地质、环境、生物、农业、林业、化工、语言、控制、遥感、教育、体育等方面取得具体的研究成果。

模糊数学的基本思想就是：用精确的数学手段对现实世界中大量存在的模糊概念和模糊现象进行描述、建模，以达到对其

来自百度百科

2.10.2 模糊集合

2.10.2.1 模糊集合

2.10.2.1.1 引言 (不确定集合)

在经典集合论中 (参见集合的概念与性质 (页 10)), 元素 x 是否属于集合 S 是确定的, 且只有两种情况, 即属于或不属于. 如果元素 x 以一定的可能性属于或不属于集合 S 呢? 这就是 **模糊集合** (*Fuzzy Set*).

Zadeh 于 1965 年提出模糊集合 [4] , 作为对经典集合的扩展, 在模糊集理论中, 经典集合通常被称为 **明确集合** (*Crisp Set*), 通过引入隶属函数 (*Membership Function*) 与隶属度 (*Membership Degree*), 来衡量元素的模糊性, 从而明确集为模糊集的特例, 即隶属函数仅有两个值 (0, 1), 隶属度为 0 或 1.

注解： 比如, 对于 “58 岁是不是老年人” 这个问题, 很难说 “是” 还是 “不是”, 很容易想到的方法是用 $[0, 1]$ 之间的一个数来表示, 如 0.8 表示: 58 岁的人有 80% 的可能性是老年人. 如果用 $[0, 100]$ 来表示岁数, 那么一个岁数为 $x \in [0, 100]$ 的人, 是否是老年人可以用隶属函数 $f(x) = x/100$ 表示, 隶属度为 $f(x)$ 的值. 然而, 对于超过 100 岁的人呢, 此时 $f(x) > 1$.

容易发现, 实际应用中, 我们很难确认隶属函数的具体形式. 因而, Pawlak 于 1982 年提出基于等价关系的近似表示方法, 即用两个隶属函数的上下近似来表示, 从而引入 **粗糙集** (*Rough Set*) 的概念 [5] .

2.10.2.1.2 模糊集的定义

Definition 42 (明确集合) 明确集合 的定义可以扩展为集合 (或论域) \mathbb{X} 和映射 $f : \mathbb{X} \rightarrow \{0, 1\}$ 的二元组 $\mathbb{A} = (\mathbb{X}, f)$, 其中, f 为集合 \mathbb{A} 的特征函数 (也称示性函数, 参见集合的特征函数 (页 12))

$$f(x) = \begin{cases} 1 & \text{if } x \in \mathbb{A} \\ 0 & \text{if } x \notin \mathbb{A} \end{cases}.$$

扩展上述定义, 可以得到模糊集合的定义

Definition 43 (模糊集合) 设有集合 (或论域) \mathbb{X} 和映射 $f : \mathbb{X} \rightarrow [0, 1]$ 的组合, 二元组 $\mathbb{A} = (\mathbb{X}, f)$ 确定了一个集合, 称为 模糊集合 (Fuzzy Set). 函数 $f(x) = \mu_{\mathbb{A}}$ 称为模糊集 \mathbb{A} 的 隶属函数 (Membership Function). 对于 $\forall x \in \mathbb{X}$, 称 $f(x)$ 为元素 x 在模糊集 \mathbb{A} 中的 隶属度 (Membership Degree, Membership Grade). 记 $F(\mathbb{X})$ 为集合 \mathbb{X} 的全体模糊子集.

举例说明如何使用模糊集对语言表达进行建模, 如要表示 “年轻” 的概念, 可以用模糊集 $\mathbb{A} = (\mathbb{X}, f)$ 表示, 其中岁数集合为 $\in [0, 100]$, 隶属函数可以表示为

$$\mu_{\mathbb{A}} = \begin{cases} 1 & \text{if } 0 \leq x \leq 20 \\ \frac{40-x}{20} & \text{if } 20 \leq x \leq 40 \\ 0 & \text{otherwise} \end{cases}$$

提示: 明确集相当于模糊集的特例, 即当模糊集的隶属函数取明确集的特征函数时 (即: $f : \mathbb{X} \rightarrow \{0, 1\}$), 模糊集退化为明确集.

geq

2.10.2.1.3 模糊集与明确集

给定模糊集 $\mathbb{A} = (\mathbb{X}, f)$, $\alpha \in [0, 1]$, 有如下明确集:

- **α -割集** (α -cut set, α -level set): $\mathbb{A}^{\geq \alpha} = \mathbb{A}_{\alpha} = \{x \in \mathbb{X} | f(x) \geq \alpha\}$
- **强 α -割集** (strong α -cut set, strong α -level set): $\mathbb{A}^{> \alpha} = \mathbb{A}'_{\alpha} = \{x \in \mathbb{X} | f(x) > \alpha\}$
- **紧集** (support set): $S(\mathbb{A}) = Supp(\mathbb{A}) = \mathbb{A}^{> 0} = \{x \in \mathbb{X} | f(x) > 0\}$
- **核集** (core set, kernel set): $C(\mathbb{A}) = Core(\mathbb{A}) = \mathbb{A}^{=1} = \{x \in \mathbb{X} | f(x) = 1\}$

2.10.2.1.4 模糊集的表示

记模糊集 \mathbb{A} , 隶属函数为 $\mu_{\mathbb{A}}$, 元素 x 属于模糊集 \mathbb{A} 的隶属度为 $\mu_{\mathbb{A}}(x)$, 简记为 $\mu(x)$. 则有以下表示方法.

2.10.2.1.4.1 Zadeh 表示法

Zadeh 表示法如下:

$$\mathbb{A} = \frac{\mu(x_1)}{x_1} + \frac{\mu(x_2)}{x_2} + \cdots + \frac{\mu(x_n)}{x_n},$$

或

$$\mathbb{A} = \mu(x_1)/x_1 + \mu(x_2)/x_2 + \cdots + \mu(x_n)/x_n,$$

其中, 符号 $+$, $/$ 不表示四则运算中的加法和除法, $\frac{\mu(x_i)}{x_i}$ 表示元素 x_i 隶属于模糊集 \mathbb{A} 的隶属度是 $\mu(x_i)$.

若集合 \mathbb{A} 为无限集合, 则有以下表示方法

$$\mathbb{A} = \int \frac{\mu(x)}{x}$$

2.10.2.1.4.2 序偶表示法

$$\mathbb{A} = \{(x_1, \mu(x_1)), (x_2, \mu(x_2)), \dots, (x_n, \mu(x_n))\},$$

2.10.2.1.4.3 向量表示法

$$\mathbb{A} = (\mu(x_1), \mu(x_2), \dots, \mu(x_n))$$

2.10.2.1.5 特殊模糊集

由于模糊集合用隶属度表征, 因而对于两个模糊集合 \mathbb{A}, \mathbb{B} , 其隶属函数分别为 $\mu_{\mathbb{A}}, \mu_{\mathbb{B}}$, 则

- 空集 (Empty Set): $\mathbb{A} = \emptyset \Leftrightarrow \mu_{\mathbb{A}}(x) = 0$
- 全集 (Full Set): $\mathbb{A} = \mathbb{E} \Leftrightarrow \mu_{\mathbb{A}}(x) = 1$
- 子集 (Subset): $\mathbb{B} \subset \mathbb{A} \Leftrightarrow \mu_{\mathbb{B}}(x) \leq \mu_{\mathbb{A}}(x)$
- 相等 (equal): $\mathbb{A} = \mathbb{B} \Leftrightarrow \mu_{\mathbb{A}}(x) = \mu_{\mathbb{B}}(x)$

2.10.2.1.6 模糊集的运算

2.10.2.1.6.1 模糊算子

由于模糊集合用隶属度表征, 因而对于两个模糊集合 \mathbb{A}, \mathbb{B} , 其隶属函数分别为 $\mu_{\mathbb{A}}, \mu_{\mathbb{B}}$, 一般地, 定义

- 交 (Intersection): $\mathbb{A} \cap \mathbb{B} \Leftrightarrow \mu_{\mathbb{A} \cap \mathbb{B}}(x) = \min(\mu_{\mathbb{A}}(x), \mu_{\mathbb{B}}(x)) = \mu_{\mathbb{A}}(x) \wedge \mu_{\mathbb{B}}(x)$
- 并 (Union): $\mathbb{A} \cup \mathbb{B} \Leftrightarrow \mu_{\mathbb{A} \cup \mathbb{B}}(x) = \max(\mu_{\mathbb{A}}(x), \mu_{\mathbb{B}}(x)) = \mu_{\mathbb{A}}(x) \vee \mu_{\mathbb{B}}(x)$

- 补 (Complement): $\bar{A} = \mathbb{B} \Leftrightarrow \mu_{\bar{A}}(x) = 1 - \mu_A(x)$

两个模糊集合的运算的实质是隶属度函数的运算, 上述定义中采用 \max, \min 常用模糊算子, 也可以采用其它算子.

- 交 (Intersection): $C = A \cap B$

- 模糊交算子: $\mu_C(x) = \min(\mu_A(x), \mu_B(x))$
- 代数积算子: $\mu_C(x) = \mu_A(x) \cdot \mu_B(x)$
- 有界积算子: $\mu_C(x) = \max(0, \mu_A(x) + \mu_B(x) - 1)$

- 并 (Union): $C = A \cup B$

- 模糊并算子: $\mu_C(x) = \max(\mu_A(x), \mu_B(x))$
- 代数和算子: $\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$
- 有界和算子: $\mu_C(x) = \min(1, \mu_A(x) + \mu_B(x) - 1)$

- 平衡算子: $\mu_C(x) = [\mu_A(x) \cdot \mu_B(x)]^{1-\gamma} \cdot [1 - (1 - \mu_A(x))(1 - \mu_B(x))]$, $\gamma \in [0, 1]$

提示: 取最大最小不可避免地会丢失信息, 平衡算子可以起到信息补偿作用.

2.10.2.1.6.2 模糊集间的运算定律

1. 交换律: $A \cap B, A \cup B$
2. 结合律: $A \cap (B \cap C) = (A \cap B) \cap C, A \cup (B \cup C) = (A \cup B) \cup C$
3. 分配律: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C), A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
4. 对偶律: $\overline{A \cup B} = \bar{A} \cap \bar{B}, \overline{A \cap B} = \bar{A} \cup \bar{B}$
5. 两极律: $A \cap E = A, A \cup E = E$
6. 零一律: $A \cap \emptyset = \emptyset, A \cup \emptyset = A$
7. 吸收律: $A \cap (A \cup B) = A, A \cup (A \cap B) = A$
8. 等幂律: $A \cap A = A, A \cup A = A$
9. 复原率: $\bar{\bar{A}} = A$

2.10.2.2 隶属函数

2.10.2.2.1 什么是隶属函数

隶属函数 (*Membership Function*) 是模糊集合的核心概念, 反映了元素属于模糊集合的隶属度 (*Membership Degree, Membership Grade*). 模糊集的隶属函数与明确集的指示函数 (*Indicator function*) 或特征函数 (*characteristic function*) 相对应.

The membership function of a fuzzy set is a generalization of the indicator function in classical sets. In fuzzy logic, it represents the degree of truth as an extension of valuation. Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition. Membership functions were introduced by Zadeh in the first paper on fuzzy sets (1965). Zadeh, in his theory of fuzzy sets, proposed using a membership function (with a range covering the interval (0,1)) operating on the domain of all possible values.

Definition 44 (隶属函数) 对于任意集合 \mathbb{X} , 其上的 隶属函数 为从 \mathbb{X} 到区间 $[0, 1]$ 的任意映射函数 $f : \mathbb{X} \rightarrow [0, 1]$.

隶属函数与模糊集合一一对应, 模糊集合 $\mathbb{A} \subset \mathbb{X}$ 的隶属函数通常表示为 $\mu_{\mathbb{A}}$, 对于 $x \in \mathbb{X}$, $\mu_{\mathbb{A}}(x)$ 表示隶属度, $\mu_{\mathbb{A}}(x) = 0$ 意味着 x 不属于 \mathbb{A} ; $\mu_{\mathbb{A}}(x) = 1$ 意味着 x 完全属于 \mathbb{A} ; $0 < \mu_{\mathbb{A}}(x) < 1$ 意味着 x 部分属于 \mathbb{A} .

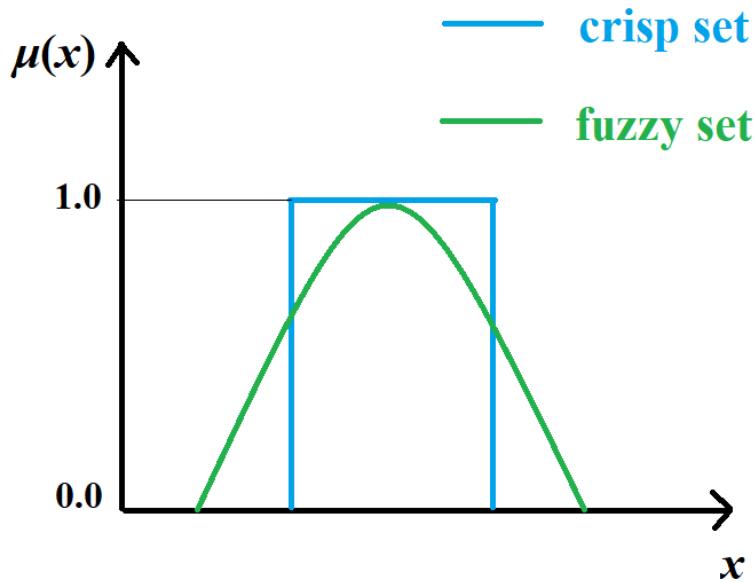


图 2.19: Membership Function of crisp set(blue) and fuzzy set (green).

Membership Function of crisp set and fuzzy set.

2.10.2.2.2 常见隶属函数

2.10.2.2.2.1 钟形曲线

钟形 (bell shape) 隶属函数表达式如下:

$$f(x) = \frac{1}{1 + | \frac{x-c}{a} |^{2b}}$$

其中, c 决定了钟形曲线的位置, a, b 决定了钟形曲线的形状.

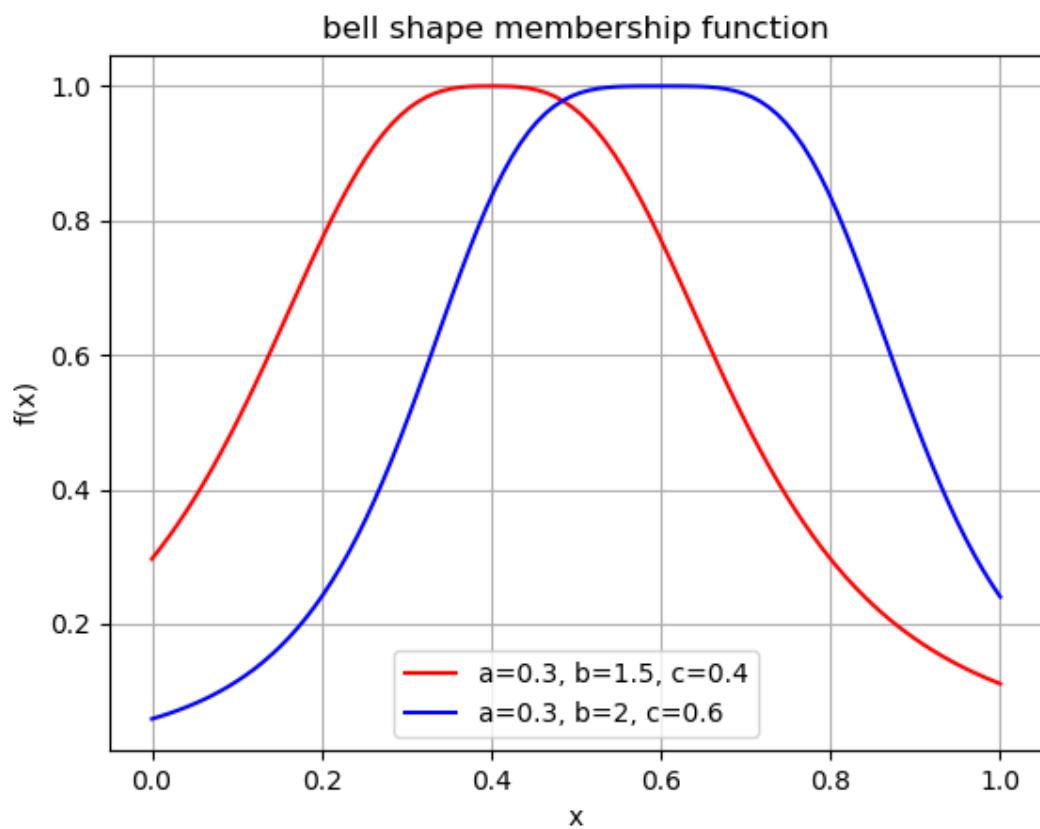


图 2.20: Bell Shape Membership Function.

Bell Shape Membership Function.

2.10.2.2.2 高斯函数

高斯函数 (Gauss Function) 也称高斯径向基函数 (Gauss Radial Basis function, RBF), 其表达式如下:

$$f(x) = \exp\left(\frac{-\|x - c\|^2}{a^2}\right)$$

其中, c 为中/均值, a 表示标准差.

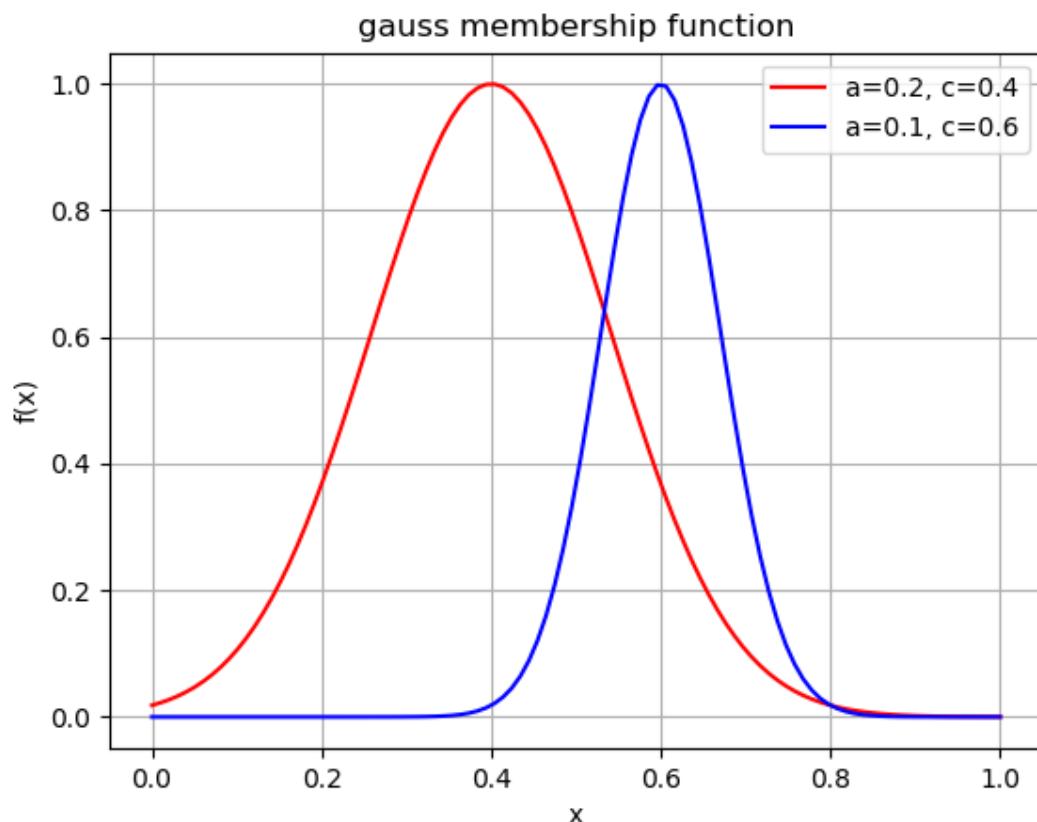


图 2.21: Gauss Membership Function.

Gauss Membership Function.

2.10.2.3 模糊逻辑

2.10.2.3.1 布尔逻辑与模糊逻辑

在经典二值逻辑(也称布尔逻辑)运算中, 变量只能取 0 或 1, 与经典的二值逻辑运算不同, 模糊逻辑(*Fuzzy Logic*)允许变量取 0, 1 之间的实数.

2.10.2.3.2 布尔逻辑与模糊逻辑算子

常用布尔逻辑与模糊逻辑算子定义如下, 更多参见[模糊算子](#)(页 125)

	布尔逻辑	模糊逻辑
	====	=====
与 AND(x, y) MIN(x, y)		
----- ----- -----		
或 OR(x, y) MAX(x, y)		
----- ----- -----		
非 NOT(x) 1 - x		

2.10.2.3.3 模糊规则

模糊规则(Fuzzy Rules)可用于对专家知识和常识进行建模. 一个模糊规则可以表示成三元组(A, B, R)

2.10.2.3.3.1 IF-THEN

IF-THEN 模糊规则(Fuzzy Rules)

- 前提(Premise): x is A
- 含义(Implication): IF x is A THEN y is B
- 结果(Consequent): y is B

2.10.2.3.4 模糊逻辑推理步骤

模糊逻辑推理步骤如下

1. 模糊化(Fuzzification):
2. 模糊运算(Fuzzy Logic Operator):
3. 模糊规则推理(Fuzzy Rule):
4. 去模糊化(De-fuzzyfication):

提示: 参考文献 [3] Chapter 13 Reasoning and Fuzzy Logic

- [模糊逻辑与模糊推理](<https://wenku.baidu.com/view/3039bbd73186bceb19e8bbf8.html>)
- [模糊逻辑与模糊推理](https://wenku.baidu.com/view/261aac1eaeaad1f347933fbf.html?rec_flag=default)
- [模糊推理算法及应用](<https://wenku.baidu.com/view/a89c6ac8a58da0116c1749c3.html>)

2.10.2.4 模糊子代数

模糊子代数 (*Fuzzy Subalgebra*)

2.10.2.5 模糊集合理论拓展

参见书籍 [2] Chapter 10

2.10.2.5.1 Lattice Valued Fuzzy Sets

Lattice Valued Fuzzy Sets (L-Fuzzy Sets)

2.10.2.5.2 Intuitionistic Fuzzy Sets

2.10.2.5.3 Interval Type II Fuzzy Sets

2.10.2.5.4 Fuzzy Sets of Type 2

2.10.3 模糊算术

2.10.3.1 模糊数

2.10.3.2 模糊算术

2.10.3.3 模糊分析

2.10.3.4 模糊差分方程

2.10.4 模糊关系

2.10.4.1 模糊关系

2.10.4.1.1 明确关系与模糊关系

Definition 45 (明确关系 (Crisp Relation)) 给定明确集合 \mathbb{X}, \mathbb{Y} , 直积 $\mathbb{X} \times \mathbb{Y} = \{(x, y) | x \in \mathbb{X}, y \in \mathbb{Y}\}$ 的一个子集 \mathbb{R} 称为集合 \mathbb{X} 到 \mathbb{Y} 的一个 **明确关系 (Crisp Relation)**. 这种关系可以用特征函数表示为

$$\mu_{\mathbb{R}}(x, y) : \mathbb{X} \times \mathbb{Y} \rightarrow \{0, 1\}$$

更近一步地, 可表示为:

$$\mu_{\mathbb{R}}(x, y) = \begin{cases} 1, & \text{if } (x, y) \in \mathbb{R} \\ 0, & \text{otherwise} \end{cases}$$

Definition 46 (模糊关系 (Fuzzy Relation)) 给定明确集合 \mathbb{X}, \mathbb{Y} , 直积 $\mathbb{X} \times \mathbb{Y} = \{(x, y) | x \in \mathbb{X}, y \in \mathbb{Y}\}$ 的一个模糊子集 \mathbb{R} 称为集合 \mathbb{X} 到 \mathbb{Y} 的一个 **模糊关系**. 用隶属函数可以表示为

$$\mu_{\mathbb{R}}(x, y) : \mathbb{X} \times \mathbb{Y} \rightarrow [0, 1],$$

反应了 \mathbb{X}, \mathbb{Y} 之间的关系程度.

2.10.4.1.2 模糊矩阵

设集合 \mathbb{X} 含有 m 个元素, 集合 \mathbb{Y} 含有 n 个元素, 模糊子集 \mathbb{R} 为集合 \mathbb{X} 到 \mathbb{Y} 的一个模糊关系, 则模糊关系可以用 $m \times n$ 的矩阵表示, 称为 **模糊关系矩阵或隶属矩阵**, 简称 **模糊矩阵** (Fuzzy Matrix).

$$\mu_{\mathbb{R}} = \begin{bmatrix} \mu_{\mathbb{R}}(x_1, y_1) & \mu_{\mathbb{R}}(x_1, y_2) & \cdots & \mu_{\mathbb{R}}(x_1, y_n) \\ \mu_{\mathbb{R}}(x_2, y_1) & \mu_{\mathbb{R}}(x_2, y_2) & \cdots & \mu_{\mathbb{R}}(x_2, y_n) \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{\mathbb{R}}(x_m, y_1) & \mu_{\mathbb{R}}(x_m, y_2) & \cdots & \mu_{\mathbb{R}}(x_m, y_n) \end{bmatrix}$$

2.10.4.1.3 模糊关系的性质

- 自反性: $\mu_{\mathbb{R}}(x, x) = 1$
- 对称性: $\mu_{\mathbb{R}}(x, y) = \mu_{\mathbb{R}}(y, x)$
- 传递性:

2.10.4.2 模糊关系合成

1. Min-Max Composition
2. Max-Min Composition
3. Min Composition

待补充, 参见书籍 [2] p34

2.10.5 模糊推理

2.10.5.1 语言变量

待补充, 参见书籍 [2] p79

2.10.5.2 模糊规则

待补充, 参见书籍 [2] p81

2.10.5.3 模糊推理

模糊输入模糊输出

待补充, 参见书籍 [2] p87

Mamdani Larsen Generalized modus ponens or T-norm-based Gödel Inference Gödel residual inference

2.10.6 模糊系统

2.10.6.1 单入单出模糊系统

单入单出模糊系统 (Single Input Single Output Fuzzy System)

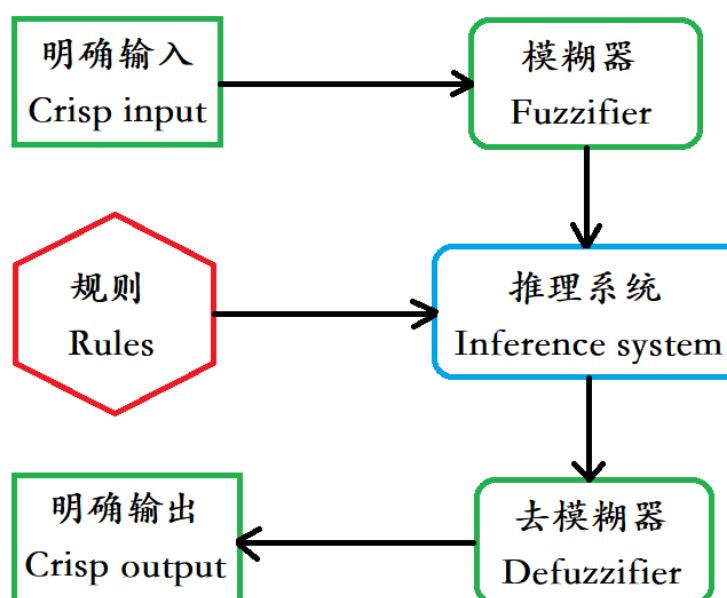


图 2.22: Single Input Single Output Fuzzy System

Single Input Single Output Fuzzy System

2.10.6.2 Takagi-Sugeno 模糊系统

2.10.6.2.1 Takagi-Sugeno 模糊系统

Takagi-Sugeno(TS) 模糊系统由 Takagi 等人于 1985 年提出 [5] , TS 是单入单出模糊系统 (页 133) , TS 系统的规则包括语言类型的先行词和分段线性的明确输出的结论, 这使得去模糊化步骤变得多余.

对于单个先行词 (antecedent), TS 系统规则举例如下:

注解:

- 假定 (premise): If x is A_i then $z_i = a_i x + b_i, i = 1, \dots, n$
 - 事实 (fact): x is x_0
 - 结论 (conclusion): z is z_0 .
-

单个先行词下的 TS 模糊系统控制算法:

提示:

1. 输入明确值 x_0
 2. 计算每个模糊规则的触发强度 (firing strengths) : $\alpha_i = A_i(x_0)$
 3. 计算规则输出: $z_i = a_i x_0 + c_i$
 4. 最终输出: $z_0 = \frac{\sum_{i=1}^n \alpha_i z_i}{\sum_{i=1}^n \alpha_i}$
-

对于多个先行词 (antecedent), TS 系统规则举例如下:

注解:

- 假定 (premise): If x is A_i and y is B_i then $z_i = a_i x + b_i y + c_i, i = 1, \dots, n$
 - 事实 (fact): x is x_0 and y is y_0
 - 结论 (conclusion): z is z_0 .
-

多个先行词下的 TS 模糊系统控制算法:

提示:

1. 输入明确值 x_0, y_0
 2. 计算每个模糊规则的触发强度 (firing strengths) : $\alpha_i = A_i(x_0) \wedge B_i(y_0)$
 3. 计算规则输出: $z_i = a_i x_0 + b_i y_0 + c_i$
 4. 最终输出: $z_0 = \frac{\sum_{i=1}^n \alpha_i z_i}{\sum_{i=1}^n \alpha_i}$
-

2.10.6.2.2 Takagi-Sugeno 模糊系统的近似特性

理论上可以近似任意连续可微函数 (differentiable function)

$$F(f, x) = \frac{\sum_{i=1}^n A_i(x) \cdot (a_i x + b_i)}{\sum_{i=1}^n A_i(x)}$$

待补充, 参见书籍 [2] p126

2.10.6.3 Takagi-Sugeno-Kang 模糊系统

如今比较常用的是 Takagi-Sugeno-Kang (TSK) 型模糊推理系统 (Fuzzy Inference System, FIS) [5][6]. Petr 等人于 2005 年提出在线学习式 TSK 模型 [1C.Petr.2005], Gu 等人于 2018 年提出基于贝叶斯的 TSK-FIS [8].

2.10.6.3.1 TSKFIS 原理

一个 TSK 模糊系统主要由 “IF-THEN” 模糊规则构成. 给定 d -维输入向量 $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$, 记第 k 个规则为 R^k , 在则 TSK 模糊系统中, R^k 可以表示为

IF $x_1(k)$ is A_{k1} , $x_2(k)$ is $A_{k2}, \dots, x_d(k)$ is A_{kd}

$$\text{THEN } f_k(\mathbf{x}) = q_{k0} + \sum_{j=1}^d q_{kj} x_j = \mathbf{q}_k^T \tilde{\mathbf{x}}$$

其中, $A_{ki} (i = 1, 2, \dots, d)$ 是一个模糊集, $k = 1, 2, \dots, K$, $\tilde{\mathbf{x}} = [1, \mathbf{x}_n^T]^T$. 且 $\mathbf{q}_k = [q_{k0}, q_{k1}, \dots, q_{kd}]^T$ 是第 k 个规则对应结论的参数向量, 为指定或可学习 (如自适应神经模糊推理系统中的权重).

若使用高斯核表示模糊集 A_{ki} 对应的隶属函数 $A_{ki}(x_i)$, 则

$$\mu_{A_{ki}}(x_i) = \exp\left(\frac{-\|x_i - c_{ki}\|^2}{\sigma_{ki}^2}\right)$$

其中, c_{ki}, σ_{ki} 分别对应高斯隶属函数中心 (均值) 和宽度 (标准差).

TSK 模糊系统的输出可以表示为

$$\hat{y} = \sum_{k=1}^K \frac{\mu_k(\mathbf{x})}{\sum_{k=1}^K \mu_k(\mathbf{x})} f_k(\mathbf{x}) = \sum_{k=1}^K \hat{\mu}_k(\mathbf{x}) f_k(\mathbf{x})$$

其中, $\mu_k(\mathbf{x})$ 和 $\hat{\mu}(\mathbf{x})$ 分别为第 k 个规则对应的模糊隶属函数和归一化模糊隶属函数:

$$\mu_k(\mathbf{x}) = \prod_{i=1}^d \mu_{A_{ki}}(x_i), \quad \hat{\mu}_k(\mathbf{x}) = \frac{\mu_k(\mathbf{x})}{\sum_{k=1}^K \mu_k(\mathbf{x})}$$

2.10.7 模糊变换

2.10.8 参考文献

2.10.9 名词术语

Crisp Set 明确集合 ([Crisp Set](http://en.volupedia.org/wiki/Crisp_set) (http://en.volupedia.org/wiki/Crisp_set)) 指经典集合, 主要强调元素是否属于集合是确定的.

Fuzzy Logic 模糊逻辑 ([Fuzzy Logic](http://en.volupedia.org/wiki/Fuzzy_logic) (http://en.volupedia.org/wiki/Fuzzy_logic))

Fuzzy Mathematics 模糊数学 ([Fuzzy Mathematics](http://en.volupedia.org/wiki/Fuzzy_mathematics) (http://en.volupedia.org/wiki/Fuzzy_mathematics))

Fuzzy Set 模糊集合 ([Fuzzy Set](http://en.volupedia.org/wiki/Fuzzy_set) (http://en.volupedia.org/wiki/Fuzzy_set)) 与经典集合不同, 主要强调元素是否属于集合是不确定的, 用隶属函数与隶属度来表示.

Fuzzy Subalgebra 模糊子代数 ([Fuzzy Subalgebra](http://en.volupedia.org/wiki/Fuzzy_subalgebra) (http://en.volupedia.org/wiki/Fuzzy_subalgebra))

Fuzzy System 模糊系统 ([Fuzzy System](http://en.volupedia.org/wiki/Fuzzy_system) (http://en.volupedia.org/wiki/Fuzzy_system))

Indicator Function 示性函数 ([Indicator Function](http://en.volupedia.org/wiki/Indicator_function) (http://en.volupedia.org/wiki/Indicator_function)), 参见集合的特征函数 (页 12)

Membership Degree 隶属度 ([Membership Degree](http://en.volupedia.org/wiki/Membership_degree) (http://en.volupedia.org/wiki/Membership_degree)), 参见隶属函数 (页 126)

Membership Function 隶属函数 ([Membership Function](http://en.volupedia.org/wiki/Membership_function_(mathematics)) ([http://en.volupedia.org/wiki/Membership_function_\(mathematics\)](http://en.volupedia.org/wiki/Membership_function_(mathematics)))), 参见隶属函数 (页 126)

Membership Grade 隶属度 ([Membership Grade](http://en.volupedia.org/wiki/Membership_grade) (http://en.volupedia.org/wiki/Membership_grade)), 参见隶属函数 (页 126)

Neuro Fuzzy 神经模糊 ([Neuro Fuzzy](http://en.volupedia.org/wiki/Neuro-fuzzy) (<http://en.volupedia.org/wiki/Neuro-fuzzy>))

Rough Set 粗糙集合 ([Rough Set](http://en.volupedia.org/wiki/Rough_set) (http://en.volupedia.org/wiki/Rough_set)) 与经典集合不同, 主要强调元素是否属于集合是不确定的, 用两个上下隶属函数与隶属度来表示.

2.11 优化理论

2.11.1 优化问题概述

2.11.1.1 优化中的对偶问题

优化中的对偶问题 ([Dual Problem](#))

Fenchel's duality theorem

Wolfe dual problem

2.11.1.1.1 约束优化的对偶问题

设有原优化问题

$$P : \max f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$
$$\text{s.t. } \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{cases} \quad (2.18)$$

其对偶优化问题为

$$D : \min g(\mathbf{y}) = \mathbf{b}^T \mathbf{y}$$
$$\text{s.t. } \begin{cases} \mathbf{A}^T \mathbf{y} \geq \mathbf{c}^T \\ \mathbf{y} \geq \mathbf{0} \end{cases} \quad (2.19)$$

上述问题 P 和 D 称为一对对偶优化问题.

2.11.2 无约束优化方法

2.11.3 约束优化方法

2.11.3.1 约束优化方法概述

2.11.3.1.1 一般方法

将约束优化问题转换为无约束优化

2.11.3.2 对偶上升算法

2.11.3.2.1 概念与内涵

对偶上升法 (Dual Ascent) [1] p5

考虑如下等式约束凸优化问题

$$\begin{aligned} P_0 : \quad & \min_x \quad f(x) \\ \text{s.t.} \quad & Ax = b \end{aligned} \tag{2.20}$$

其中, $x \in \mathbb{R}^{N \times 1}$, $A \in \mathbb{R}^{M \times N}$, $f : \mathbb{R}^{N \times 1} \rightarrow \mathbb{R}$ 为凸函数. 问题 P_0 (式.2.20) 的拉格朗日函数形式为

$$L(x, \lambda) = f(x) + \lambda^T(Ax - b)$$

其对偶函数为

$$D : \max g(x) = \inf_x L(x, \lambda) = -f^*(-A^T\lambda) - b^T\lambda,$$

其中, λ 为对偶变量也是拉格朗日乘子变量, f^* 是函数 f 的凸共轭.

对偶上升法迭代更新过程为

$$\begin{aligned} x^{k+1} &:= \underset{x}{\operatorname{argmin}} L(x, \lambda^k) \\ \lambda^{k+1} &:= \lambda^k + \mu^k (Ax^{k+1} - b) \end{aligned}$$

其中, k 为迭代次数, μ^k 为迭代步长. 选择合适的 μ^k , 对偶函数每次迭代后会上升 ($g(x)$), 因而称为对偶上升法.

2.11.3.2.2 实验与分析

2.11.3.3 KKT 条件

Karush–Kuhn–Tucker (KKT) 条件给出了一类约束优化问题的解的约束, 通过这些约束可以求解优化问题.

2.11.3.3.1 概念与内涵

考虑如下非线性最小或最大化问题

$$P_0 : \begin{aligned} & \min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ & \text{s.t. } \begin{cases} g_i(\boldsymbol{x}) = 0, i = 1, 2, \dots, M_g \\ h_j(\boldsymbol{x}) \leq 0, j = 1, 2, \dots, M_h \end{cases} \end{aligned}$$

其中, $\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ 为待求变量. $\boldsymbol{\alpha}, \boldsymbol{\beta}$ 为由 KKT 乘子组成的向量, 上式转化为拉格朗日函数

$$P_1 : L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{x}) + \sum_{i=1}^{M_g} \alpha_i g_i(\boldsymbol{x}) + \sum_{j=1}^{M_h} \beta_j h_j(\boldsymbol{x}) \quad (2.21)$$

则其解满足如下 KKT 条件

$$\begin{aligned} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) &= 0 \\ \beta_j^* h_j(\boldsymbol{x}^*) &= 0, j = 1, 2, \dots, M_h \\ g_i(\boldsymbol{x}^*) &= 0, i = 1, 2, \dots, M_g \\ h_j(\boldsymbol{x}^*) &\leq 0, j = 1, 2, \dots, M_h \\ \beta_j^* &\geq 0, j = 1, 2, \dots, M_h \end{aligned} \quad (2.22)$$

其中, $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \dots, \alpha_{M_g}^*]^T$, $\boldsymbol{\beta}^* = [\beta_1^*, \beta_2^*, \dots, \beta_{M_h}^*]^T$, $\boldsymbol{x}^* = [x_1^*, x_2^*, \dots, x_N^*]^T$.

KKT 条件是使一组解成为最优解的必要条件, 当原问题是凸优化问题时, KKT 条件也是充分条件.

提示: 当 $M_h = 0$ 即不含不等式约束时, 一般称为拉格朗日乘子法.

2.11.3.4 拉格朗日乘子法

2.11.3.4.1 概念与内涵

在优化理论中, **拉格朗日乘子** (*Lagrange Multiplier*) 法是用于求解等式约束优化问题的局部极小极大问题. 其基本思想是将约束优化问题转化为无约束优化问题. 函数的不动点/稳定点 (stationary point) 也满足等式约束, 即函数在该不动点的梯度可以表示成约束在该点的梯度的线性组合, 从而原始问题可转化为拉格朗日函数形式 (Lagrangian Function).

Definition 47 设有等式约束优化问题:

$$P_0 : \min_{\boldsymbol{x}} f(\boldsymbol{x}) \text{ s.t. } g_i(\boldsymbol{x}) = 0, \quad (2.23)$$

其中, $g_i, i = 1, 2, \dots, M$ 为 M 个等式约束. 化为无约束优化的拉格朗日函数形式为

$$P_1 : \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^M \lambda_i g_i(\boldsymbol{x}), \quad (2.24)$$

其中, $\lambda_i, i = 1, 2, \dots, M$ 为 **拉格朗日乘子** (*Lagrange Multiplier*), 等号右边两项可以为加也可以为减. 其矩阵形式为

$$P_1 : \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{g}, \quad (2.25)$$

其中, $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_M]^T$ 为 M 个拉格朗日乘子构成的向量, $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$ 为 N 个变量构成的向量, $\mathbf{g} = [g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_M(\mathbf{x})]^T$ 为 M 个约束构成的向量.

易知式2.25 对 λ 求导为

$$\nabla_{\lambda} L(\mathbf{x}, \lambda) = \mathbf{g}$$

式2.25 对 \mathbf{x} 求导为

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = \nabla_{\mathbf{x}} f(\mathbf{x}) - \sum_{i=1}^M \lambda_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x}) - \lambda^T \nabla_{\mathbf{x}} \mathbf{g}$$

从而有

$$\nabla_{\mathbf{x}, \lambda} L(\mathbf{x}, \lambda) = 0 \iff \begin{cases} \nabla f(\mathbf{x}) = \lambda^T \nabla_{\mathbf{x}} \mathbf{g} = \sum_{i=1}^M \lambda_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) \\ g_1(\mathbf{x}) = \dots = g_M(\mathbf{x}) = 0 \end{cases} \quad (2.26)$$

2.11.3.4.2 拉格朗日函数的对偶问题

2.11.3.4.3 约束优化与流形学习

寻找满足等式/不等式约束的极小极大问题, 可推广为寻找不同流形 \mathcal{M} 上的最小值与最大值. \mathcal{M} 不一定是欧式空间, 也可以是黎曼空间.

[Modern formulation via differentiable manifolds](http://en.volupedia.org/wiki/Lagrange_multiplier) (http://en.volupedia.org/wiki/Lagrange_multiplier)

2.11.3.4.4 实验与分析

2.11.3.5 增广拉格朗日法

增广拉格朗日法 (*Augmented Lagrangian Method*, ALM)

2.11.3.5.1 概念与内涵

设有等式约束优化问题:

$$P_0 : \min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } g_i(\mathbf{x}) = 0, \quad (2.27)$$

其中, $g_i, i = 1, 2, \dots, M$ 为 M 个等式约束. 化为无约束优化的增广拉格朗日函数形式为

$$P_1 : \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \sum_{i=1}^M \lambda_i g_i(\mathbf{x}) + \frac{\mu}{2} \sum_{i=1}^M g_i(\mathbf{x})^2, \quad (2.28)$$

其中, $i = 1, 2, \dots, M$, λ_i 为 **拉格朗日乘子** (Lagrange Multiplier), μ 为增广拉格朗日乘子 (Augmented Lagrange Multiplier). 其矩阵形式为

$$P_1 : \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \lambda^T \mathbf{g} + \frac{\mu}{2} \|\mathbf{g}\|_2^2, \quad (2.29)$$

其中, $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_M]^T$ 为 M 个拉格朗日乘子构成的向量, $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$ 为 N 个变量构成的向量, $\mathbf{g} = [g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_M(\mathbf{x})]^T$ 为 M 个约束构成的向量. $\|\cdot\|_2^2$ 表示向量的 ℓ_2 范数.

即增广拉格朗日在拉格朗日的基础上加上了一个光滑惩罚项 $\|g\|_2^2$.

与对偶上升法类似 (对偶上升算法 (页 137)), 其迭代迭代更新规则为

$$\begin{aligned}\boldsymbol{x}^{k+1} &:= \underset{\boldsymbol{x}}{\operatorname{argmin}} L(\boldsymbol{x}, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &:= \boldsymbol{\lambda}^k - \mu_k \boldsymbol{g}(\boldsymbol{x}^{k+1})\end{aligned}$$

2.11.3.5.2 实验与分析

2.11.3.6 迭代收缩阈值类算法

2.11.3.6.1 概念与内涵

迭代收缩阈值算法 (*Iterative Shrinkage Thresholding Algorithm*, ISTA) [4]

快速迭代收缩阈值算法 (*Fast Iterative Shrinkage Thresholding Algorithm*, FISTA)

FISTA (A fast iterative shrinkage-thresholding algorithm) 是一种快速的迭代收缩阈值算法 (ISTA). FISTA 和 ISTA 都是基于梯度下降的思想, 在迭代过程中进行了更为聪明 (smarter) 的选择, 从而达到更快的迭代速度。理论证明: FISTA 和 ISTA 的迭代收敛速度分别为 $O(1/k^2)$ 和 $O(1/k)$.

2.11.3.6.2 ISTA

2.11.3.6.3 FISTA

2.11.3.6.4 实例分析

2.11.3.6.4.1 图像去模糊与去噪

2.11.3.7 交替方向乘子法

交替方向乘子法 (*Alternating Direction Method of Multipliers*, ADMM) [1] 结合了对偶分解 (dual decomposition) 和增广拉格朗日方法的优点, 最早可追溯到 20 世纪 70 年代中期 [2][3].

主页¹ 中提供了所有相关论文, 代码及应用文档.

该问题与如下约束优化问题等价

$$P_1 : \min_{\boldsymbol{x}, \boldsymbol{y}} f(\boldsymbol{x}) + g(\boldsymbol{y}) \text{ s.t. } \boldsymbol{x} = \boldsymbol{y}.$$

尽管这种变化可能看起来微不足道, 但现在可以使用约束优化方法来解决这个问题, 如增广拉格朗日方法. 此时目标函数被 \boldsymbol{x} 与 \boldsymbol{y} 分离, 对偶更新问题在于求解一个同时满足 $\boldsymbol{x}, \boldsymbol{y}$ 的近似函数. ADMM 算法通过交替迭代的方法求解该问题, 即先固定变量 \boldsymbol{y} 优化另一个变量 \boldsymbol{x} , 再固定变量 \boldsymbol{x} 优化变量 \boldsymbol{y} , 如此交替优化求解. 虽然这并非准确地最优化求解, 但该方法最终可以收敛到很好的解.

¹ <https://web.stanford.edu/~boyd/admm.html>

2.11.3.7.1 ADMM 原理

ADMM 用于求解如下形式的约束优化问题

$$\begin{aligned} \min & \quad f(\mathbf{x}) + g(\mathbf{y}) \\ \text{s.t.} & \quad \mathbf{Ax} + \mathbf{By} = \mathbf{c} \end{aligned} \quad (2.30)$$

其中, f, g 为凸函数, $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^M$, $\mathbf{A} \in \mathbb{R}^{p \times N}$, $\mathbf{B} \in \mathbb{R}^{p \times M}$, $\mathbf{c} \in \mathbb{R}^p$.

与乘子法类似, 式.2.30 的增广拉格朗日形式为

$$L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^T (\mathbf{Ax} + \mathbf{By} - \mathbf{c}) + \frac{\mu}{2} \|\mathbf{Ax} + \mathbf{By} - \mathbf{c}\|_2^2, \quad (2.31)$$

其中, μ 为惩罚因子.

ADMM 算法的参数迭代更新规则为

$$\begin{aligned} \mathbf{x}^{k+1} &:= \underset{\mathbf{x}}{\operatorname{argmin}} L_\mu(\mathbf{x}, \mathbf{y}^k, \boldsymbol{\lambda}^k) \\ \mathbf{y}^{k+1} &:= \underset{\mathbf{y}}{\operatorname{argmin}} L_\mu(\mathbf{x}^{k+1}, \mathbf{y}, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &:= \boldsymbol{\lambda}^k + \mu (\mathbf{Ax}^{k+1} + \mathbf{By}^{k+1} - \mathbf{c}) \end{aligned}$$

式.2.30 增广拉格朗日乘子法参数更新规则为

$$\begin{aligned} (\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) &:= \underset{\mathbf{x}, \mathbf{y}}{\operatorname{argmin}} L_\mu(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &:= \boldsymbol{\lambda}^k + \mu (\mathbf{Ax}^{k+1} + \mathbf{By}^{k+1} - \mathbf{c}) \end{aligned} \quad (2.32)$$

可以看出, 增广拉格朗日乘子法需要同时优化两个原始变量 \mathbf{x}, \mathbf{y} . 与增广拉格朗日乘子法不同的是, 在 ADMM 中, \mathbf{x} 与 \mathbf{y} 采用交替迭代 (alternating direction) 更新的方法.

2.11.3.7.2 Scale 版 ADMM 原理

通过组合增广拉格朗日函数中的线性项与二次项并缩放对偶变量.

$$\begin{aligned} \boldsymbol{\lambda}^T \mathbf{r} + \frac{\mu}{2} \|\mathbf{r}\|_2^2 &= \frac{\mu}{2} \|\mathbf{r} + \frac{1}{\mu} \boldsymbol{\lambda}\|_2^2 - \frac{1}{2\mu} \|\boldsymbol{\lambda}\|_2^2 \\ &= \frac{\mu}{2} \|\mathbf{r} + \mathbf{u}\|_2^2 - \frac{\mu}{2} \|\mathbf{u}\|_2^2 \end{aligned}$$

其中, $\mathbf{r} = \mathbf{Ax} + \mathbf{By} - \mathbf{c}$ 为残差, $\mathbf{u} = \frac{1}{\mu} \boldsymbol{\lambda}$ 为 scale 版对偶变量, 对应的 scale 版更新规则为

$$\begin{aligned} \mathbf{x}^{k+1} &:= \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) + (\mu/2) \|\mathbf{Ax} + \mathbf{By}^k - \mathbf{c} + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{y}^{k+1} &:= \underset{\mathbf{y}}{\operatorname{argmin}} \left(g(\mathbf{y}) + (\mu/2) \|\mathbf{Ax}^{k+1} + \mathbf{By} - \mathbf{c} + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{u}^{k+1} &:= \mathbf{u}^k + \mathbf{Ax}^{k+1} + \mathbf{By}^{k+1} - \mathbf{c} \end{aligned}$$

记第 k 次的残差为 $\mathbf{r}^k = \mathbf{Ax}^k + \mathbf{By}^k - \mathbf{c}$, 则有 scale 版对偶变量更新规则

$$\mathbf{u}^k = \mathbf{u}^0 + \sum_{j=1}^k \mathbf{r}^j.$$

2.11.3.7.3 收敛性分析

2.11.3.7.4 最优条件与停止准则

2.11.3.7.5 实验与分析

<https://blog.csdn.net/pizibing880909/article/details/21192243>

2.11.3.8 近端算法

<http://proximity-operator.net/tutorial.html>

2.11.3.8.1 什么是近端算法

2.11.3.8.1.1 近端算子

http://en.volupedia.org/wiki/Proximal_operator

2.11.3.8.1.2 近端梯度

http://en.volupedia.org/wiki/Proximal_gradient_method

2.11.3.9 分裂增广拉格朗日收缩算法

2.11.3.9.1 变量分裂

2.11.3.9.2 SALSA 与 C-SALSA

变量 分裂增广拉格朗日收缩算法 (*Split Augmented Lagrangian Shrinkage Algorithm* , SALSA) [5][6]

变量 约束分裂增广拉格朗日收缩算法 (Constrained Split Augmented Lagrangian Shrinkage Algorithm , C-SALSA) [7][8] 是 SALSA 的约束优化版本.

2.11.4 微分方法

2.11.4.1 简介

2.11.4.2 数值微分法

数值微分 (*Numerical Differentiation*)

2.11.4.3 符号微分法

符号微分 (*Symbolic Differentiation*)

2.11.4.4 自动微分法

自动微分 (*Automatic Differentiation*)

2.11.5 参考文献

2.11.6 名词术语

Alternating Direction Method of Multipliers 交替方向乘子法 ([Alternating Direction Method of Multipliers](http://en.volupedia.org/wiki/Alternating_direction_method_of_multipliers#Alternating_direction_method_of_multipliers))

(http://en.volupedia.org/wiki/Alternating_direction_method_of_multipliers#Alternating_direction_method_of_multipliers), ADMM)

Augmented Lagrangian Method 增广拉格朗日法 ([Augmented Lagrangian Method](http://en.volupedia.org/wiki/Augmented_Lagrangian_method))

(http://en.volupedia.org/wiki/Augmented_Lagrangian_method), ALM)

Dual Problem 对偶问题 ([Dual Problem](http://en.volupedia.org/wiki/Dual_problem) (http://en.volupedia.org/wiki/Dual_problem))，也称对偶性 Duality)

Fast Iterative Shrinkage Thresholding Algorithm 快速迭代阈值收缩算法 ([Fast Iterative Shrinkage Thresholding Algorithm](http://en.volupedia.org/wiki/Augmented_Lagrangian_method) (http://en.volupedia.org/wiki/Augmented_Lagrangian_method), FISTA)

Iterative Shrinkage Thresholding Algorithm 迭代阈值收缩算法 ([Iterative Shrinkage Thresholding Algorithm](http://en.volupedia.org/wiki/Augmented_Lagrangian_method))

(http://en.volupedia.org/wiki/Augmented_Lagrangian_method), ISTA)

Karush–Kuhn–Tucker KKT 条件 ([Karush–Kuhn–Tucker](http://en.volupedia.org/wiki/Karush–Kuhn–Tucker) (<http://en.volupedia.org/wiki/Karush–Kuhn–Tucker>))

Lagrange Multiplier 拉格朗日乘子 ([Lagrange Multiplier](http://en.volupedia.org/wiki/Lagrange_multiplier) (http://en.volupedia.org/wiki/Lagrange_multiplier), ALM)

Split Augmented Lagrangian Shrinkage Algorithm 分裂增广拉格朗日收缩算法 ([Split Augmented Lagrangian Shrinkage Algorithm](http://en.volupedia.org/wiki/Augmented_Lagrangian_Shrinkage_Algorithm) (http://en.volupedia.org/wiki/Augmented_Lagrangian_method), SALSA)

2.12 名词术语

Cholesky decomposition Cholesky 分解 ([Cholesky decomposition](http://en.volupedia.org/wiki/Cholesky_decomposition) (http://en.volupedia.org/wiki/Cholesky_decomposition))

三角分解的一种, 把一个对称正定的矩阵表示成一个下三角矩阵 \mathbf{L} 和其转置 \mathbf{L}^T 的乘积的分解. 它要求矩阵的所有特征值必须大于零, 故分解的下三角的对角元也是大于零的. Cholesky 分解法又称平方根法, 是当 \mathbf{A} 为实对称正定矩阵时, LU 三角分解法的变形.

Conjugate transpose 共轭转置 ([Conjugate transpose](http://en.volupedia.org/wiki/Conjugate_transpose) (http://en.volupedia.org/wiki/Conjugate_transpose)), 也叫 Hermitian 转置 (Hermitian transpose), 是对矩阵中的元素取共轭复数后再转置的操作.

Diagonal Matrix 对角矩阵 ([Diagonal Matrix](http://en.volupedia.org/wiki/Diagonal_matrix) (http://en.volupedia.org/wiki/Diagonal_matrix)) 主对角线以外的所有元素都为零的矩阵称为对角矩阵.

Doolittle decomposition Doolittle 分解 ([Doolittle Decomposition](http://en.volupedia.org/wiki/LU_decomposition) (http://en.volupedia.org/wiki/LU_decomposition)) 三角分解的一种, 把一个对称正定的矩阵表示成一个下三角矩阵 \mathbf{L} 和其转置 \mathbf{L}^T 的乘积的分解. 它要求矩阵的所有特征值必须大于零, 故分解的下三角的对角元也是大于零的. Cholesky 分解法又称平方根法, 是当 \mathbf{A} 为实对称正定矩阵时, LU 三角分解法的变形.

Elementary transformation 初等变换 (Elementary transformation) 线性代数的基本概念之一, 包含初等行变换与初等列变换. 以矩阵的初等变换为例, 具体包含三种操作: 1) 某一行 (列) 乘以一个数; 2) 某一行 (列) 乘以一个数的结果加到另一行 (列); 3) 互换两行 (列).

Full Rank Decomposition 满秩分解 (Full Rank Decomposition) 是指将一矩阵分解为行满秩 \mathbf{F} 与列满秩 \mathbf{G} 的两个矩阵的乘积的分解.

Generalized eigenvalue problem 广义特征值问题 (Generalized eigenvalue problem (http://en.volupedia.org/wiki/Eigendecomposition_of_a_matrix#Generalized_eigenvalue_problem))

Givens transformation Givens 变换 (Givens transformation (http://en.volupedia.org/wiki/Givens_rotation)) 在线性代数中, Givens 变换, 也称 Givens 旋转 (Givens rotation), 是描述将某一平面内的向量进行旋转的线性变换. 由 Wallace Givens 于 1950 年提出.

Gram–Schmidt process Gram–Schmidt 过程 (Gram–Schmidt process (http://en.volupedia.org/wiki/Gram%E2%80%93Schmidt_process)) 是指实内积空间的等度量线性变换, 保持内积不变, 对应于复内积空间的酉变换.

Hermitian matrix Hermitian 矩阵 ([Hermitian matrix](http://en.volupedia.org/wiki/Hermitian_matrix) (http://en.volupedia.org/wiki/Hermitian_matrix))

Householder transformation Householder 变换 (Householder transformation (http://en.volupedia.org/wiki/Householder_transformation)) 在线性代数中, Householder 变换, 也称为 Householder 反射 (Householder reflection), 是描述包含原点的平面或超平面的反射的线性变换. 由 Alston Scott Householder 于 1958 年提出.

Inner product space 内积空间 (Inner product space (http://en.volupedia.org/wiki/Inner_product_space)) 在线性代数中, 内积空间是具有称为内积的附加结构的向量空间. 这个附加结构将空间中的每对向量与称为向量内积的标量量相关联. 内积允许严格引入直观的几何概念, 如向量的长度或两个向量之间的角度. 它们还提供了定义向量之间的正交性(零内积)的方法. 内积空间将欧氏空间(其中内积是点积, 也称为标量积)推广到任意(可能是无限)维的向量空间, 并在泛函分析中加以研究. 带内积的向量空间概念的首次使用是由于 Peano 在 1898 年提出的.

Linear Combination 线性组合 (Linear Combination) 线性代数的基本概念之一, 它表示一些项与其相应系数相乘后的累加和, 如 $z = ax + by$.

Linear Representation 线性表示 (Linear Representation)

Linearly Dependent 线性相关 (Linear Correlation, Linearly Dependent) 线性代数的基本概念之一,

Linearly Independent 线性无关 (Linear Independence, Linearly Independent) 线性代数的基本概念之一,

Lower Triangular Matrix 下三角矩阵 (Triangular Matrix (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上方的所有元素都为零的矩阵.

Non-Homogeneous 非齐次 (Non-Homogeneous), 常数项不为零.

Normal Matrix 正规矩阵 ([Normal Matrix](http://en.volupedia.org/wiki/Normal_matrix) (http://en.volupedia.org/wiki/Normal_matrix))

Orthogonal Matrix 正交矩阵 ([Orthogonal Matrix](http://en.volupedia.org/wiki/Orthogonal_matrix) (http://en.volupedia.org/wiki/Orthogonal_matrix))

Orthogonal transformation 正交变换 ([Orthogonal transformation](http://en.volupedia.org/wiki/Orthogonal_transformation) (http://en.volupedia.org/wiki/Orthogonal_transformation)) 是指实内积空间的等度量线性变换, 保持内积不变, 对应于复内积空间的酉变换.

Orthogonal Triangular Decomposition 正交三角分解 ([Orthogonal Triangular Decomposition](http://en.volupedia.org/wiki/QR_decomposition) (http://en.volupedia.org/wiki/QR_decomposition)) 分解是把一个矩阵分解为一个正交矩阵 \mathbf{Q} 与一个上三角矩阵 \mathbf{R} 乘积的分解.

Permutation matrix 置换矩阵 ([Permutation matrix](http://en.volupedia.org/wiki/Permutation_matrix) (http://en.volupedia.org/wiki/Permutation_matrix)) 指每行或每列只有 1 个元素为 1, 其余均为 0 的二值矩阵, 这种矩阵具有置换矩阵两行或两列的功能.

Positive-definite matrix 正定矩阵 ([Positive-definite matrix](http://en.volupedia.org/wiki/Positive-definite_matrix) (http://en.volupedia.org/wiki/Positive-definite_matrix))

Series 级数 ([Series \(mathematics\)](http://en.volupedia.org/wiki/Series_(mathematics)) ([http://en.volupedia.org/wiki/Series_\(mathematics\)](http://en.volupedia.org/wiki/Series_(mathematics)))),

Singular Value Decomposition 奇异值分解 ([Singular value decomposition](http://en.volupedia.org/wiki/Singular_value_decomposition) (http://en.volupedia.org/wiki/Singular_value_decomposition)) 分解是把一个矩阵分解为一个酉矩阵 (unitary matrix) \mathbf{U} , 矩形对角矩阵 $\mathbf{\Sigma}$ 与一个酉矩阵 \mathbf{V} 乘积的分解.

Skew-Hermitian matrix 反 Hermitian 矩阵 ([Skew-Hermitian matrix](http://en.volupedia.org/wiki/Skew-Hermitian_matrix) (http://en.volupedia.org/wiki/Skew-Hermitian_matrix))

Skew-symmetric matrix 反对称矩阵 ([Skew-symmetric matrix](http://en.volupedia.org/wiki/Skew-symmetric_matrix) (http://en.volupedia.org/wiki/Skew-symmetric_matrix))

Spectral decomposition 谱分解 ([Spectral decomposition](http://en.volupedia.org/wiki/Spectral_theorem) (http://en.volupedia.org/wiki/Spectral_theorem)) 分解是把一个矩阵分解为一个酉矩阵 (unitary matrix) \mathbf{U} , 矩形对角矩阵 $\mathbf{\Sigma}$ 与一个酉矩阵 \mathbf{V} 乘积的分解.

Symmetric matrix 对称矩阵 ([Symmetric matrix](http://en.volupedia.org/wiki/Symmetric_matrix) (http://en.volupedia.org/wiki/Symmetric_matrix))

Symmetry transformation 对称变换 ([Symmetry transformation](http://en.volupedia.org/wiki/Symmetry) (<http://en.volupedia.org/wiki/Symmetry>)) 是指实内积空间的一种具有对称性的线性变换, 对应于复内积空间的酉对称变换.

System of linear equations 线性方程组

Triangular Decomposition 三角分解 ([Triangular Decomposition](http://en.volupedia.org/wiki/Triangular_decomposition) (http://en.volupedia.org/wiki/Triangular_decomposition)) 分解是把一个矩阵分解为一个下三角矩阵 \mathbf{L} 与一个上三角矩阵 \mathbf{R} 乘积的分解, 也可分解为一个下三角矩阵 \mathbf{L} 与一个对角阵 \mathbf{D} 及一个上三角矩阵 \mathbf{R} 乘积.

Triangular Matrix 三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 在线性代数中, 三角矩阵是一种特殊的方阵. 如果主对角线上方的所有元素都为零, 则称矩阵为下三角矩阵. 类似地, 如果主对角线下方的所有元素都为零, 则方形矩阵称为上三角矩阵.

Unit Lower Triangular Matrix 单位下三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上所有元素均为 1 的下三角矩阵.

Unit Triangular Matrix 单位三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 在线性代数中, 单位三角矩阵是指主对角线上的元素均为 1 的三角矩阵.

Unit Upper Triangular Matrix 单位上三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线上所有元素均为 1 的上三角矩阵.

Unitary Symmetry transformation 酉对称变换 ([Unitary Symmetry transformation](http://en.volupedia.org/wiki/Unitary_transformation) (http://en.volupedia.org/wiki/Unitary_transformation)) 是指酉空间的一种具有对称性的线性变换, 对应于实内积空间的对称变换.

Unitary transformation 酉变换 ([Unitary transformation](http://en.volupedia.org/wiki/Unitary_transformation) (http://en.volupedia.org/wiki/Unitary_transformation)) 是指复内积空间的等度量线性变换, 保持内积不变, 对应于实内积空间的正交变换.

Upper Triangular Matrix 上三角矩阵 ([Triangular Matrix](http://en.volupedia.org/wiki/Triangular_matrix) (http://en.volupedia.org/wiki/Triangular_matrix)) 指主对角线下方的所有元素都为零的矩阵.

CHAPTER 3

第二卷物理学基础

3.1 引言

物理学基础

- Physics Course with Symbolic Math Toolbox and Live Editor (<https://ww2.mathworks.cn/matlabcentral/fileexchange/68799-physics-course-with-symbolic-math-toolbox-and-live-editor>)
- Basic Physics Course (https://uni-tuebingen.de/fileadmin/Uni_Tuebingen/Fakultaeten/MathePhysik/Institute/ITP/Braeuer/Skript)

3.2 经典力学

3.2.1 牛顿力学

3.2.2 天体力学

3.2.3 天体动力学

3.2.4 固体力学

3.2.5 流体力学

3.2.6 工程力学

3.2.7 生物力学

3.2.8 纳米力学

3.3 电磁学

3.3.1 电与磁

3.3.1.1 电与电场

3.3.1.1.1 什么是电

3.3.1.2 磁与磁场

3.3.1.2.1 什么是磁

3.3.1.2.2 自旋磁矩与磁力矩

Definition 48 (磁矩自旋) 磁矩 (Magnetic Moment) 是磁铁物质的物理性质, 决定了其处于外磁场时的转矩. 载流回路、电子、分子或行星等都有磁矩.

自旋 (Spin) 是粒子所具有的内禀性质 (如质量、电量), 为粒子与生俱来的一种角动量, 为量子化的且大小不可变, 自旋可以产生磁矩. 自旋为 0 的粒子, 从各方向看都一样; 自旋为 1, 2 的粒子分别旋转 360 度 (如手)、180 度后看起来一样; 自旋为 $\frac{1}{2}$ 的粒子, 旋转 720 度后看起来一样.

- 磁矩 (μ) 描述载流线圈或微观粒子磁性的物理量, 与外磁场无关;
- 磁力矩 ($M = \mu \times B$ $M = \mu \times B$) 是载流线圈或微观粒子在外磁场中受到的力矩, 与外磁场有关.
- 力矩做功: $W = \int_{\theta_0}^{\theta} M d\theta = -\mu B \cos \theta + \mu B \cos \theta_0 = -\mu \cdot B + C$

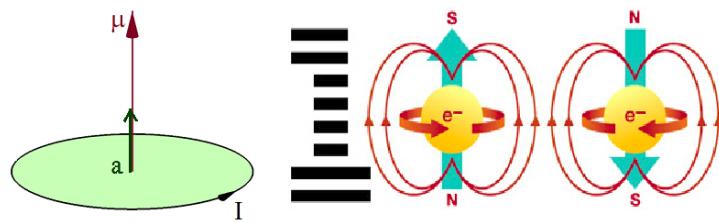


图 3.1: Spin (left), Loop Current Magnetic Moment (middle right)
Spin (left), Loop Current Magnetic Moment (middle right)

- 磁矩在外磁场中具有的能量: $E = -\mu \cdot B + C$, 如果取磁矩 μ 与外磁场 B 垂直时具有的能量为 0, 则 $E = -\mu \cdot B$. 磁矩在与磁矩方向相反的外加磁场中有了附加能量 $E = \mu \cdot B$, 磁矩在与磁矩方向相同的外加磁场中有了附加能量 $E = -\mu \cdot B$.

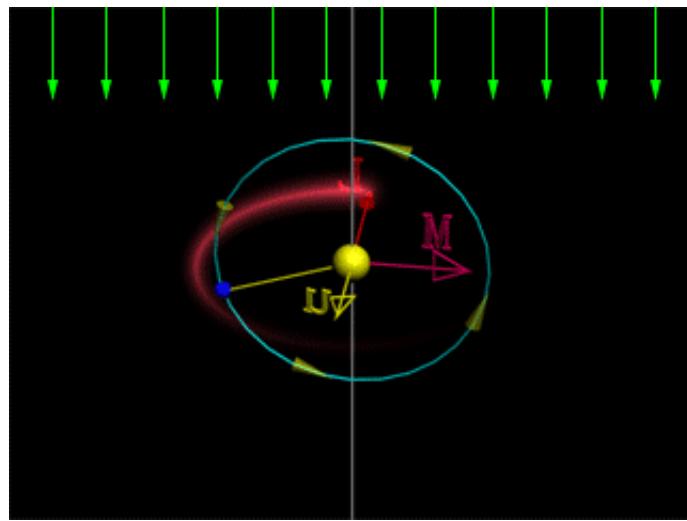


图 3.2: Magnetic Force Moment
Magnetic Force Moment

3.3.1.2.3 磁化

3.4 热力学

3.5 量子力学

3.6 量子光学

3.6.1 量子光学成像

3.6.1.1 简介

3.6.1.2 深度学习与量子成像

3.6.2 名词术语

Quantum Optics 量子光学 ([Quantum Optics](http://en.volupedia.org/wiki/Quantum_optics) (http://en.volupedia.org/wiki/Quantum_optics))

3.7 数学解释

3.7.1 参考文献

3.8 名词术语

Classical Mechanics 经典力学 ([Classical Mechanics](http://en.volupedia.org/wiki/Classical_mechanics) (http://en.volupedia.org/wiki/Classical_mechanics))

Electromagnetics 电磁学 ([Electromagnetics](http://en.volupedia.org/wiki/Electromagnetics) (<http://en.volupedia.org/wiki/Electromagnetics>))

Magnetic Moment 磁矩 ([Magnetic Moment](http://en.volupedia.org/wiki/Magnetic_moment) (http://en.volupedia.org/wiki/Magnetic_moment))

Mechanics 力学 ([Mechanics](http://en.volupedia.org/wiki/Mechanics) (<http://en.volupedia.org/wiki/Mechanics>))

Quantum Mechanics 经典力学 ([Quantum Mechanics](http://en.volupedia.org/wiki/Quantum_mechanics) (http://en.volupedia.org/wiki/Quantum_mechanics))

Spin 磁矩 ([Spin](http://en.volupedia.org/wiki/Spin) (<http://en.volupedia.org/wiki/Spin>))

Thermodynamics 热力学 ([Thermodynamics](http://en.volupedia.org/wiki/Thermodynamics) (<http://en.volupedia.org/wiki/Thermodynamics>))

CHAPTER 4

第四卷计算机学

4.1 操作系统

4.1.1 Linux 操作系统

4.1.1.1 常用命令

4.1.1.1.1 查找相关

4.1.1.1.1.1 查找文件

4.1.1.1.1.2 定位文件

1. 更新数据库: sudo updatedb
2. 查找定位文件: locate filename

4.1.1.1.2 更新源

```
sudo gedit /etc/apt/source.list
```

代码 4.1: /etc/apt/source.list

```
1 # deb cdrom:[Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64 (20160420.1)]/ xenial
  ↳main restricted
2 deb-src http://archive.ubuntu.com/ubuntu xenial main restricted #Added by software-
  ↳properties
```

(下页继续)

(续上页)

```
3 deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted
4 deb-src http://mirrors.aliyun.com/ubuntu/ xenial main restricted multiverse_
  ↪universe #Added by software-properties
5 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
6 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted_
  ↪multiverse universe #Added by software-properties
7 deb http://mirrors.aliyun.com/ubuntu/ xenial universe
8 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
9 deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse
10 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse
11 deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe_
  ↪multiverse
12 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted_
  ↪universe multiverse #Added by software-properties
13 deb http://archive.canonical.com/ubuntu xenial partner
14 deb-src http://archive.canonical.com/ubuntu xenial partner
15 deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
16 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted_
  ↪multiverse universe #Added by software-properties
17 deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
18 deb http://mirrors.aliyun.com/ubuntu/ xenial-security multiverse
```

4.1.1.2 常用技巧

4.1.1.2.1 第三方工具

4.1.1.2.1.1 多线程下载

Ubuntu 下可用的多线程下载工具有 `aria2c` (<http://aria2.github.io/>), `uget` (<https://ugetdm.com/>), `axel` (<https://wiki.ubuntu.org/Axel>) 等等, 且支持断点续传. 这里介绍使用 `aria2c` 多线程下载需要用户名密码认证的文件, 以下载 [ERS 雷达数据](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 为例, 对于单个文件可以使用

代码 4.2: 下载单个文件

```
1 # 无需认证下载
2 aria2c -x 16 downloadlink
3
4 # 需要认证
5 aria2c -x 16 -p --http-user=$yourusername --http-passwd=$yourpasswd downloadlink
```

对于多个文件, 可以自己写个脚本逐个下载, 新建 `urls` 文件, 添加文件下载链接 (每行代表一个下载链接), 如下

```
https://datapool.asf.alaska.edu/L0/E2/E2_84699_STD_L0_F303.zip
https://datapool.asf.alaska.edu/L1/E2/E2_84699_STD_F303.zip
https://datapool.asf.alaska.edu/L0/E2/E2_84699_STD_L0_F301.zip
```

(下页继续)

(续上页)

```
https://datapool.asf.alaska.edu/L1/E2/E2_84699_STD_F301.zip
https://datapool.asf.alaska.edu/L1/E2/E2_84690_STD_F137.zip
https://datapool.asf.alaska.edu/L0/E2/E2_84690_STD_L0_F137.zip
https://datapool.asf.alaska.edu/L1/E2/E2_84686_STD_F203.zip
https://datapool.asf.alaska.edu/L0/E2/E2_84686_STD_L0_F203.zip
https://datapool.asf.alaska.edu/L1/E2/E2_84697_STD_F273.zip
https://datapool.asf.alaska.edu/L0/E2/E2_84697_STD_L0_F273.zip
https://datapool.asf.alaska.edu/L0/R1/R1_65200_ST1_L0_F301.zip
https://datapool.asf.alaska.edu/L1/R1/R1_65200_ST1_F301.zip
```

新建 `download.sh` 文件, 添加如下代码, 执行 `./download.sh`, 即可实现从 `urls` 文件中读取每个文件的下载链接, 逐个下载.

代码 4.3: 下载单个文件

```
1 #!/bin/bash
2
3 # 1-16 -x, --max-connection-per-server=NUM The maximum number of connections to
4 # one for each download.
5 thread=16
6
7 FILE=urls
8 yourusername=[username]
9 yourpasswd=[passwd]
10
11 echo "#####
12 while read line;do
13     echo "Line # $k: $line"
14
15     aria2c -x $thread -p --http-user=$yourusername --http-passwd=$yourpasswd $line
16
17     ((k++))
18 done < $FILE
19 echo "$k files are downloaded!"
```

4.1.2 Windows 操作系统

4.2 程序设计

4.2.1 C 程序设计

4.2.1.1 引言

4.2.1.1.1 什么是 C 语言

4.2.1.2 图形交互界面开发

4.2.1.2.1 简介

4.2.2 Lua 程序设计

4.2.2.1 Lua 笔记

4.2.2.1.1 Lua 相关资料

4.2.2.1.1.1 基础

- [Lua 官网](http://www.lua.org) (<http://www.lua.org>)
- [Lua Users](http://lua-users.org/) (<http://lua-users.org/>)
- [Lua 5.3 英文参考手册](http://www.lua.org/manual/5.3/manual.html) (<http://www.lua.org/manual/5.3/manual.html>)
- [Lua 5.3 中文参考手册](http://www.lua.org/manual/5.3/manual.html) (<http://www.lua.org/manual/5.3/manual.html>)
- [Programming in Lua](http://vdisk.weibo.com/s/z63ACRNxK4cLS? sudaref=www.baidu.com) (<http://vdisk.weibo.com/s/z63ACRNxK4cLS? sudaref=www.baidu.com>)

4.2.2.1.2 从这里开始

在类 Unix 系统上 (Linux、OSMax 等等), 可能已经安装了 Lua, 或者有可用的安装包; 对于 Windows 可以使用 [LuaDist](http://luadist.org/) (<http://luadist.org/>) 等发布的预编译包.

由于 Lua 开源, 所以还有另外一种安装方法: 通过源码安装, 分构建和安装两步完成.

4.2.2.1.2.1 类 Unix 系统

- **构建 Lua (Build Lua) ** Lua 在类 Unix 系统上的构建很简单, 输入 make + 平台名即可, 如 Linux 平台下:

```
make linux
```

- **安装 Lua (Install Lua) ** 通过以下命令即可安装至默认位置.

```
make install
```

4.2.2.1.2.2 Windows

Windows 下的构建过程稍微复杂一些, 看到两种方案都挺好, 大同小异.

- 参考 Lua 官网给出的 Wiki 上的文章[Building Lua In Windows For Newbies](http://lua-users.org/wiki/BuildingLuaInWindowsForNewbies) (<http://lua-users.org/wiki/BuildingLuaInWindowsForNewbies>) 很容易实现, 基本上需要两样工具, 一是 Lua 源码, 二是编译工具 TDM-GCC.
- 参照[这里](http://www.thijsschreijer.nl/blog/?p=863) (<http://www.thijsschreijer.nl/blog/?p=863>), 需要安装 MinGW, 文中还给出了 LuaRocks 的安装流程.

4.2.2.1.2.3 方法一

构建 Lua

- 安装 GCC 编译器 TDM-GCC 的安装参照[这里](http://blog.csdn.net/enjoyyl/article/details/46545263#tdm-gccgccc) (<http://blog.csdn.net/enjoyyl/article/details/46545263#tdm-gccgccc>).
- 新建一个名为 **lua-gcc-Install** 的文件夹, 在里面新建文本文档, 命名为 **buil.cmd**, 将下面的代码复制到该文件中, 注意更改你的编译器路径和 lua 版本.

```
@echo off
:: =====
:: file build.cmd
:: =====
setlocal
:: you may change the following variable's value
:: to suit the downloaded version
set lua_version=5.3.2

set work_dir=%~dp0
:: Removes trailing backslash
:: to enhance readability in the following steps
set work_dir=%work_dir:~0,-1%
set lua_install_dir=%work_dir%\lua
set compiler_bin_dir=E:\devtools\TDM-GCC-64\bin
set lua_build_dir=%work_dir%\lua-%lua_version%
```

(下页继续)

(续上页)

```

set path=%compiler_bin_dir%;%path%

cd /D %lua_build_dir%
mingw32-make PLAT=mingw

echo.
echo **** COMPILATION TERMINATED ****
echo.
echo **** BUILDING BINARY DISTRIBUTION ****
echo.

:: create a clean "binary" installation
mkdir %lua_install_dir%
mkdir %lua_install_dir%\doc
mkdir %lua_install_dir%\bin
mkdir %lua_install_dir%\include

copy %lua_build_dir%\doc\*.* %lua_install_dir%\doc\*.*
copy %lua_build_dir%\src\*.exe %lua_install_dir%\bin\*.*
copy %lua_build_dir%\src\*.dll %lua_install_dir%\bin\*.*
copy %lua_build_dir%\src\luaconf.h %lua_install_dir%\include\*.*
copy %lua_build_dir%\src\lua.h %lua_install_dir%\include\*.*
copy %lua_build_dir%\src\lualib.h %lua_install_dir%\include\*.*
copy %lua_build_dir%\src\lauxlib.h %lua_install_dir%\include\*.*
copy %lua_build_dir%\src\lua.hpp %lua_install_dir%\include\*.*

echo.
echo **** BINARY DISTRIBUTION BUILT ****
echo.

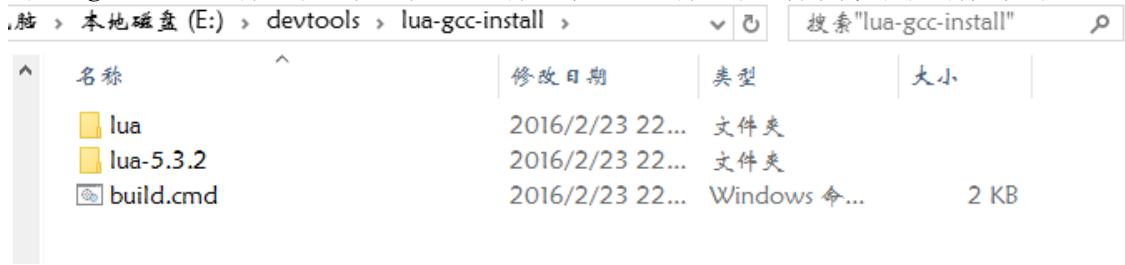
%lua_install_dir%\bin\lua.exe -e"print [[Hello!]];print[[Simple Lua test_
→successful!!!]]"

echo.

pause

```

- 将解压后的 lua 源码文件放入 **lua-gcc-install** 文件夹中, 运行 **build.cmd** 文件, 等待提示完成即可看到 **lua-gcc-install** 文件夹下多了一个 **lua** 文件夹 (**bin** 子文件夹下包含了需要的文件), 如下图所示:



安装 Lua 将上述构建的文件 (**bin** 下的文件) 复制到你的安装目录, 并添加路径至 PATH 环境变量即可.

**注: **由于是用 GCC 编译生成的, 所以在其它木有 GCC 的平台上是不能运行的.

4.2.2.1.2.4 方法 2

参考[Installing Lua on a Windows system](http://www.thijsschreijer.nl/blog/?p=863) (<http://www.thijsschreijer.nl/blog/?p=863>), 包含以下内容:

1. install a compiler ([MinGW \(and MSYS\)](http://www.mingw.org/wiki/Getting_Started) (http://www.mingw.org/wiki/Getting_Started))
2. compile and install [Lua](http://www.lua.org) (<http://www.lua.org>)
3. install [LuaRocks](http://www.luarocks.org/) (<http://www.luarocks.org/>)

install a compiler ([MinGW (and MSYS)])

这里, 我用的[MinGW](http://www.mingw.org/) (<http://www.mingw.org/>) 中的 GCC 版本为 4.8.1, 且安装在“E:”目录下, 所以添加: E:\devtools\MinGw\bin; 到系统环境变量 PATH.

compile and install Lua

去[Lua](http://www.lua.org) (<http://www.lua.org>) 下载 Lua 源码, 我用的 Lua5.3.2, 解压到任意路径, 如: “E:-5.3.2”, 然后以管理员身份打开 ** 命令提示符(管理员) **, 输入如下指令:

```
SET PATH=%PATH%;E:\devtools\MinGW\msys\1.0\bin
cd /d E:\lua-5.3.2
make clean
make mingw
make install INSTALL_TOP=E:/devtools/lua/5.3 TO_BIN="lua.exe luac.exe lua53.dll"
```

输出结果类似这样:

```
C:\windows\system32>>SET PATH=%PATH%;E:\devtools\MinGW\msys\1.0\bin

C:\windows\system32>cd /d E:\lua-5.3.2

E:\lua-5.3.2>make clean
cd src && make clean
make[1]: Entering directory `/e/lua-5.3.2/src'
rm -f liblua.a lua luac lapi.o lcode.o lctype.o ldebug.o ldo.o ldump.o lfunc.o lgc.
    .o llex.o lmem.o lobject.o lpcodes.o lparser.o lstate.o lstring.o ltable.o ltm.o_
    .o lundump.o lvm.o lzio.o lauxlib.o lbaselib.o lbitlib.o lcorolib.o ldblib.o liolib.
    .o lmathlib.o loslib.o lstrlib.o ltablib.o lutf8lib.o loadlib.o linit.o lua.o_
    .o luac.o
make[1]: Leaving directory `/e/lua-5.3.2/src'

E:\lua-5.3.2>make mingw
cd src && make mingw
make[1]: Entering directory `/e/lua-5.3.2/src'
make "LUA_A=lua53.dll" "LUA_T(lua.exe" \
      "AR=gcc -std=gnu99 -shared -o" "RANLIB=strip --strip-unneeded" \
      "SYSCFLAGS=-DLUA_BUILD_AS_DLL" "SYSLIBS=" "SYSLDFLAGS=-s" lua.exe
make[2]: Entering directory `/e/lua-5.3.2/src'
```

(下页继续)

(续上页)

```

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lua.
↳ o lua.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lapi.
↳ o lapi.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lcode.
↳ lcode.o lcode.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lctype.
↳ lctype.o lctype.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o ldebug.
↳ ldebug.o ldebug.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o ldo.
↳ o ldo.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o ldump.
↳ ldump.o ldump.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lfunc.
↳ lfunc.o lfunc.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lgc.
↳ o lgc.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o llex.
↳ o llex.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lmem.
↳ o lmem.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lobject.
↳ lobject.o lobject.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lopcodes.
↳ lopcodes.o lopcodes.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lparser.
↳ lparser.o lparser.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lstate.
↳ lstate.o lstate.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lstring.
↳ lstring.o lstring.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o ltable.
↳ ltable.o ltable.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o ltm.
↳ o ltm.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lundump.
↳ lundump.o lundump.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lvm.
↳ o lvm.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lzio.
↳ o lzio.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lauxlib.
↳ lauxlib.o lauxlib.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lbaselib.
↳ lbaselib.o lbaselib.c

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o lbitlib.
↳ lbitlib.o lbitlib.c

```

(下页继续)

(续上页)

```

gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳lcorolib.o lcorolib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳ldblib.o ldblib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳liolib.o liolib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳lmathlib.o lmathlib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳loslib.o loslib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳lstrlib.o lstrlib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳ltablib.o ltablib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳lutf8lib.o lutf8lib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳loadlib.o loadlib.c
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -DLUA_BUILD_AS_DLL -c -o \
↳linit.o linit.c
gcc -std=gnu99 -shared -o lua53.dll lapi.o lcode.o lctype.o ldebug.o ldo.o ldump.o \
↳lfunc.o lgc.o llex.o lmem.o lobject.o lopcodes.o lparser.o lstate.o lstring.o \
↳ltable.o ltm.o lundump.o lvm.o lzio.o lauxlib.o lbaselib.o lbitlib.o lcorolib.o \
↳ldblib.o liolib.o lmathlib.o loslib.o lstrlib.o ltablib.o lutf8lib.o loadlib.o \
↳linit.o
strip --strip-unneeded lua53.dll
gcc -std=gnu99 -o lua.exe -s lua.o lua53.dll -lm
make[2]: Leaving directory `/e/lua-5.3.2/src'
make "LUAC_T=luac.exe" luac.exe
make[2]: Entering directory `/e/lua-5.3.2/src'
gcc -std=gnu99 -O2 -Wall -Wextra -DLUA_COMPAT_5_2 -c -o luac.o luac.c
ar rcu liblua.a lapi.o lcode.o lctype.o ldebug.o ldo.o ldump.o lfunc.o lgc.o llex.o \
↳lmem.o lobject.o lopcodes.o lparser.o lstate.o lstring.o ltable.o ltm.o \
↳lundump.o lvm.o lzio.o lauxlib.o lbaselib.o lbitlib.o lcorolib.o ldblib.o liolib.o \
↳lmathlib.o loslib.o lstrlib.o ltablib.o lutf8lib.o loadlib.o linit.o
ranlib liblua.a
gcc -std=gnu99 -o luac.exe luac.o liblua.a -lm
make[2]: Leaving directory `/e/lua-5.3.2/src'
make[1]: Leaving directory `/e/lua-5.3.2/src'

E:\lua-5.3.2>make install INSTALL_TOP=E:/devtools/lua/5.3 TO_BIN="lua.exe luac.exe \
↳lua53.dll"
cd src && mkdir -p E:/devtools/lua/5.3/bin E:/devtools/lua/5.3/include E:/devtools/lua/5.3/lib E:/devtools/lua/5.3/man/man1 E:/devtools/lua/5.3/share/lua/5.3 E:/devtools/lua/5.3/lib/lua/5.3
cd src && install -p -m 0755 lua.exe luac.exe lua53.dll E:/devtools/lua/5.3/bin
cd src && install -p -m 0644 lua.h luaconf.h lualib.h lauxlib.h lua.hpp E:/devtools/lua/5.3/include

```

(下页继续)

(续上页)

```
cd src && install -p -m 0644 liblua.a E:/devtools/lua/5.3/lib  
cd doc && install -p -m 0644 lua.1 luac.1 E:/devtools/lua/5.3/man/man1  
  
E:\lua-5.3.2>
```

最后一步是添加 Lua 安装路径 E:\devtools\lua\5.3\bin 到系统环境变量 PATH.

在 cmd 终端输入: lua 即可进入 Lua 解释器交互界面, 也可以输入 lua -e "print('hello world')" 来测试安装是否正确(打印出 hello world).

install LuaRocks

下载 Windows 版的LuaRocks (<http://luarocks.org/releases/>), 解压到 E 盘, 然后输入如下命令安装, 其中, /P E:\devtools\LuaRocks 指定安装目录, 可以使用 install /? 查看更多选项.

```
cd luarocks-2.3.0-win32  
install /MW /F /LV 5.3 /P E:\devtools\LuaRocks
```

注: 执行完毕后, 末尾打印如下类似信息, 告诉你如何添加环境变量:

```
=====  
== LuaRocks is installed! ==  
=====  
  
You may want to add the following elements to your paths;  
Lua interpreter;  
  PATH      :   E:\devtools\lua\5.3\bin\  
  PATHEXT  :   .LUA  
LuaRocks;  
  PATH      :   E:\devtools\LuaRocks  
  LUA_PATH :   E:\devtools\LuaRocks\lua\?.lua;E:\devtools\LuaRocks\lua\?\init.lua  
Local user rocktree (Note: %APPDATA% is user dependent);  
  PATH      :   %APPDATA%\LuaRocks\bin  
  LUA_PATH :   %APPDATA%\LuaRocks\share\lua\5.3\?.lua;%APPDATA%\LuaRocks\share\lua\5.3\?\init.lua  
  LUA_CPATH:   %APPDATA%\LuaRocks\lib\lua\5.3\?.dll  
System rocktree  
  PATH      :   e:\devtools\lua\5.3\bin  
  LUA_PATH :   e:\devtools\lua\5.3\share\lua\5.3\?.lua;e:\devtools\lua\5.3\share\lua\5.3\?\init.lua  
  LUA_CPATH:   e:\devtools\lua\5.3\lib\lua\5.3\?.dll  
  
Note that the %APPDATA% element in the paths above is user specific and it MUST be  
→ replaced by its actual value.  
For the current user that value is: C:\Users\liu\AppData\Roaming.
```

根据上述, 添加必要的环境变量, 由于 PATH 已经存在, 所以只需要 LUA_PATH 和 LUA_CPATH, 文[Installing Lua on a Windows system](http://www.thijschreijer.nl/blog/?p=863) (<http://www.thijschreijer.nl/blog/?p=863>) 中说, 对于 Lua5.3, 添加

LUA_PATH_5_3 和 LUA_CPATH_5_3, 我试了一下, 两个都对, 如果你安装了不同版本的 Lua, 应该用后者指示不同版本.

为测试 LuaRocks 安装正确, cmd 命令行输入 luarocks help 会打印很多帮助信息.

比如可以通过 luarocks install pakage 来安装包, 通过 luarocks remove pakage 来卸载包
举例: 安装 luafsystem

```
C:\windows\system32>luarocks install luafsystem
Installing https://luarocks.org/luafsystem-1.6.3-2.src.rock...
Using https://luarocks.org/luafsystem-1.6.3-2.src.rock... switching to 'build' mode
mingw32-gcc -O2 -c -o src/lfs.o -IE:/devtools/lua/5.3/include/ src/lfs.c
In file included from src/lfs.c:67:0:
src/lfs.h:21:0: warning: "fileno" redefined [enabled by default]
#define fileno(f) (_fileno(f))
^
In file included from src/lfs.c:38:0:
e:\devtools\mingw\include\stdio.h:595:0: note: this is the location of the previous definition
#define fileno(__F) ((__F)->_file)
^
mingw32-gcc -shared -o lfs.dll src/lfs.o E:/devtools/lua/5.3/bin/lua53.dll -lm
Updating manifest for e:\devtools\lua\5.3\lib\luarocks\rocks
luafsystem 1.6.3-2 is now built and installed in e:\devtools\lua\5.3\ (license: MIT/X11)
```

4.2.2.1.3 语法

- [Lua 5.3 英文参考手册](http://www.lua.org/manual/5.3/manual.html) (<http://www.lua.org/manual/5.3/manual.html>)
- [Lua 5.3 中文参考手册](http://www.lua.org/manual/5.3/manual.html) (<http://www.lua.org/manual/5.3/manual.html>)
- [Programming in Lua](http://vdisk.weibo.com/s/z63ACRNxK4cLS? sudaref=www.baidu.com) (<http://vdisk.weibo.com/s/z63ACRNxK4cLS? sudaref=www.baidu.com>)

4.2.2.1.3.1 多重赋值

在 Lua 中, 多重赋值是合法的, 如 `a, b, c = 1, 2, 3`, 则相当于 `a=1; b=2; c=3`. 参考[Lua 5.3 Reference Manual §3.3.3 –Assignment](http://www.lua.org/manual/5.3/manual.html#3.3.3) (<http://www.lua.org/manual/5.3/manual.html#3.3.3>), 可以看到赋值规则是这样的: [Lua5.3][Lua53refman]

Before the assignment, the list of values is adjusted to the length of the list of variables. If there are more values than needed, the excess values are thrown away. If there are fewer values than needed, the list is extended with as many nil's as needed. If the list of expressions ends with a function call, then all values returned by that call enter the list of values, before the adjustment (except when the call is enclosed in parentheses; see §3.4). 在作赋值操作之前, 那值列表会被调整为左边变量列表的个数. 如果值比需要的更多的话, 多余的值就被扔掉. 如果值的数量不够需求, 将会按所需扩展若

干个 nil. 如果表达式列表以一个函数调用结束, 这个函数所返回的所有值都会在调整操作之前被置入值列表中 (除非这个函数调用被用括号括了起来; 参见 §3.4) .

The assignment statement first evaluates all its expressions and only then the assignments are performed. 赋值语句首先让所有的表达式完成运算, 之后再做赋值操作.

看下面一段代码及其对应输出, 函数 **maxmin** 有两个返回值, 依次输出两个输入参数的最大和最小.

```

function maxmin( a, b )
  if a < b then
    return b, a
  else
    return a,b
  end
end

a, b, c, d = maxmin(2,3), 1, 4
print(a,b,c,d)           -- <--> 3   1   4   nil

a, b, c, d = 1, maxmin(2,3), 4
print(a,b,c,d)           -- <--> 1   3   4   nil

a, b, c, d = 1, 4, maxmin(2,3)
print(a,b,c,d)           -- <--> 1   4   3   2

a, b, c, d = 1, 4, (maxmin(2,3))
print(a,b,c,d)           -- <--> 1   4   3   nil

i = 3; a = {}
i, a[i] = i+1, 20
print(i, a[3], a[4])      -- <--> 4   20   nil

-- swap value
x = 1; y = 2; z = 3
x,y,z = y,z,x
print(x,y,z)             -- <--> 2   3   1

```

由上面的规则可知: 1. 只有当表达式列表以一个函数调用结束, 并且函数调用没有被用括号 “`O`“ 括起来, 这个函数所返回的所有值才会在调整操作之前都被置入值列表中, 所以只有 `a, b, c, d = 1, 4, maxmin(2,3)` 对应的右端值列表中有 4 个值; 2. 赋值语句首先让所有的表达式完成运算, 之后再做赋值操作, 所以 `i, a[i] = i+1, 20`, 只改变了 `a[3]` 的值.

4.2.2.1.3.2 协同

- 线程和协同主要区别: 线程同时运行多个线程, 协同在同一时刻只有一个协同程序运行, 且只有在运行的协同程序明确要求挂起才挂起;
- 给 yield 的参数会传给 resume, 主函数返回值也会传给 resume, 传给 resume 的参数也会传给 yield;
- yield 是挂起程序, resume 时, yield 接受 resume 传来的新的参数, 从挂起处开始执行;

```

co = coroutine.create(function ()
    print('suspended follows: ')
    print("co", coroutine.yield())
    -- body
end)
print(coroutine.resume(co))           --> suspended follows -->true
print(coroutine.status(co))          --> suspended
print(coroutine.resume(co, 4, 5))     -- co 4 5 -->true
print(coroutine.status(co))          --> dead
print(coroutine.resume(co, 4, 5))     --> false cannot resume dead coroutine

print('-----')
co = coroutine.create(function()
    return 6, 7
end)
print(coroutine.resume(co))           --> true 6 7

```

4.2.2.1.4 GUI 开发

可以使用 [wxLua](http://wxlua.sourceforge.net/) (<http://wxlua.sourceforge.net/>) 进行 GUI 开发, 参考 [Build using GCC in Linux](http://wxlua.sourceforge.net/docs/install.html#C2.5) (<http://wxlua.sourceforge.net/docs/install.html#C2.5>) 在 Linux 上安装, Windows 上直接下载二进制包安装即可.

4.2.2.2 wxLua 简明教程

4.2.3 IDL 程序设计

4.2.3.1 引言

4.2.3.1.1 什么是 IDL 语言

交互开发语言 (Interactive Data Language, IDL)

4.2.3.1.2 基于 IDL 语言的库与软件

- [idlastro](https://idlastro.gsfc.nasa.gov/homepage.html) (<https://idlastro.gsfc.nasa.gov/homepage.html>) : The IDL Astronomy User's Library
- [Solarsoft](#) : A library for solar data analysis and spacecraft operation activities written predominately in IDL
- [idlcoyote](http://www.idlcoyote.com/index.html) (<http://www.idlcoyote.com/index.html>) : A tutorial for IDL programing,
- [ENVI IDL 博客教程](http://blog.sina.com.cn/s/articlelist_1984634525_2_1.html) (http://blog.sina.com.cn/s/articlelist_1984634525_2_1.html)

4.2.4 Lisp 程序设计

4.2.4.1 引言

4.2.4.1.1 什么是 Lisp 语言

Common Lisp 语言的实现(解释器)有多种,如 **CLISP**, **SBCL (Steel Bank Common Lisp)**, **CCL (Clozure Common Lisp)**等等,其功能特性对比如下表 4.1 所示

表 4.1: Common Lisp 语言的实现特性对比

实现名称	首页	功能特性	适用平台	其它
CLISP	http://clisp.org/	解释、编译、运行、调试、外部函数接口等等	Linux、Windows、MacOS X、其它类 Unix 系统	开源
SBCL	http://www.sbcl.org/	解释、编译、运行、调试、统计分析、代码覆盖等等	Linux、Windows、MacOS X、其它类 Unix 系统	开源免费
CCL	https://ccl.clozure.com/	快速编译、本机线程、精确、压缩的垃圾收集、方便的外部函数接口等等	Linux (x86, x86-64, ppc32, ppc64, armv7l/armv6)、Windows (x86, x86-64)、Mac OS X (x86, x86-64)、其它类 Unix 系统	开源

提示: 资源链接

- [common lisp](https://www.common-lisp.net/) (<https://www.common-lisp.net/>) : The amazing world of Common Lisp, the programmable programming language.
- [lispbox](https://common-lisp.net/project/lispbox/) (<https://common-lisp.net/project/lispbox/>) : 集成了 Lisp 语言包, Emacs, Slime 的集成开发环境.
- [ANSI Common Lisp ZH](https://acl.readthedocs.io/en/latest/zhCN/) (<https://acl.readthedocs.io/en/latest/zhCN/>) : ANSI Common Lisp 中文版.
- [practical common lisp](http://www.gigamonkeys.com/book/) (<http://www.gigamonkeys.com/book/>) : An book on Common Lisp programing.

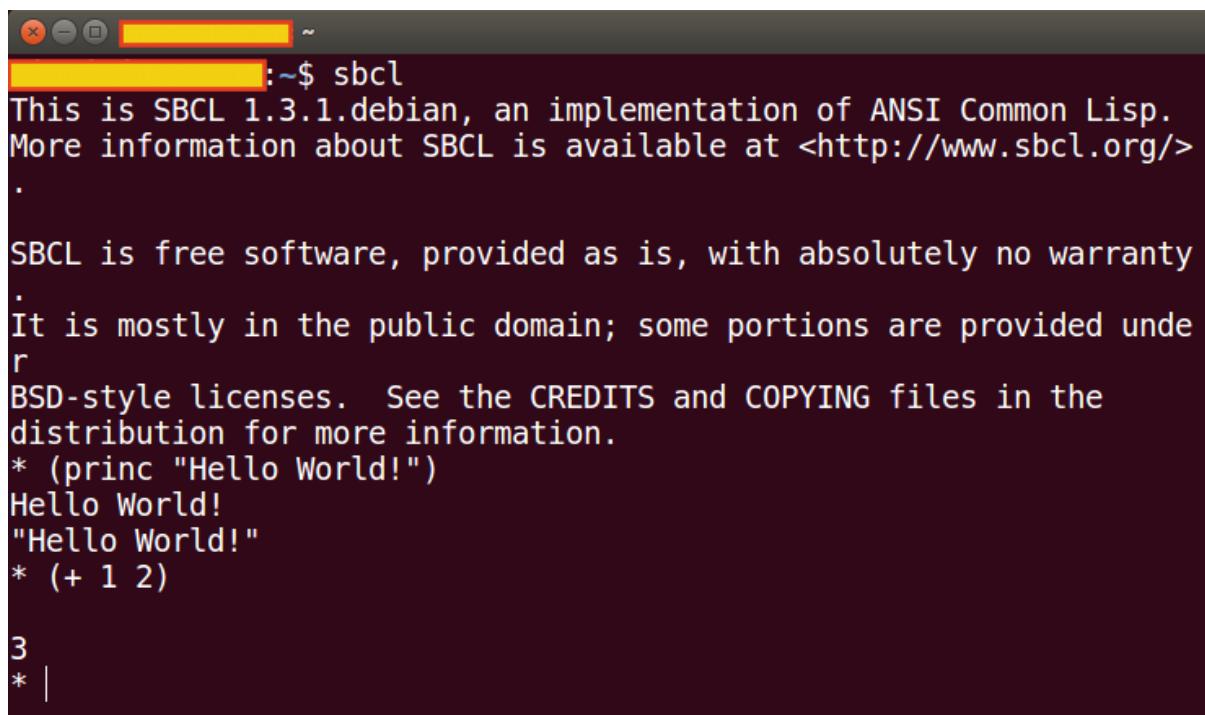
4.2.4.1.2 开发环境配置

4.2.4.1.2.1 安装 Common Lisp 实现

本节介绍 Common Lisp 实现的安装.

4.2.4.1.2.2 安装 SBCL 实现

Ubuntu 上安装 SBCL 十分简单, 只需通过执行命令 `sudo apt install sbcl` 即可安装, 安装完成后, 终端输入 `sbcl` 即可进入交互界面, 如 图 4.1 所示.



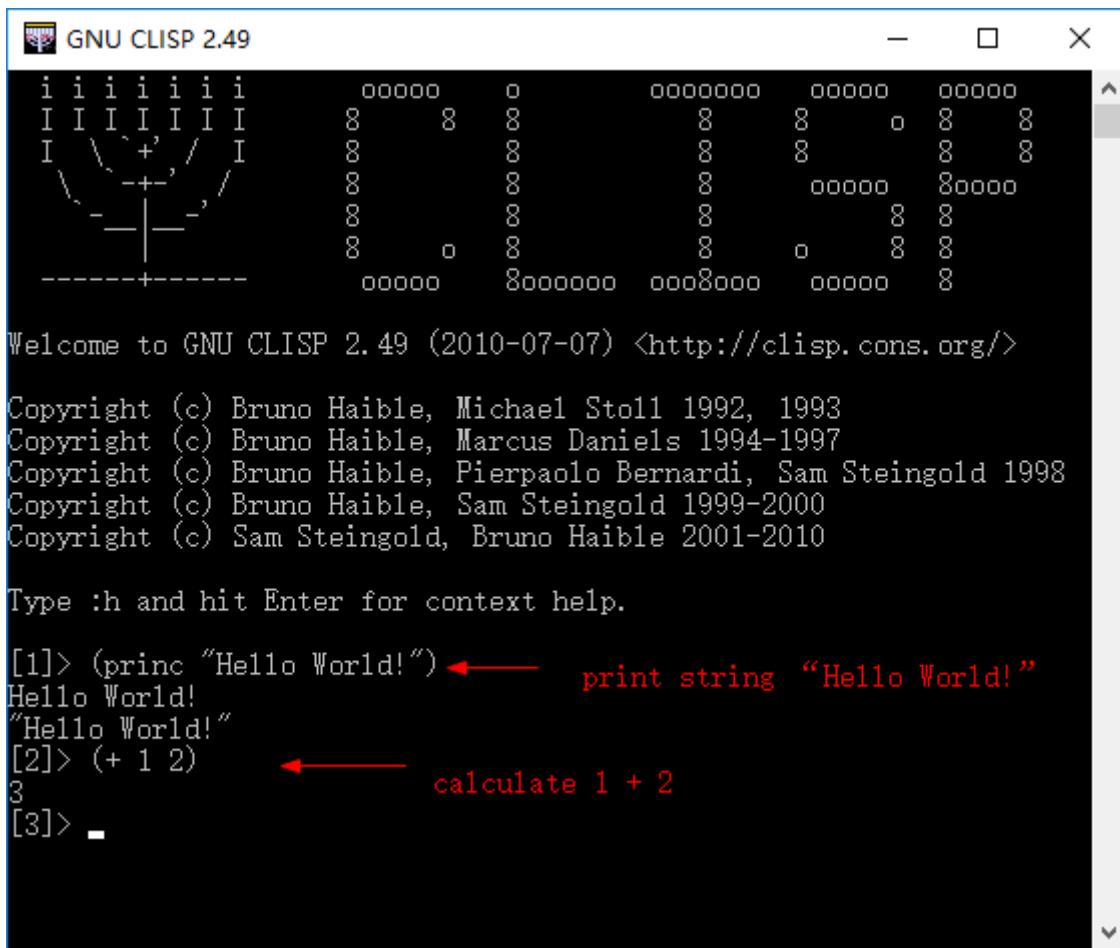
```
:~$ sbcl
This is SBCL 1.3.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>
.
SBCL is free software, provided as is, with absolutely no warranty
.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.
* (princ "Hello World!")
Hello World!
"Hello World!"
* (+ 1 2)

3
* |
```

图 4.1: Welcome interface of SBCL on Ubuntu
Ubuntu 上的 SBCL 欢迎界面

4.2.4.1.2.3 安装 CLISP 实现

CLISP 安装很简单, 从 clisp.org (`http://clisp.org/`) 可以下载安装包或源码. 对于 Windows 系统, 执行下载的可执行文件 `clisp-2.49-win32-mingw-big.exe` 安装即可, 安装完成后双击打开 `clisp.exe` 即可进入交互式界面, 如 图 4.2 所示. 对于 Ubuntu 系统, 可以直接通过包管理器安装, 打开终端, 输入命令 `sudo apt-get install clisp` 即可安装, 安装完成后, 终端输入 `clisp` 命令进入交互式界面, 如 图 4.3 所示.



The screenshot shows the GNU CLISP 2.49 window on Windows. The title bar reads "GNU CLISP 2.49". The window contains the following text:

```
i i i i i i i      00000   o      00000000  00000   00000
I I I I I I I      8     8   8      8     8   o   8     8
I \ ` + / I        8     8   8      8     8   8   8     8
\ -+ , /           8     8   8      8     8   00000   80000
- -| -             8     8   8      8     8   8   8
-----+-----       8     o   8      8     8   o   8   8
          00000   8000000  0008000   00000   8

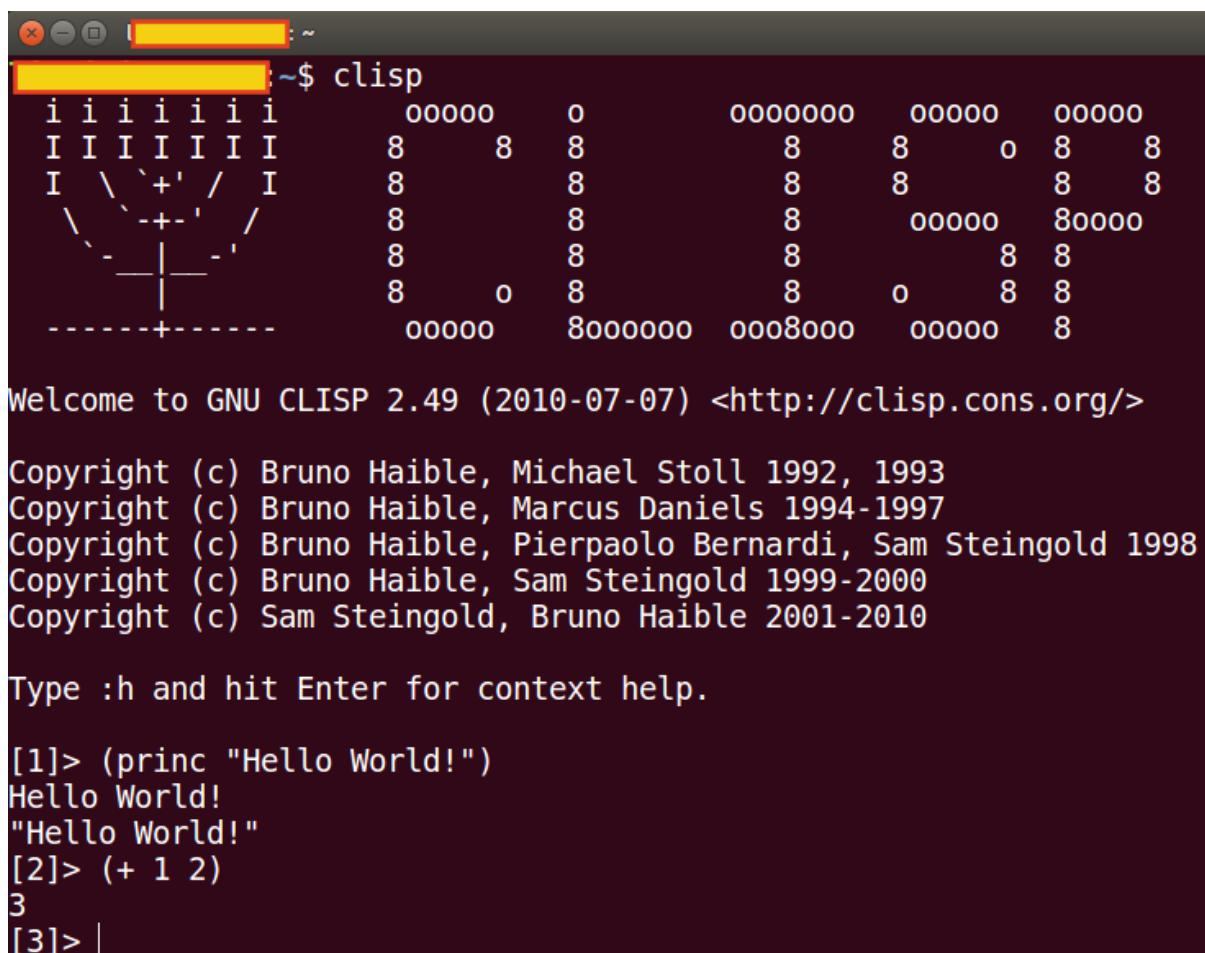
Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>

Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010

Type :h and hit Enter for context help.

[1]> (princ "Hello World!") ← print string "Hello World!"
Hello World!
"Hello World!"
[2]> (+ 1 2) ← calculate 1 + 2
3
[3]> -
```

图 4.2: Welcome interface of Lisp on Windows
Windows 上的 Lisp 欢迎界面



The screenshot shows a terminal window on a dark-themed desktop environment. The title bar is yellow. The terminal prompt is `~$ clisp`. Below the prompt, there is a large, stylized ASCII art logo consisting of letters and numbers. The text "Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>" follows. A series of copyright notices from 1992 to 2010 are listed. The text "Type :h and hit Enter for context help." is displayed. Finally, three command-line interactions are shown:

```
[1]> (princ "Hello World!")
Hello World!
"Hello World!"
[2]> (+ 1 2)
3
[3]> |
```

图 4.3: Welcome interface of Lisp on Ubuntu
Ubuntu 上的 Lisp 欢迎界面

4.2.4.1.2.4 开发环境配置

4.2.4.1.2.5 Emacs + Slime 环境

4.2.4.1.2.6 Sublime 环境

4.2.4.1.3 Lisp 编程入门

4.2.4.1.3.1 交互式界面使用

图 4.2 和 图 4.3 中同时给出了两个示例: 1) 打印字符串 ((princ "Hello World!")); 2) 计算 1+2 的和 ((+ 1 2)), 你可以尝试一下其它表达式的计算.

4.2.4.1.3.2 第一个 Lisp 程序

这里介绍如何像在 Python 和 Lua 中那样导入其它文件中的函数. 首先建立文件 *add.lisp*, 加入如下代码, 代码仅含 1 行, 定义了一个返回两个数加和的函数 *add()*.

代码 4.4: *add.lisp*

```
1 (defun add (&key a b) (+ a b))
```

然后新建 *first.lisp* 文件, 并加入如下代码, 代码第 1 行完成加载 ‘add’ 模块的功能; 第 3 行完成打印字符串 Hello world! 的功能; 第 5 行调用 *add()*, 传入参数 $a = 1, b = 2$, 并将返回值赋予变量 *c*; 第 7 行以两位小数格式化打印结果, 第 9 行是将程序编译成可执行文件 *main.exe*.

代码 4.5: *first.lisp*

```
1 (load "D:\\zhiliu\\ws\\Lisp\\add.lisp")
2
3 (princ "Hello world!")
4
5 (setq c (add :a 1 :b 2))
6
7 (print c)
8
9 (EXT:SAVEINITMEM "main" :QUIET t :INIT-FUNCTION 'main :EXECUTABLE t :NORC t)
```

打开 lisp 交互界面, 输入 (load "D:\\zhiliu\\ws\\Lisp\\first.lisp") 可以加载并执行 *first* 模块. 也可以打开系统命令行终端, 输入 sbcl --load .\first.lisp 或 clisp .\first.lisp 运行程序, 如 图 4.4 所示

提示: 以上代码可以直接在交互式界面窗口执行.

```

Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation. 保留所有权利。
PS C:\Users\gaochu> D:
PS D:> cd ..\zhiliu\ws\Lisp\
PS D:\zhiliu\ws\Lisp> ls
    目录: D:\zhiliu\ws\Lisp
Mode LastWriteTime      Length Name
---- -----          ---- -
-a-- 2019/12/11 18:16          32 add.lisp
-a-- 2019/12/11 18:27         108 first.lisp
PS D:\zhiliu\ws\Lisp> clisp.exe .\first.lisp
Hello world!
3

```

图 4.4: Run lisp file

运行 lisp 程序

4.2.4.1.4 包库管理

4.2.4.1.4.1 quicklisp

4.2.4.1.4.2 安装

[quicklisp](https://www.quicklisp.org/beta/) (<https://www.quicklisp.org/beta/>) 是 Common Lisp 的库管理器, 支持用简单的几条命令下载, 安装和加载库。quicklisp 的安装很简单, 以 Ubuntu 系统为例, 从 quicklisp 首页下载 *quicklisp.lisp* 安装脚本文件, 打开终端, 进入 *quicklisp* 文件所在目录, 执行 `clisp --load quicklisp.lisp` 或 `sbcl --load quicklisp.lisp` 进入安装界面, 如图 4.5 所示, 按照提示, 执行 `(quicklisp-quickstart:install :path "/mnt/e/sfw/lisplib/quicklisp/")` lisp 命令既可安装, 其中 path 用于指定安装路径。

```

/mnt/e/library/lisp/quicklisp
:/mnt/e/library/lisp/quicklisp$ sbcl --load quicklisp.lisp
This is SBCL 1.3.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.

===== quicklisp quickstart 2015-01-28 loaded =====

To continue with installation, evaluate: (quicklisp-quickstart:install)

For installation options, evaluate: (quicklisp-quickstart:help)

* (quicklisp-quickstart:help)

===== quicklisp quickstart install help =====

```

图 4.5: Install quicklisp on ubuntu

Install quicklisp on ubuntu

4.2.4.1.4.3 设置默认加载

在 Lisp 交互界面, 首先加载 quicklisp, 然后设置为启动默认加载, 命令如下:

```
(load "/mnt/e/sfw/lisplib/quicklisp/setup.lisp")
(ql:add-to-init-file)
```

命令输出如下:

```
[1]> (load "/mnt/e/sfw/lisplib/quicklisp/setup.lisp")
;; Loading file /mnt/e/sfw/lisplib/quicklisp/setup.lisp ...
;; Loading file /mnt/e/sfw/lisplib/quicklisp/cache/asdf-fasls/0hq63s/asdf.fas ...
;; Loaded file /mnt/e/sfw/lisplib/quicklisp/cache/asdf-fasls/0hq63s/asdf.fas
;; Loaded file /mnt/e/sfw/lisplib/quicklisp/setup.lisp
T
[2]> (ql:add-to-init-file)
I will append the following lines to #P"/home/liu/.clisprc.lisp":

;; The following lines added by ql:add-to-init-file:
 #-quicklisp
 (let ((quicklisp-init
       (merge-pathnames "/mnt/e/sfw/lisplib/quicklisp/setup.lisp"
                     (user-homedir-pathname))))
    (when (probe-file quicklisp-init)
      (load quicklisp-init)))

Press Enter to continue.
```

4.2.4.1.4.4 使用

提示: quicklisp 用法

- To load/install a system, use: (ql:quickload “system-name”)
 - To remove a system, use: (ql:uninstall “system-name”)
 - To find systems, use: (ql:system-apropos “term”)
 - To get updated software, use: (ql:update-dist “quicklisp”)
 - To update the Quicklisp client, use: (ql:update-client)
 - To see what systems depend on a particular system, use: (ql:who-depends-on “system-name”)
 - To load Quicklisp every time you start Lisp, use: (ql:add-to-init-file)
 - For more information, see <http://www.quicklisp.org/beta/>
-

以安装 `lisp-binary` (<https://github.com/j3pic/lisp-binary>) 进入 Lisp 交互环境, 执行 (ql:quickload "lisp-binary") 命令即可安装, 安装日志如下

(下页继续)

(续上页)

```
[package simple-bit-stream] .....  
[package reverse-stream] .....  
[package lisp-binary] .....  
.....  
("lisp-binary")  
*
```

4.2.4.1.5 生成可独立执行文件

Lisp 是脚本语言，具备跨平台的特性。Ecl 是一个开源的 Lisp 语言实现，可以将 lisp 文件编译成独立可执行程序。

3. 编译 CL-USER> (setf c::delete-files nil) CL-USER> (compile-file “test.lisp” :system-p t) CL-USER> (c:build-program “test” :lisp-files ‘(“test.o”))

4. 执行 #./test

5.github demo:<https://github.com/cwndrws/lisp-c-example>

4.2.4.2 基础语法

4.2.5 Python 程序设计

4.2.5.1 简介

4.2.5.1.1 资源汇总

- [Python tutorials](https://www.tutorialspoint.com/python/) (<https://www.tutorialspoint.com/python/>)

4.2.5.2 Anaconda 简明教程

4.2.5.2.1 安装第三方包

4.2.5.2.1.1 本地文件安装

```
conda install --use-local your-pkg-path
```

4.2.5.2.2 安装包测试

```
conda install --debug package_name  
conda install --debug mkl
```

4.2.6 Matlab 程序设计

4.2.6.1 引言

4.2.6.1.1 什么是 Matlab 语言

4.2.7 Julia 程序设计

4.2.7.1 引言

4.2.7.1.1 什么是 Julia 语言

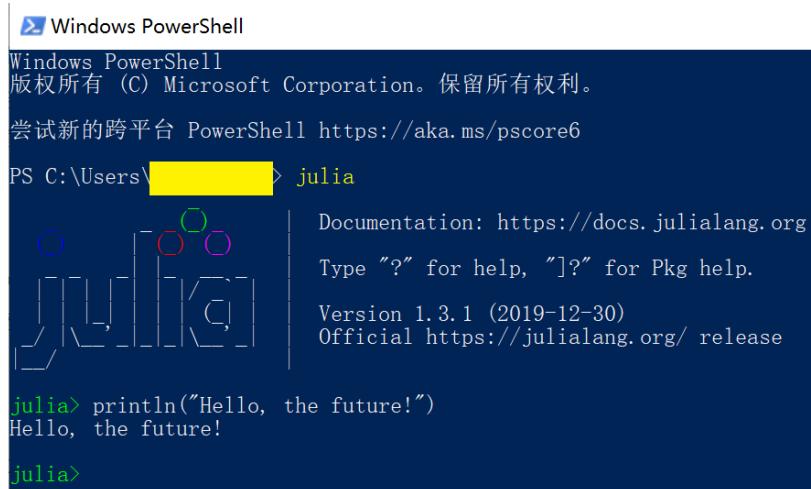
提示: 资源链接

- [Homepage of Julia](https://julialang.org/) (<https://julialang.org/>)
 - [Github of Julia](https://github.com/JuliaLang) (<https://github.com/JuliaLang>)
 - [Julia doc, en](https://docs.julialang.org/) (<https://docs.julialang.org/>)
 - [Julia doc, zh](https://docs.juliacn.com/latest/) (<https://docs.juliacn.com/latest/>)
 - [Julia By Example](https://juliabyexample.helpmanual.io/) (<https://juliabyexample.helpmanual.io/>)
-

4.2.7.1.2 开发环境配置

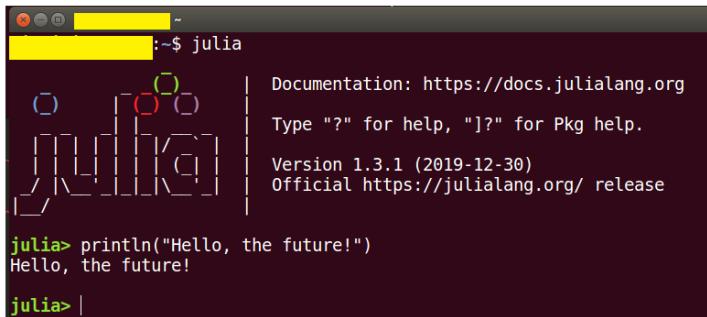
4.2.7.1.2.1 安装 Julia 实现

Julia 的安装十分简单, 从 [主页](https://julialang.org/) (<https://julialang.org/>) 下载对应操作系统版本的安装包. 对于 Windows 系统, 双击安装包根据提示安装即可; 对于 Linux 系统, 下载解压后将 bin 目录添加至系统环境变量 PATH 中即可. 添加好环境变量, 打开终端, 输入 julia 进入 Julia 交互式界面, 如 图 4.6 所示.



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Users\yellow> julia
julia> Documentation: https://docs.julialang.org
julia> Type "?" for help, "]??" for Pkg help.
julia> Version 1.3.1 (2019-12-30)
julia> Official https://julialang.org/ release
julia> println("Hello, the future!")
Hello, the future!
julia>
```

(a)



```
:~$ julia
julia> Documentation: https://docs.julialang.org
julia> Type "?" for help, "]??" for Pkg help.
julia> Version 1.3.1 (2019-12-30)
julia> Official https://julialang.org/ release
julia> println("Hello, the future!")
Hello, the future!
julia> |
```

(b)

图 4.6: Welcome interface of Julia on Windows and Ubuntu

4.2.7.1.2.2 开发环境配置

4.2.7.1.2.3 Sublime 环境

如果想直接在 Sublime 中运行 Julia 程序, 可以添加 Julia 编译选项, 然后通过 $\text{Ctrl}+\text{b}$ 或 $\text{Ctrl}+\text{Shift}+\text{b}$ 选择编译。具体方法为: 新建 `sublime_rootdir/Packages/Users/Julia.sublime-build` 文件, 输入如下代码, 保存即可:

```
{
  "cmd": ["julia", "$file"],
  "file_regex": "^(?:julia:)?(\\t)(...*)?([0-9]*):?([0-9]*)",
  "selector": "source.jl, source.julia"
}
```

4.2.7.1.3 Julia 编程入门

4.2.7.1.3.1 交互式界面使用

4.2.7.1.3.2 第一个 Julia 程序

这里介绍如何像在 Python 和 Lua 中那样导入其它文件中的函数。首先建立文件 `add.lisp`, 加入如下代码, 代码仅含 1 行, 定义了一个返回两个数加和的函数 `add()`。

代码 4.6: add.lisp

```
1 (defun add (&key a b) (+ a b))
```

然后新建 `first.lisp` 文件, 并加入如下代码, 代码第 1 行完成加载 ‘`add`’ 模块的功能; 第 3 行完成打印字符串 `Hello world!` 的功能; 第 5 行调用 `add()`, 传入参数 $a = 1, b = 2$, 并将返回值赋予变量 `c`; 第 7 行以两位小数格式化打印结果, 第 9 行是将程序编译成可执行文件 `main.exe`。

代码 4.7: first.lisp

```
1 (load "D:\\zhiliu\\ws\\Lisp\\add.lisp")
2
3 (princ "Hello world!")
4
5 (setq c (add :a 1 :b 2))
6
7 (print c)
8
9 (EXT:SAVEINITMEM "main" :QUIET t :INIT-FUNCTION 'main :EXECUTABLE t :NORC t)
```

打开 lisp 交互界面, 输入 `(load "D:\\zhiliu\\ws\\Lisp\\first.lisp")` 可以加载并执行 `first` 模块。也可以打开系统命令行终端, 输入 `sbcl --load .\\first.lisp` 或 `clisp .\\first.lisp` 运行程序, 如 图 4.4 所示

```

Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation. 保留所有权利。
PS C:\Users\gaochu> D:
PS D:> cd ..\zhiliu\ws\Lisp\
PS D:\zhiliu\ws\Lisp> ls
    目录: D:\zhiliu\ws\Lisp
Mode LastWriteTime      Length Name
---- -----          ---- 
-a--- 2019/12/11 18:16          32 add.lisp
-a--- 2019/12/11 18:27         108 first.lisp
PS D:\zhiliu\ws\Lisp> clisp.exe .\first.lisp
Hello world!
3
T

```

```

GNU CLISP 2.49
Break 11 [12]>
Break 11 [12]> (load "D:\zhiliu\ws\Lisp\first.lisp")
;; Loading file D:\zhiliu\ws\Lisp\first.lisp ...
;; Loading file D:\zhiliu\ws\Lisp\add.lisp ...
;; Loaded file D:\zhiliu\ws\Lisp\add.lisp
Hello world!
3      1+2=3
T      Loaded file D:\zhiliu\ws\Lisp\first.lisp
Break 11 [12]> load succeed

```

图 4.7: Run lisp file

运行 lisp 程序

提示: 以上代码可以直接在交互式界面窗口执行.

4.2.7.1.4 包库管理

4.2.7.1.4.1 quicklisp

4.2.7.1.4.2 安装

quicklisp (<https://www.quicklisp.org/beta/>) 是 Common Lisp 的库管理器, 支持用简单的几条命令下载, 安装和加载库. quicklisp 的安装很简单, 以 Ubuntu 系统为例, 从 quicklisp 首页下载 *quicklisp.lisp* 安装脚本文件, 打开终端, 进入 *quicklisp* 文件所在目录, 执行 *clisp --load quicklisp.lisp* 或 *sbcl --load quicklisp.lisp* 进入安装界面, 如 图 4.5 所示, 按照提示, 执行 *(quicklisp-quickstart:install :path "/mnt/e/sfw/lisplib/quicklisp")* lisp 命令既可安装, 其中 path 用于指定安装路径.

4.2.7.1.4.3 设置默认加载

在 Lisp 交互界面, 首先加载 quicklisp, 然后设置为启动默认加载, 命令如下:

```
(load "/mnt/e/sfw/lisplib/quicklisp/setup.lisp")
(ql:add-to-init-file)
```

命令输出如下:

```
[1]> (load "/mnt/e/sfw/lisplib/quicklisp/setup.lisp")
;; Loading file /mnt/e/sfw/lisplib/quicklisp/setup.lisp ...
;; Loading file /mnt/e/sfw/lisplib/quicklisp/cache/asdf-fasls/0hq63s/asdf.fas ...
;; Loaded file /mnt/e/sfw/lisplib/quicklisp/cache/asdf-fasls/0hq63s/asdf.fas
;; Loaded file /mnt/e/sfw/lisplib/quicklisp/setup.lisp
T
[2]> (ql:add-to-init-file)
```

(下页继续)

```
/mnt/e/library/lisp/quicklisp
:/mnt/e/library/lisp/quicklisp$ sbcl --load quicklisp.lisp
This is SBCL 1.3.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.

===== quicklisp quickstart 2015-01-28 loaded =====

To continue with installation, evaluate: (quicklisp-quickstart:install)

For installation options, evaluate: (quicklisp-quickstart:help)

* (quicklisp-quickstart:help)

===== quicklisp quickstart install help =====
```

图 4.8: Install quicklisp on ubuntu

Install quicklisp on ubuntu

(续上页)

I will append the following lines to `#P"/home/liu/.clisprc.lisp"`:

```
;;; The following lines added by ql:add-to-init-file:
 #-quicklisp
(let ((quicklisp-init
(merge-pathnames "/mnt/e/sfw/lisp/lib/quicklisp/setup.lisp"
(user-homedir-pathname))))
  (when (probe-file quicklisp-init)
    (load quicklisp-init)))
```

Press Enter to **continue**.

4.2.7.1.4.4 使用

提示: quicklisp 用法

- To load/install a system, use: (ql:quickload "system-name")
- To remove a system, use: (ql:uninstall "system-name")
- To find systems, use: (ql:system-apropos "term")
- To get updated software, use: (ql:update-dist "quicklisp")
- To update the Quicklisp client, use: (ql:update-client)
- To see what systems depend on a particular system, use: (ql:who-depends-on "system-name")
- To load Quicklisp every time you start Lisp, use: (ql:add-to-init-file)

- For more information, see <http://www.quicklisp.org/beta/>

以安装 [lisp-binary](https://github.com/j3pic/lisp-binary) (<https://github.com/j3pic/lisp-binary>) 进入 Lisp 交互环境，执行 (ql:quickload "lisp-binary") 命令即可安装，安装日志如下

(下页继续)

(续上页)

```
[package ffi-features].....  

[package lisp-binary-utils].....  

[package lisp-binary/integer].....  

[package lisp-binary/float].....  

[package simple-bit-stream].....  

[package reverse-stream].....  

[package lisp-binary].....  

.....  

("lisp-binary")  

*
```

4.2.7.1.5 生成可独立执行文件

Lisp 是脚本语言, 具备跨平台的特性. Ecl 是一个开源的 Lisp 语言实现, 可以将 lisp 文件编译成独立可执行程序.

3. 编译 CL-USER> (setf c::delete-files nil) CL-USER> (compile-file "test.lisp" :system-p t) CL-USER> (c:build-program "test" :lisp-files `(("test.o")))

4. 执行 #./test

5.github demo:<https://github.com/cwndrws/lisp-c-example>

4.2.8 PyTorch 简明教程

4.2.8.1 从这里开始

4.2.8.1.1 安装

4.2.8.1.1.1 在线安装

参考 [START LOCALLY](https://pytorch.org/get-started/locally/) (<https://pytorch.org/get-started/locally/>) , 选择环境, 执行安装命令即可.

```
conda install pytorch torchvision cudatoolkit=9.0 -c pytorch
```

4.2.8.1.1.2 本地安装包安装

4.2.8.1.1.3 Anaconda

```
conda install --use-local ./pytorch-1.1.0-py3.7_cuda9.0.176_cudnn7.5.1_0.tar.bz2 -c pytorch conda install torchvision cudatoolkit=9.0 -c pytorch
```

4.2.8.1.1.4 Python

从 [pypi](https://pypi.org/) (<https://pypi.org/>) 下载对应版本的安装包, 执行 pip install 命令安装即可, 如

```
sudo pip3 install torch-1.1.0-cp35-cp35m-manylinux1_x86_64.whl  
sudo pip3 install torchvision
```

4.2.8.1.1.5 本地源码安装

4.2.8.1.1.6 Pytorch

Pytorch 的源码安装并不复杂, 与其它深度学习平台的源码安装过程相比, 极为简单, 具体可参考 [pytorch from source](https://github.com/pytorch/pytorch#from-source) (<https://github.com/pytorch/pytorch#from-source>).

代码 4.8: Pytorch 源码安装

```
1 conda install numpy pyyaml mkl mkl-include setuptools cmake cffi typing  
2  
3 # Add LAPACK support for the GPU if needed  
4 conda install -c pytorch magma-cuda90 # or [magma-cuda80 / magma-cuda91] depending  
5 # on your cuda version  
6  
7 git clone --recursive https://github.com/pytorch/pytorch  
8 cd pytorch  
9  
10 export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-$dirname $(which conda)}/.."  
11 python setup.py install
```

4.2.8.1.1.7 Torchvision

torchvision 的源码安装也非常简单, 进入 torchvision 根目录, 执行 python setup.py install 即可, 具体可参考 [torchvision from source](https://github.com/pytorch/vision) (<https://github.com/pytorch/vision>).

4.2.8.1.2 构建本地文档

4.2.8.1.2.1 API 手册

4.2.8.1.2.2 Pytorch

Sphinx 格式文档, 执行如下命令即可生成 html 与 epub 格式的文档

代码 4.9: 构建 Pytorch 文档

```
1 cd docs/  
2 pip install -r requirements.txt
```

(下页继续)

(续上页)

```

3
4 make html
5 make epub

```

4.2.8.1.2.3 Torchvision

Sphinx 格式文档, 执行如下命令即可生成 html 与 epub 格式的文档

代码 4.10: 构建 Torchvision 文档

```

1 cd docs/
2 pip install -r requirements.txt
3
4 make html
5 make epub

```

4.2.8.1.3 问题与解决

4.2.8.1.3.1 安装

4.2.8.1.3.2 PyQt

代码 4.11: 构建 Torchvision 文档

```

1 ImportError: anaconda3/lib/python3.6/site-packages/PyQt5/../../libQt5Core.so.5:_
  ↪version `Qt_5.9' not found (required by /home/liu/anaconda3/lib/python3.6/site-
  ↪packages/PyQt5/QtCore.so)

```

matplotlib 绘图使用 PyQt, 需要安装该库, 通过 pip install PyQt 安装即可.

4.2.8.1.3.3 使用

4.2.8.1.3.4 Torch 导入错误 1

无论是通过 Anaconda 还是 pip 安装完毕 PyTorch 后, 在 Python 解释器中导入 torch(import torch) 均会报出如下错误:

```

from torch._C import * ImportError: libcurand.so.8.0: cannot open shared object_
  ↪file

```

系统为 Ubuntu16.04LTS, Python3.5, Python3.7(Anaconda), CUDA9.0 和 CUDA8.0 共存. 安装前环境已经切换为 CUDA9.0, 上述错误提示找不到 CUDA8.0 相关文件, 因而怀疑一些编译的库使用的是 CUDA8.0. 最终发现是之前安装的 caffe2 的影响, 导致 PyTorch 中的 caffe2 不能正确安装, 卸载之前安装的 caffe2, 问题解决.

4.2.8.1.3.5 Torch 导入错误 2

安装没有问题, 导入时提示如下错误信息

在另一个不含 torch 的目录使用即可解决该问题.

4.2.8.1.3.6 Torchvision 导入失败

可能没安装或者安装失败, 请注意环境, 如果是 conda 环境, 要在相应环境下执行, 如果不是, 要退出 conda 环境。

4.2.8.2 数据集制作与加载

借助 Torch 提供的 DataSet 和 DataLoader 类实现自定义数据集的制作与加载.

4.2.8.2.1 DataSet

数据集有以下几个类

- Dataset (): 无输入参数
- TensorDataset (*tensors): 将多个 “Tensor” 做成一个数据集, 输入为多个 “Tensors”
- ConcatDataset (datasets): 将多个数据集拼成一个数据集, 输入为多个数据集
- Subset (dataset, indices): 取数据集的子集, 输入为数据集与索引

4.2.8.2.1.1 使用 Dataset

需要重写该类的 `__len__`, `__getitem__`, `__init__` 方法. 下面举例说明, 假设网络含两个输入 x_1, x_2 , 一个输出 y , 构造 MyDataset 类, 实现代码为

代码 4.12: demo_DatasetDataLoader.py

```
1
2 import torch as th
3 from torch.utils.data import Dataset, DataLoader, TensorDataset
4 from torch.autograd import Variable
5
6
7 # ===data
8
9 x1 = th.randn((10, 3, 128, 128))
10 x2 = th.randn((10, 2, 128, 128))
11 y = th.randn((10, 128, 128))
12
13 epochs = 4
```

(下页继续)

(续上页)

```

15 # ===DataSet
16
17 print("----Dataset")
18
19 class MyDataset(Dataset):
20
21     def __init__(self, x1, x2, y):
22         self.x1 = x1
23         self.x2 = x2
24         self.y = y
25         self.len = y.shape[0]
26
27
28     def __getitem__(self, index):
29         return self.x1[index], self.x2[index], self.y[index]
30
31     def __len__(self):
32         return self.len
33
34 mydataset = MyDataset(x1, x2, y)
35
36 dataloader = DataLoader(dataset=mydataset,
37                         batch_size=3, shuffle=True, num_workers=2)
38
39 for epoch in range(epochs):
40     print(epoch)
41     for i, data in enumerate(dataloader):
42         x1v, x2v, yv = data
43         print(x1v.size(), x2v.size(), yv.size())
44
45

```

4.2.8.2.1.2 使用 TensorDataset

TensorDataset 类的使用非常简单, 省去了重写类方法的麻烦, 下面举例说明, 假设网络含两个输入 x_1, x_2 , 一个输出 y , 构造 MyDataset 类, 实现代码为

代码 4.13: demo_TensorDatasetDataLoader.py

```

1 import torch as th
2 from torch.utils.data import Dataset, DataLoader, TensorDataset
3 from torch.autograd import Variable
4
5
6 # ===data
7

```

(下页继续)

(续上页)

```

8 x1 = th.randn((10, 3, 128, 128))
9 x2 = th.randn((10, 2, 128, 128))
10 y = th.randn((10, 128, 128))
11
12 epochs = 4
13
14 # ===TensorDataSet
15
16 print("----TensorDataset")
17 mydataset = TensorDataset(x1, x2, y)
18
19 dataloader = DataLoader(dataset=mydataset,
20                         batch_size=3, shuffle=True, num_workers=2)
21
22 # epoch = 0
23 for epoch in range(epochs):
24     print(epoch)
25     for i, data in enumerate(dataloader):
26         x1v, x2v, yv = data
27         print(x1v.size(), x2v.size(), yv.size())

```

4.2.8.2.2 DataLoader

4.2.9 Shell 程序设计

4.2.9.1 bash 命令

4.2.9.1.1 常用命令

4.2.9.1.1.1 tee

- 只输出到屏幕: python main.py
- 只输出到文件 (*log.log*): python main.py 2>&1 | tee >*log.log* 或 python main.py | tee >*log.log*
- 输出到屏幕和文件 (*log.log*): python main.py | tee *log.log*

4.2.9.2 ssh 协议

4.2.9.2.1 常用命令

- 远程登录: ssh *username@serverip*
- 远程拷贝文件: ssh *username@serverip:remotefilepath localfilepath*
- 远程拷贝文件夹: ssh -r *username@serverip:remoteFolderPath localFolderPath*

提示: 多机拷贝假设 A, B, C 等多台机器处于同一局域网, 可以使用上述命令共享数据.

4.3 算法设计

4.3.1 区域填充

4.3.1.1 圆形区域填充

4.3.1.1.1 问题分析

4.3.1.1.2 填充算法

4.3.1.1.2.1 圆面方程

$$\begin{aligned}x &= x_{c_1} + r\cos\theta \\y &= y_{c_2} + r\sin\theta\end{aligned}$$

4.3.1.2 凸多边形区域填充

4.3.1.2.1 问题分析

4.3.1.2.2 填充算法

4.3.1.2.2.1 扫描线法

4.3.1.2.2.2 夹角和法

内部点与各顶点夹角和为 360 度.

4.3.1.2.2.3 多边形方程法

该方法的核心思想是 多边形内的点满足多边形区域的边构成的不等式方程组, 设有由 N 个有序顶点 $\mathbb{P} = \{(x_i, y_i)^T\}_{i=1}^N$ 按顺序 (逆时针或顺时针) 构成的多边形区域 \mathcal{P} .

若按逆时针方向, 则多边形内的区域可以表示为不等式方程组

$$\begin{aligned}l_1 : a_1x + b_1y + c_1 &\geq 0 \\l_2 : a_2x + b_2y + c_2 &\geq 0 \\&\vdots \\l_N : a_Nx + b_Ny + c_N &\geq 0\end{aligned}\tag{4.1}$$

若按顺时针方向, 则多边形内的区域可以表示为不等式方程组

$$\begin{aligned} l_1 &: a_1x + b_1y + c_1 \leq 0 \\ l_2 &: a_2x + b_2y + c_2 \leq 0 \\ &\vdots \\ l_N &: a_Nx + b_Ny + c_N \leq 0 \end{aligned} \quad (4.2)$$

其中, $a_n = y_n - y_{n+1}$, $b_n = x_{n+1} - x_n$, $c_n = y_{n+1}(x_n - x_{n+1}) - x_{n+1}(y_n - y_{n+1})$, $n = 1, 2, \dots, N$, 且 $x_{N+1} = x_1$, $y_{N+1} = y_1$.

综上可知, 凸多边形区域内的点对应的凸多边形不等式方程组中的不等式同号.

故若要判断一点 $P_0 = (x_0, y_0)^T$ 是否在某凸多边形区域内, 只需要将改点坐标代入式 4.3, 判断是否同号即可. 若满足则待填充点在多边形区域内, 反之不在.

$$\begin{aligned} l_1 &: a_1x + b_1y + c_1 \\ l_2 &: a_2x + b_2y + c_2 \\ &\vdots \\ l_N &: a_Nx + b_Ny + c_N \end{aligned} \quad (4.3)$$

4.3.1.2.3 实验与分析

4.3.1.2.3.1 实验 1

4.3.1.2.3.2 实验设置

假如对由有序顶点集合 $\{(0, 0), (200, 0), (200, 100), (0, 100)\}$ 确定的方形区域 \mathcal{R} 内的凸多边形区域 \mathcal{P} 进行区域填充, 其中, \mathcal{P} 由有序顶点集合 $\{(50, 50), (40, 40), (42, 25), (55, 5), (65, 20), (70, 40), (60, 50)\}$ 确定. 记水平与垂直方向上的离散化尺度分别为 d_x, d_y , 分析对比不同尺度下, 各种凸多边形填充算法的执行效率与填充效果.

4.3.1.2.3.3 实验代码

实验中用到的代码可在 [这里](https://iridescent.ink/mattools/) (<https://iridescent.ink/mattools/>) 和 [这里](https://iridescent.ink/mpathplanning/) (<https://iridescent.ink/mpathplanning/>) 下载, 主程序代码见 代码 4.14

代码 4.14: demo_ConvexPolygonFilling.m

```

1 clear all
2 close all
3 clc
4
5 H = 100;
6 W = 200;
% dy = 0.1;
% dx = 0.1;
```

(下页继续)

(续上页)

```

9  % dy = 0.5;
10 % dx = 0.5;
11 dy = 1;
12 dx = 1;
13
14 Polygon = [50 50; 40 40; 42 25; 55 5; 65 20; 70 40; 60 50]; % Anti Clockwise
15 % Polygon = [60 50; 70 40; 65 20; 55 5; 42 25; 40 40; 50 50]; % Clockwise
16 % Polygon = [50 50; 40 40; 55 5; 70 40; 58 42];
17
18 [y, x] = meshgrid(1:dy:H, 1:dx:W);
19 y = y(:);
20 x = x(:);
21 Points = [x y];
22
23 tic;
24 s1 = isincvxplg(Points, Polygon, 'Angle');
25 toc;
26
27 tic;
28 s2 = isincvxplg(Points, Polygon, 'PolygonEquations');
29 toc;
30
31 figure
32 hold on
33 subplot(121)
34 cpplot(Polygon, '-r', 'linewidth', 2)
35 opplot(Points(s1, :), '*r', 'linewidth', 1)
36
37 subplot(122)
38 cpplot(Polygon, '-r', 'linewidth', 2)
39 opplot(Points(s2, :), '*r', 'linewidth', 1)

```

凸多边形区域填充算法核心代码如 代码 4.15 所示:

代码 4.15: isincvxplg.m

```

1 function [ s ] = isincvxplg( Points, Polygon, Method )
2 %ISINCVXPLG judges whether a point is in a convex polygon area
3 %
4 % s = ISINCVXPLG( Points, Polygon ) returns the the status (in Polygon --> 1,
5 % not in Polygon --> 0) of each point in Points.
6 %
7 % s = ISINCVXPLG( Points, Polygon, Method ) returns the the status (in Polygon -->_
8 %> 1,
9 % not in Polygon --> 0) of each point in Points using method 'Method'.
10 % Points: N-2 array, [x y;x y;...]
11 % Polygon: L-2 array, [x y;x y;...]

```

(下页继续)

(续上页)

```

11 % Method: 'Angle' (default), 'PolygonEquations'
12 %
13 % Examples
14 % -----
15 %
16 % Points = [1 1; 2 2;3 3];
17 % Polygon = [1 2;1.5 1;2.5 1;3 2;2 3];
18 % cpplot(Polygon, '-r', 'linewidth', 2)
19 % hold on
20 % opplot(Points, '*r', 'linewidth', 1)
21 % s = isinpolygon(Points, Polygon)
22 % s =
23 %
24 %     0
25 %     1
26 %     0
27 % s = isinpolygon(Points, Polygon, 'PolygonEquations')
28 %
29 %
30 % See also isincircle.
31 %
32 % Copyright 2019-2030 Zhi Liu, https://iridescent.ink/.
33 %

34
35 if nargin < 3
    Method = 'Angle';
36 end
37 EPS = 1e-6;
38 nPoints = size(Points, 1);
39 nVertex = size(Polygon, 1);
40 PI2 = pi + pi;
41 Polygon = Polygon(:, 1:2);
42 s = false(nPoints, 1);

43
44 if strcmp(Method, 'Angle')
45     for n = 1:nPoints
46         Pg = bsxfun(@minus, Polygon, Points(n, :));
47         Pa = Pg(1:nVertex, :);
48         Pb = [Pg(2:nVertex, :); Pg(1, :)];
49         A = angle2(Pa, Pb, 0);
50         A = abs(A);
51         sumA = sum(A, 1);
52         if abs(sumA - PI2) < EPS
53             s(n) = 1;
54         else
55             s(n) = 0;
56         end
57     end

```

(下页继续)

(续上页)

```

58 end
59 end
60
61 if strcmp(Method, 'PolygonEquations')
62 Q = mean(Polygon, 1);
63 Ls = zeros(nVertex, 4); % edges
64 for nv = 1:nVertex-1
65 Ls(nv, :) = [Polygon(nv, :), Polygon(nv+1, :)];
66 end
67 Ls(end, :) = [Polygon(end, :), Polygon(1, :)];
68
69 A = Ls(:, 2) - Ls(:, 4);
70 B = Ls(:, 3) - Ls(:, 1);
71 C = Ls(:, 4) .* (Ls(:, 1) - Ls(:, 3)) - Ls(:, 3) .* (Ls(:, 2) - Ls(:, 4));
72
73 for n = 1:nPoints
74 fp = A * Points(n, 1) + B * Points(n, 2) + C;
75 s(n) = (sum((fp >= 0)) == nVertex) || (sum((fp > 0)) == 0);
76 end
77 end
78
79 end

```

4.3.1.2.3.4 实验结果

我们在 MatlabR2019 上实现上述算法, 硬件平台为 Intel® Xeon(R) CPU E5-2696 v3 @ 2.30GHz × 36, 实验中仅使用单线程, 运行时间如 表 4.2 所示, 填充效果如 图 4.9 所示. 对比表格与填充效果图可知, 多边形方程法计算复杂度更低, 耗时极短, 效率更高.

表 4.2: 不同填充算法耗时对比

方法	$d_x = d_y = 1$	$d_x = d_y = 0.5$	$d_x = d_y = 0.1$
夹角和法	0.758408s	2.397315s	57.775898s
多边形方程法	0.020063s	0.056518s	0.943451s

4.3.1.3 凹多边形区域填充

4.3.1.3.1 问题分析

该方法的核心思想是根据 **多边形内的点满足多边形区域的边构成的不等式方程组**的思想, 即找到凸多边形区域内一点 (如重心 $P_c = (x_c, y_c)^T$), 判断改点确定的多边形的边构成的不等式方程组的符号, 然后对于待填充点 $P_0 = (x_0, y_0)^T$, 代入方程组并判断符号是否与 P_c 一致, 若一致则在多边形区域内. 设有由 N 个

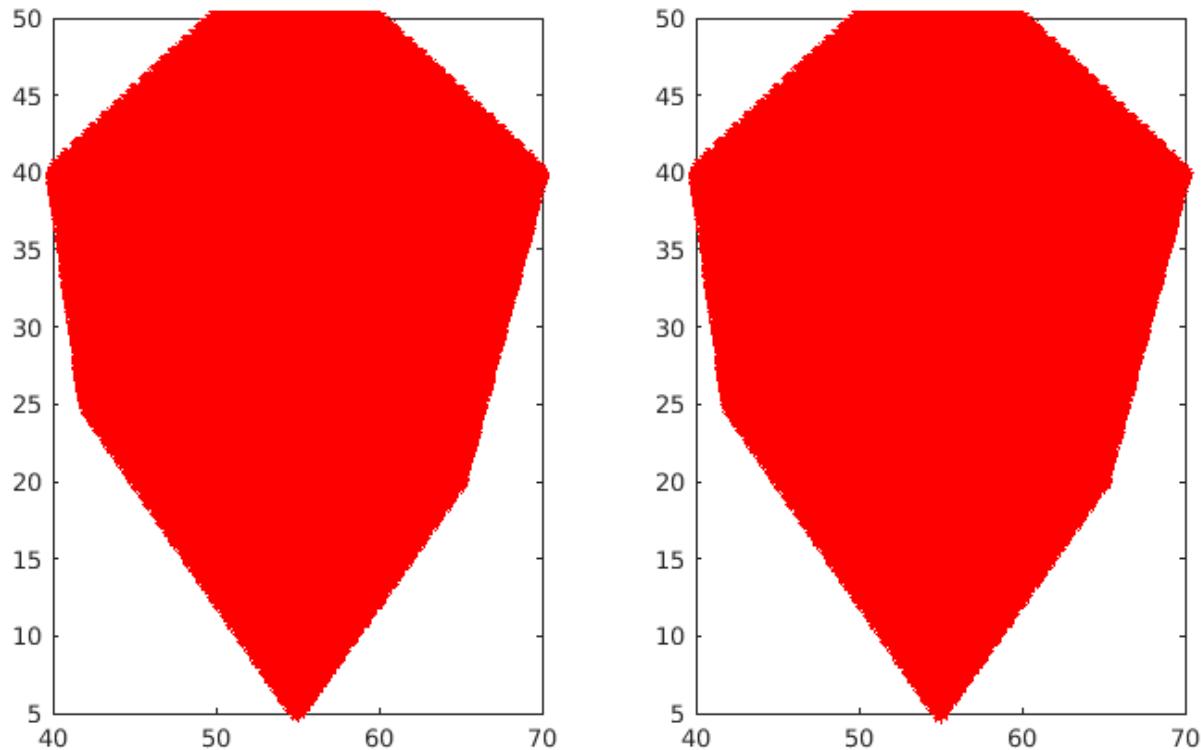


图 4.9: 凸多边形区域填充结果对比图. 夹角和法 (左), 多边形方程法 (右)

顶点 $\mathbb{V} = (x_i, y_i)^T_{i=1}^N$ 按顺序构成的多边形 \mathcal{P} , 将其边构成的方程组记为

$$\begin{aligned} l_1 : & a_1x + b_1y + c_1 = 0 \\ l_2 : & a_2x + b_2y + c_2 = 0 \\ & \vdots \\ l_{N-1} : & a_{N-1}x + b_{N-1}y + c_{N-1} = 0 \end{aligned} \quad (4.4)$$

将重心 P_c 的坐标代入方程组式 4.4 并判断符号, 记为 $\mathbf{f}_c = (f_{c_1}, f_{c_2}, \dots, f_{c_{N-1}})^T$, 其中大于等于 0 记为 1, 小于 0 记为 0.

对于待填充点 $P_0 = (x_0, y_0)^T$, 代入方程组式 4.4 并判断符号, 记为 $\mathbf{f}_0 = (f_{0_1}, f_{0_2}, \dots, f_{0_{N-1}})^T$, 其中大于等于 0 记为 1, 小于 0 记为 0.

若符号向量 \mathbf{f}_c 与 \mathbf{f}_0 的值相同, 则待填充点在多边形区域内, 反之则不在区域内.

4.4 互联部分

4.4.1 网站搭建

4.5 人工智能处理器

4.5.1 基础篇

4.5.1.1 引言

4.5.1.1.1 什么是人工智能芯片

人工智能 (Artificial Intelligence, AI) 芯片: 从广义上讲只要能够运行人工智能算法的芯片都叫作 AI 芯片. 但是通常意义上的 AI 芯片指的是针对人工智能算法做了特殊加速设计的芯片, 现阶段, 这些人工智能算法一般以深度学习算法为主, 也可以包括其它机器学习算法.

4.5.1.1.2 AI 芯片分类

AI 芯片按计算架构分别可以分为: 图形处理单元 (GPU), 场可编程阵列 (FPGA), 专用集成电路 (ASIC) 与脑启发 (Brain-Inspired) 式; 按功能可分为训练型与推理性; 按应用场景可分为服务端 (云端) 和移动端 (终端).

不同类型 AI 芯片比较

类型	GPU	FPGA	ASIC	脑启发
定制化	通用	半定制化	全定制化	通用
优点	通用性强	功耗低	专用性强, 功耗低	认知能力强, 效率高, 功耗极低
缺点	功耗高	开发难	开发较难	开发很难
适用算法	大规模并行运算	大规模并行运算	深度学习算法	通用智能计算
适用场景	云端	云端, 终端	云端, 终端	云端, 终端
代表厂商及产品	NVIDIA Tesla	Xilinx Versal	Google TPU; 寒武纪: 1H8, 1M;	IBM: TrueNorth
技术成熟度	高	中上	中, 演进阶段	低, 探索阶段

4.5.1.1.3 AI 芯片选型

需要根据项目需求，确认 AI 芯片的需求，AI 芯片的需求分析需要考虑应用场景（云端/终端）、芯片性能指标（算力，功耗等）、芯片厂商（国产/非国产）、芯片价格、技术成熟度、研发成本（周期、语言等）等多种因素，只有详细分析 AI 芯片需求，才能确保所选 AI 芯片满足项目要求。

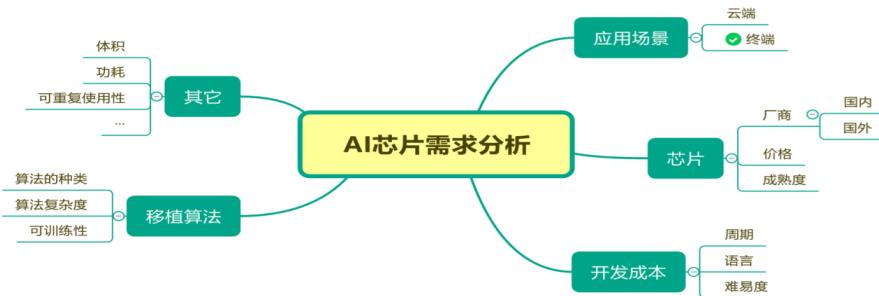


图 4.10: AI 芯片需求分析

4.5.2 实战篇

4.5.2.1 Atalas200DK 简明教程

4.5.2.1.1 简介

4.5.2.1.1.1 硬件配置

Atalas200DK 是华为开发的一款基于升腾 AI 处理器 Ascend 310 的开发套件，功耗低体积小，适合于移动嵌入式场景应用。下表 4.3 给出了华为 Atalas200DK 与比特大陆 Mini Box SE3 硬件配置对比，可知华为 Atalas200DK 开发套件性能要优于比特大陆 Mini Box SE3。

表 4.3: 华为 Atlas200DK 与比特大陆 Mini Box SE3 硬件配置对比

参数	华为 Atlas200DK	比特大陆 Mini Box SE3
芯 片 型 号	华为 Ascend 310	比特大陆 BM1682
AI 算 力	8TFLOPS@FP16, 支持 16TOPS、8TOPS、4TOPS 三种算力配 置	3TFLOPS@FP32
内 存	LPDDR4x, 8GB, 3200Mbps	8GB
存 储	1 个 Micro SD 卡, 支持 SD3.0, 最高支持速率 SDR50, 最大容量 2TB	EMMC 32GB, 支持 SD 扩 展
网 络 接 口	1 个 GE RJ45	1 x Gigabit
U S B 接 口	1 个 USB3.0 Type C 接口, 仅作为从设备, 兼容 USB2.0	—
其 它 接 口	1 个 40pin IO 连接器, 2 个 22pin MIPI 连接器, 2 个板载麦克风	—
电 源	5V~28V DC, 默认配置 12V 3A 适配器	12V DC
结 构 尺 寸	137.8mm x 93.0mm x 32.9mm	210mm * 115mm * 45mm
功 耗	20W	平均 60W, 最大 90W
重 量	234g	—
工 作 温 度	0°C~ 35°C	-10°C ~ 55°C

Atlas200DK 主要包含 Ascend 310 AI 加速模块、图像/音频接口芯片 (Hi3559C) 和 LAN Switch 三部分，系 统架构如 图 4.11 所示。

4.5.2.1.1.2 软件配置

4.5.2.1.1.3 资料汇总

- [文档中心](https://ascend.huawei.com/documentation) (<https://ascend.huawei.com/documentation>) : Atlas200DK
- [软件仓库 tools](https://github.com/Ascend/tools/) (<https://github.com/Ascend/tools/>) : 含制卡工具, MindSpore Studio, DDK 等, 注, 以下涉及的软件请换成最新版本 1.3
- [模型仓库 models](https://github.com/Ascend/models/) (<https://github.com/Ascend/models/>) : 含分类与目标检测模型
- [操作系统获取路径](http://old-releases.ubuntu.com/releases/16.04.3/) (<http://old-releases.ubuntu.com/releases/16.04.3/>) : Ubuntu 16.04.3

注解: 目前开源的 Ascend 芯片上的模型有:

- 分类: alexnet, caffenet, car_color, car_plate_recognition, car_type, densenet, dpn98, face_attribute, face_emotion, feature_extract, googlenet, inception_age, inception_gender, inception_v2, inception_v3, inception_v4, mobilenet_v1, mobilenet_v2, pedestrian, resnet101, resnet152, resnet18, resnet50, sphereface, squeezezenet, vanillacnn, vgg16, vgg19

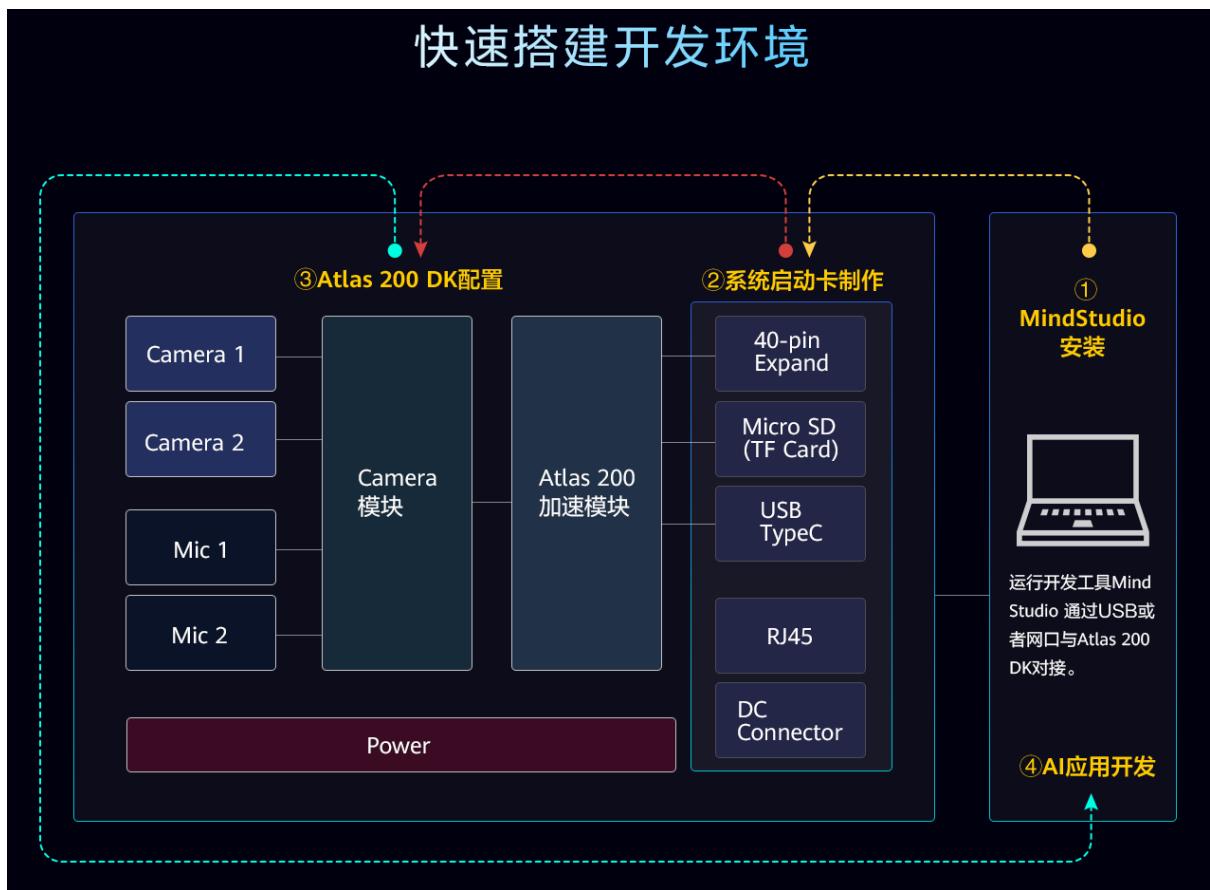


图 4.11: 华为 Atlas200DK 系统架构

华为 Atlas200DK 系统架构

- 检测: car_plate_detection, face_body_detection, face_detection, faster_rcnn, faster_rcnn_vgg16, mobilent_ssdl, occlusion_face_detection, peppapig_detection, resnet_ssd, vgg16_ssd, vgg_ssd, vgg_ssd_voc0712, yolo_v2

4.5.2.1.2 环境配置

4.5.2.1.2.1 准备

4.5.2.1.2.2 编译依赖

通过如下命令安装编译依赖

代码 4.16: 编译依赖安装

```
1 sudo apt-get install gcc g++ cmake curl libboost-all-dev unzip haveged libatlas-
  ↪base-dev python-skimage python3-skimage python-pip python3-pip liblmdbs-dev
  ↪libhdf5-serial-dev libsnappy-dev libleveldb-dev make graphviz autoconf libxml2-
  ↪dev libxml2 sqlite3 python libzip-dev libssl-dev
```

4.5.2.1.2.3 Java

通过如下命令安装 Java JDK 环境

代码 4.17: 编译依赖安装

```
1 # 添加源
2 sudo add-apt-repository ppa:openjdk-r/ppa
3 sudo apt-get update
4 # 安装 java jdk
5 sudo apt-get install -y openjdk-8-jdk
```

安装完成后, 添加如下环境变量到 `~/.bashrc` 文件即完成 Java 的安装

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
```

4.5.2.1.2.4 MindSpore Studio 安装

以下假设工作主目录为 `Atlas200`, 含如下两个文件

- mini_mind_studio_Ubuntu.rar*
- MSpore_DDK-1.3.T34.B891-x86_64.ubuntu16.04-aarch64.ubuntu16.04-aarch64.ubuntu16.04.tar.gz*

解压 `mini_mind_studio_Ubuntu.rar` 软件包, 并将压缩包 `MSpore_DDK-1.3.T34.B891-x86_64.ubuntu16.04-aarch64.ubuntu16.04-aarch64.ubuntu16.04.tar.gz` 放进解压目录 `mini_mind_studio_Ubuntu`.

4.5.2.1.2.5 准备工作

1. 配置可选项

- 打开文件 `scripts/env.conf`
- 设置 IP, 端口等变量值, 安装用户, 举例如下

代码 4.18: env.conf 配置

```
1 ip=192.168.31.233
2 port=8087
3 toolpath=/yourdir/Atlas200/tools
4 backup=/yourdir/Atlas200/wsbackup
5 profiler_port=8099
6 max_log_size=10
7 apache_user=msvpUser
8 install_user=ant
9 package_path=/usr/bin
10 use_eth0=true
11 load_data=true
```

4.5.2.1.2.6 执行安装

执行如下命令安装

代码 4.19: install mind studio

```
1 bee@Smart:~/sfw/huawei/mini_mind_studio_Ubuntu$ sudo ./add_sudo.sh bee
2 [sudo] password for bee:
3 bee@Smart:~/sfw/huawei/mini_mind_studio_Ubuntu$ ./install.sh
4 files check pass!
5 [INFO] Please make sure that you have configured in env.conf otherwise the ↵
       ↵installation will use default configuration.
6 [INFO] Your ip address is 192.168.49.189
7 [INFO] Using port: 8888
8 [INFO] Using tool_path: /home/bee/tools
9 [INFO] Using backup_path: /home/bee/wsbackup
10 [INFO] Using profiler_port: 8099
11 [INFO] Using apache_user: msvpUser
12 [INFO] Mind-Studio package found: Mind-Studio_Ubuntu-x86-64.tar
13 [INFO] DDK package found: MSpole_DDK-1.3.T34.B891-x86_64.ubuntu16.04-aarch64. ↵
       ↵ubuntu16.04-aarch64.ubuntu16.04.tar.gz
14 [INFO] Extracting package, please wait for a minute...
15 [INFO] Start installing Mind-Studio
16 ##### The current system is ubuntu 16.04.
17 install user: bee
18 Accept use_eth0 successfully!
```

(下页继续)

(续上页)

```

19 Accept load_data successfully!
20 Accept run_type successfully!
21 start check profiling pre-install software
22 四 29 8 月 2019 20:47:49 [INFO] [MSVP] [22906] install_profiling.sh: Current user
23 →is bee.
24 ====> Start install mongodb
25 =====> Successfully installed the mongod.
26 =====> Started to prepare conf.
27 =====> Successfully prepared conf.
28 =====> Started to prepare rebootshell.
29 =====> Successfully prepared rebootshell.
30 /etc/init.d
31 =====> Started to prepare envtools
32 =====> Started to import gnupg key.
33 =====> Successfully imported gnupg key
34 =====> Successfully prepared envtools
35 =====> Copy libCaffe
36 /home/bee/sfw/huawei/mini_mind_studio_Ubuntu/Mind-Studio/vendor/envtools
37 =====> Copy ArmPython
38 =====> Started to prepare Mind_Studio_Project.
39 openjdk version "1.8.0_222"
40 OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~16.04.1-b10)
41 OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
42 =====> Successfully prepared Mind_Studio_Project
43 =====> No projects need to be loaded.
44 =====> No my-datasets need to be loaded.
45 =====> No my-model needs to be loaded.
46 =====> No caffe-model need to load
47 =====> No db needs to be loaded.
48 =====> No offline-model needs to be loaded.
49 =====> Successfully installed the environment
50 Your time zone 20:49:01 is not a common time zone, Please check if the jvm virtual
51 →machine time zone matches.
52 =====> Installing the DDK...
53 MSpore_DDK-1.3.T34.B891-x86_64.ubuntu16.04-aarch64.ubuntu16.04-aarch64.ubuntu16.04.
54 →tar.gz
55 =====> Starting to extract ddk.
56 =====> Installing the DDK...
57 =====> Successfully installed the DDK.
58 =====> Write DDKConfig successfully!
59 checking the JCE in JDK ...
60 =====> Remove DDKConfig successfully!
61 =====> Write DDKConfig successfully!
62 start to check certificate expire days...
63 ==> Certificate /home/bee/tools/conf/secure_keys/server.crt will sleep 365d!
64 set ulimit -n 64000
65 ./logs/

```

(下页继续)

(续上页)

```
63 ./logs/operation.log
64 ./rootkey/
65 ./rootkey/v1/
66 ./rootkey/v1/cat/
67 ./rootkey/v1/cat/c.txt
68 ./rootkey/v1/apple/
69 ./rootkey/v1/apple/a.txt
70 ./rootkey/v1/dog/
71 ./rootkey/v1/dog/d.txt
72 ./rootkey/v1/boy/
73 ./rootkey/v1/boy/b.txt
74 ./work_key.json
75 about to fork child process, waiting until server is ready for connections.
76 forked process: 24944
77 child process started successfully, parent exiting
78 ==> Successfully started the mongod server.
79 ==> mongo password has been initialised, please change password in time. you can
     ↵refer to instruction manual for help
80 Total number of files is 0
81 ==> Starting Mind Studio...Please wait.
82 Server startup in 19688 ms
83 /home/bee/tools/log
84 ==> Successfully started Mind Studio.
85 ==> Use Google Chrome https://IP:Port
86 ==> You can check /home/bee/tools/log/mind_log/mind-20190829204746.log for
     ↵installation details!
87 the install path is not in PATH, add it to PATH.
88 ==> Waiting for profiling to finish...
89 ==> Successfully installed profiling.
90 Starting to chmod folders.
91 Installation finished.
92 [INFO] delete temporary Mind-Studio directory.
93 /home/bee/sfw/huawei/mini_mind_studio_Ubuntu
94 [INFO] Please make sure that the Linux Kernel Version is higher than 4.18 or apply
     ↵this patch https://bugzilla.kernel.org/attachment.cgi?id=277305 when Linux
     ↵Kernel Version below 4.18, otherwise maybe stuck when converting model using omg
     ↵with TE plugin.
95 [INFO] Install successfully
```

提示: 若安装过程提示 mongo 安装启动失败, 可以通过 sudo apt install mongodb-server 直接安装.

安装完成后在浏览器中输入上述 <https://IP:port> 地址, 如可以进入登录界面(图 4.12), 则证明安装成功, 否则重复上述步骤

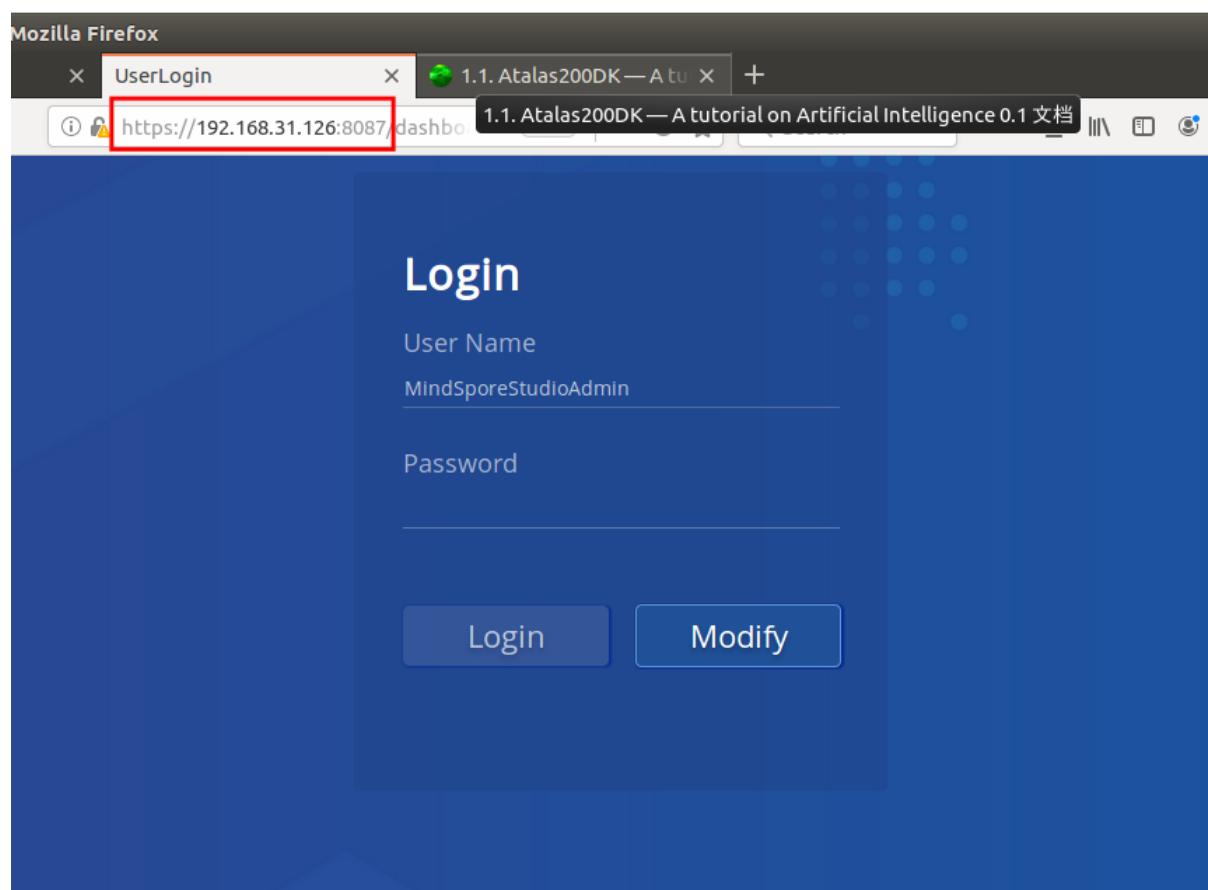


图 4.12: MindSporeStudio 登录界面

提示: 登录 MindSpore Studio 界面的用户名默认为“MindSporeStudioAdmin”，不支持修改和新建。初始密码为“Huawei123@”，请参见修改密码。登录 Profiling 界面的用户名为 mspadmin，初始密码为 Admin12#\$，请参见展示性能分析数据章节创建普通用户。

登录成功后即可进入 Mind Spore Studio 工作界面(图 4.13)

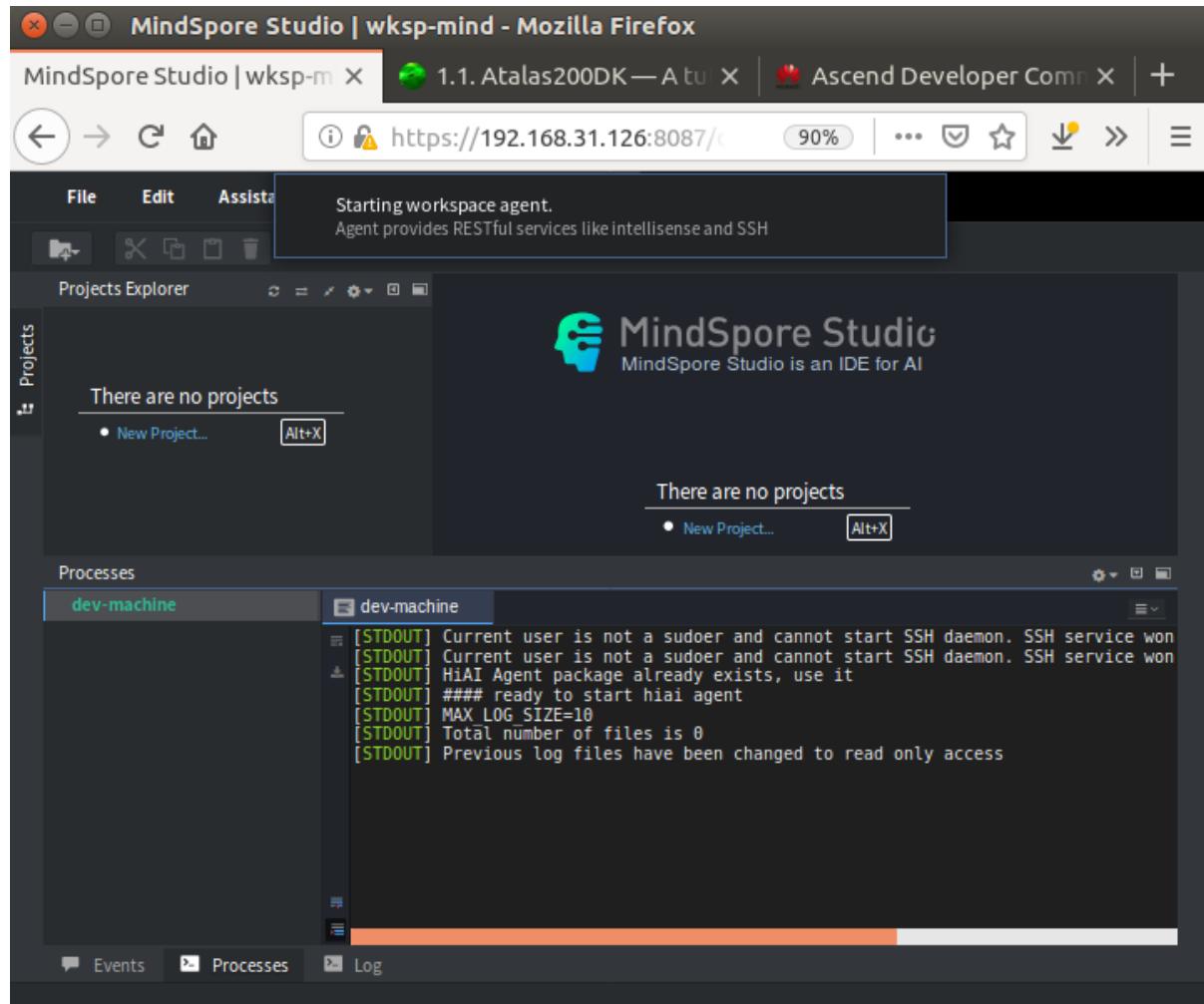


图 4.13: MindSporeStudio 工作界面

4.5.2.1.3 系统启动

4.5.2.1.3.1 SD 卡启动镜像制作

4.5.2.1.3.2 准备工作

- 硬件环境

- 一张 MicroSD 卡, 大于 16GB
- 一台装有 PC 机

• 软件环境

- PC 机装有 ubuntu-16.04.3-desktop/server-amd64, MindSpore Studio, DDK
- 嵌入式操作系统 ubuntu-16.04.3-server-arm64.iso, 可通过命令 `axel -n 10 -o ./ http://old-releases.ubuntu.com/releases/16.04.3/ubuntu-16.04.3-server-arm64.iso` 下载
- 开发套件 mini_developkit-xxx.rar
- 脚本文件 `make_sd_card.py` 和 `make_ubuntu_sd.sh`, 可从 [GIT 仓库](https://github.com/Ascend/tools/) (<https://github.com/Ascend/tools/>) 获取.

新建文件夹 `mksd` (如 `Atlas200/ascend/mksd`), 将 `ubuntu-16.04.3-server-arm64.iso`, `mini_developkit-xxx.rar`, `make_sd_card.py` 和 `make_ubuntu_sd.sh` 放入该文件夹, 准备工作完成.

4.5.2.1.3.3 依赖安装

打开终端, 输入如下命令安装依赖

代码 4.20: 安装交叉编译器与依赖

```
1 su - root
2 apt-get install qemu-user-static binfmt-support python3-yaml gcc-aarch64-linux-gnu
   ↪g++-aarch64-linux-gnu
```

4.5.2.1.3.4 配置 Atlas200DK 开发板 IP 地址

修改文件 `make_sd_card.py` 中的参数值, 使其不与 PC 机冲突

```
# 网卡 IP 地址, 通过网线连接时使用
NETWORK_CARD_DEFAULT_IP="192.168.31.2"
# USB 虚拟 IP 地址, 通过 USB 连接时使用
USB_CARD_DEFAULT_IP="192.168.31.2"
```

警告: 实测发现, IP 192.168.x.y 中, y 只能为 2, 设置成其它值无法 ping 同!

提示: 注意, 上述 IP 地址应与 PC 机处于同一网段内. 若想重新制卡, 需要先删除上述操作生成的 SD 卡上的分区, 可以通过 `sudo apt install gparted` 安装 Gparted 分区工具, 然后执行删除分区的操作即可.

4.5.2.1.3.5 制作 SD 卡启动镜像

进入 *Atlas200/ascend/mksd/* 目录, 执行命令 `python3 make_sd_card.py local /dev/sdx` 制作 SD 卡, 注意 `sdx` 中的 `x` 代表 SD 卡号, 可使用 `sudo fdisk -l` 查看, 输出日志如下

打开终端, 输入如下命令安装依赖

代码 4.21: 制作 SD 卡

```
1 mksd# su - root
2 mksd# python3 make_sd_card.py local /dev/sdc
3 Begin to make SD Card...
4 Please make sure you have installed dependency packages:
5     apt-get install -y qemu-user-static binfmt-support gcc-aarch64-linux-gnu g++
6     ↪aarch64-linux-gnu
6 Please input Y: continue, other to install them:y
7 Step: Start to make SD Card. It need some time, please wait...
8 Make SD Card successfully!
```

提示: 若提示: [ERROR] Can not get disk, please use fdisk -l to check available disk name! 首先检查是否输错 `sdx`, 若无执行 `sudo python3 make_sd_card.py local /dev/sdx`.

4.5.2.1.3.6 连接开发板与上位机

可以通过 Type-c 或网线连接, 可以直连, 也可以通过交换机/路由器相连, 保证开发板 IP 与 PC 机 IP 处于同一网段即可.

小技巧: 在执行配置前, 建议先进行设备检查, 即通过 Type-c 线连接开发板与 PC 机, 在 PC 机中执行 `ifconfig -a` 查看是否有新的网卡设备, 若有, 证明上述制卡过程没问题, 否则很可能是嵌入式系统 `ubuntu` 版本不对, 或者是开发套件 `mini_developkit-xxx.rar` 版本不对, 实验发现, [Ascend tool 仓库](https://github.com/Ascend/tools/) (<https://github.com/Ascend/tools/>) 中文件夹 `B883` 下的软件包做成镜像无法发现设备, 而 `B750SP05` 下的则可以.

无论使用 USB 连接还是网线连接, 均需要配置服务端静态 IP, 可通过图形界面配置, 或者在 PC 端执行如下命令

代码 4.22: 配置服务端静态 IP

```
1 mksd# su - root
2
3 # open file /etc/network/interfaces
4 sudo gedit /etc/network/interfaces
5
6 # add the following information
```

(下页继续)

(续上页)

```

7 auto ethname
8 iface ethname inet static
9 address xxx.xxx.xxxx.xxxx
10 netmask 255.255.255.0
11
12 # modify file /etc/NetworkManager/NetworkManager.conf to enable the above settings
13 #→permanently
13 sudo gedit /etc/NetworkManager/NetworkManager.conf
14
15 # set managed=true
16 managed=true
17
18 # USB
19 ifdown ethname
20 ifup ethname
21 service NetworkManager restart
22
23 # NIC restart network service
24 service networking restart
25 service NetworkManager restart

```

其中, ethname 为网卡名, 通过 ifconfig 查看, 举例如下

代码 4.23: 配置静态 IP

```

1 # enp0s20u2
2 auto enp0s20u2
3 iface enp0s20u2 inet static
4 address 192.168.31.233
5 netmask 255.255.255.0

```

重启网络服务后, 可通过 ifconfig 查看对应网卡 IP 是否设置成功. 接着可以通过 ping 192.168.x.2 来测试是否连接成功, 其中 x 为在制作 SD 卡中设置的值!

4.5.2.1.3.7 登录开发板系统

由于通过网络连接开发板与 PC 机, 故可通过 ssh 协议访问, 若要登录开发板系统, 在 PC 端输入指令 ssh HwHiAiUser@IP (如 ssh HwHiAiUser@192.168.31.2) 即可, 默认 HwHiAiUser 密码为 Mind@123.

代码 4.24: 登录开发板系统

```

1 bee@Smart:~$ ssh HwHiAiUser@192.168.31.2
2 HwHiAiUser@192.168.31.2's password:
3 Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.19.36+ aarch64)

4
5 * Documentation: https://help.ubuntu.com
6 * Management: https://landscape.canonical.com

```

(下页继续)

(续上页)

```
7 * Support:          https://ubuntu.com/advantage
8
9 The programs included with the Ubuntu system are free software;
10 the exact distribution terms for each program are described in the
11 individual files in /usr/share/doc/*/*copyright.
12
13 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
14 applicable law.
15
16 HwHiAiUser@davinci-mini:~$ ls
17 dump  hdcd  HIAI_PROJECTS  ide_daemon
18 HwHiAiUser@davinci-mini:~$
```

4.5.2.1.4 MindSpore Studio 开发

MindSpore Studio 的 Matrix 流程编排功能提供 AI 引擎可视化拖拽式编程及算法代码自动生成技术, 极大的降低了开发者的门槛. Matrix 中的 Engine 是一个具体的业务节点, 一个业务节点表示一次处理过程, 多个 Engine 组成一个 Graph, Graph 负责对 Engine 的管理. 每个 Engine 节点间的连接在 Graph 配置文件中配置, 节点间数据的实际流向根据具体业务在节点中实现, 通过向业务的开始节点灌入数据启动整个 Engine 计算流程. (引自华为官网)

4.5.2.1.4.1 开发概览

Mind Studio 的 Engine 流程编排功能提供 AI 引擎可视化拖拽式编程及算法代码自动生成技术, 极大的降低了开发者的门槛。其中, 最核心的步骤是编排流程, Mind Studio 的 Engine 编排流程如 图 4.14 所示, 该流程主要包含以下几个步骤:

- 模型转换: 将 Caffe/Tensorflow 等模型转换为适配硬件环境的模型。
- 图像预处理: 对数据进行特定的预处理, 比如将对信号进行时频变换, 或者需要对图片进行指定的裁剪、缩放、格式转换等操作, 以满足模型推理的需要。
- 模型推理: 将预处理后的输入进行推理执行输出结果。
- 图像后处理: 对数据进行特定的后处理, 比如将模型推理结果保存在文件中, 或者根据数据在图像上标注类别、概率、检测框等信息。

Engine 开发详细流程可参考官方文档 [Engine 编排总体流程](https://ascend.huawei.com/doc/Atlas200DK/1.3.0.0/zh-zh-cn_topic_0160786246.html) (https://ascend.huawei.com/doc/Atlas200DK/1.3.0.0/zh-zh-cn_topic_0160786246.html).

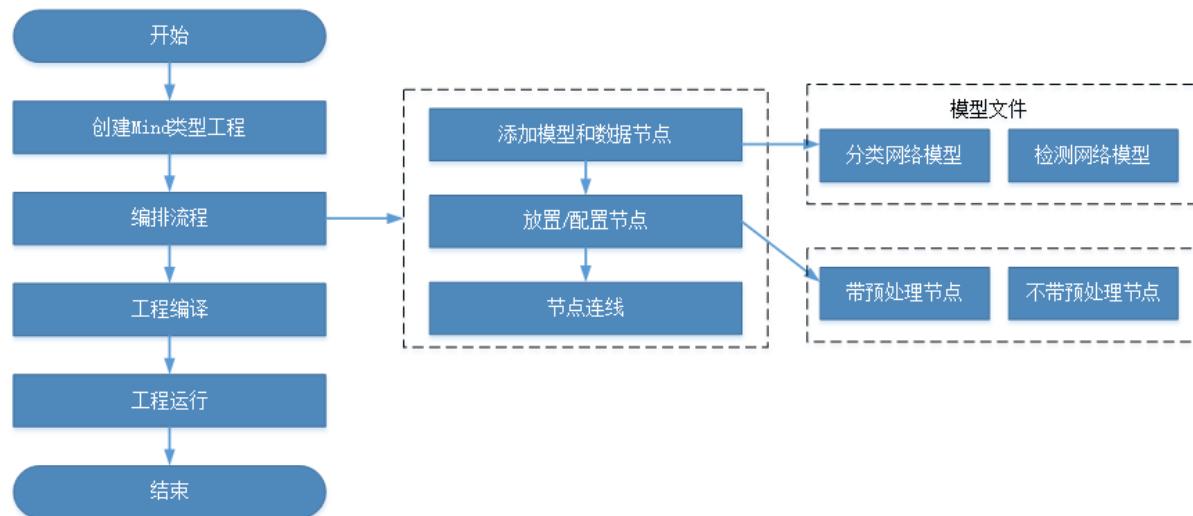


图 4.14: Engine 编排总体流程 (来自华为官网)

4.5.2.1.4.2 添加与管理开发板设备

启动 MindSpore Studio, 依次点击 Tool-->Atlas DK Configuration, 在弹出的对话框中按提示输入信息 (图 4.15), 添加后如 图 4.16 所示.

危险: 经测试, 使用 Firefox 浏览器添加设备会弹出 ERROR 对话框, 但又不提示错误信息, 使用 Google Chrome 无此问题, 可正常添加.

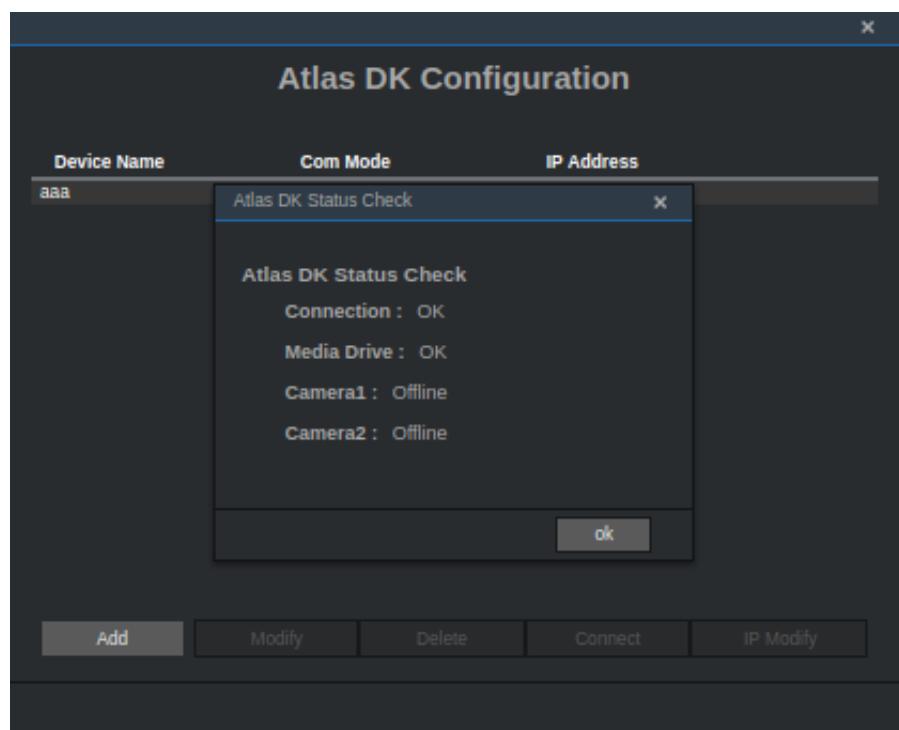


图 4.15: 添加开发板设备

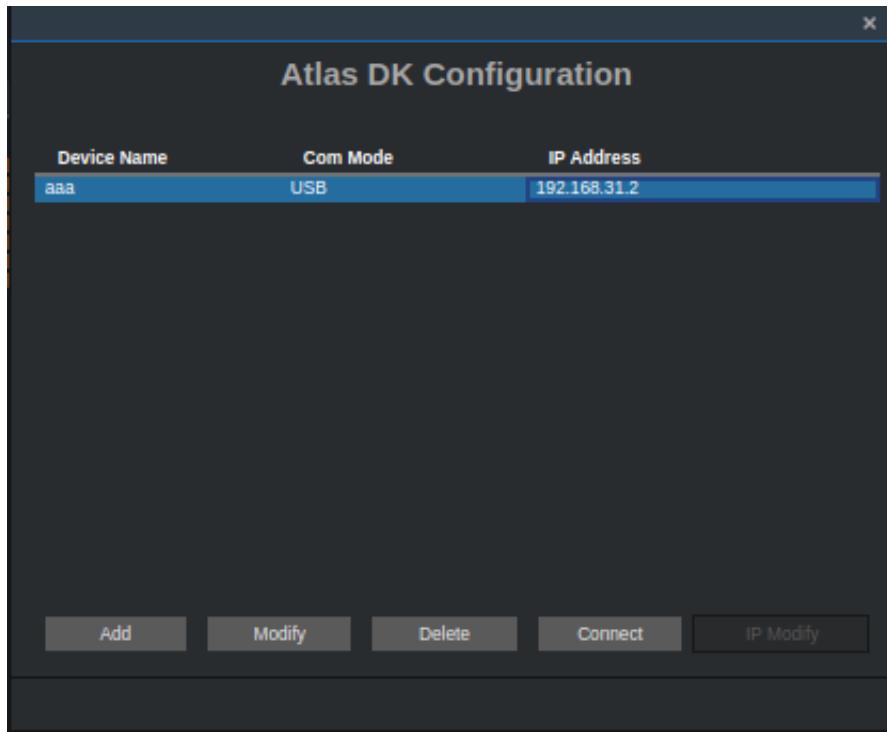


图 4.16: 添加好开发板设备的示意图

4.5.2.1.4.3 自定义算子

4.5.2.1.5 深度学习移植实例

4.5.2.1.5.1 简单例子

4.5.2.1.6 PyTorch 转 Caffe

- [PytorchToCaffe](https://github.com/xxradon/PytorchToCaffe) (<https://github.com/xxradon/PytorchToCaffe>)

4.6 嵌入式系统开发

CHAPTER 5

第五卷信号处理

5.1 模拟信号处理

5.1.1 名词术语

Analog Signal Processing 模拟信号处理 ([Digital Analog Processing](http://en.volupedia.org/wiki/Analog_signal_processing) (http://en.volupedia.org/wiki/Analog_signal_processing))

Digital Signal Processing 数字信号处理 ([Digital Signal Processing](http://en.volupedia.org/wiki/Digital_signal_processing) (http://en.volupedia.org/wiki/Digital_signal_processing))

Nolinear Signal Processing 非线性号处理 ([Nolinear Signal Processing](http://en.volupedia.org/wiki/Nolinear_Signal_processing) (http://en.volupedia.org/wiki/Nolinear_Signal_processing))

Signal Processing 信号处理 ([Signal Processing](http://en.volupedia.org/wiki/Signal_processing) (http://en.volupedia.org/wiki/Signal_processing))

Sparse Signal Processing 稀疏信号处理 ([Sparse Signal Processing](http://en.volupedia.org/wiki/Sparse_signal_processing) (http://en.volupedia.org/wiki/Sparse_signal_processing))

Statistical Signal Processing 统计信号处理 (([Statistical Signal Processing](http://en.volupedia.org/wiki/Signal_processing#Statistical) (http://en.volupedia.org/wiki/Signal_processing#Statistical)))

5.2 数字信号处理

5.2.1 引言

涉及滤波, 时频分析, 谱估计分析等理论.

5.2.2 基础内容

5.2.2.1 简介

5.2.2.2 插值算法

5.2.2.2.1 Sinc 插值

5.2.2.2.1.1 Sinc 插值原理

Sinc 插值 (*Sinc Interpolation*) 也叫 **Whittaker–Shannon interpolation** 是一种从真实离散序列构造一个连续时间带限函数的方法, 即根据实际离散采样点逼近连续函数.

$$\begin{aligned}x(t) &= \sum_{n=-\infty}^{+\infty} x[n] \operatorname{sinc}\left(\frac{t-nT}{T}\right) \\&= \sum_{n=-\infty}^{+\infty} x[n] \operatorname{sinc}\left(\frac{t}{T}-n\right)\end{aligned}$$

其中, T 为采样间隔, $\operatorname{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$ 称为归一化的 sinc, 与数学中定义的 $\operatorname{sinc}(x) = \frac{\sin(x)}{x}$ 不同.

上式可以表示为卷积形式:

$$x(t) = \left(\sum_{n=-\infty}^{+\infty} x[n] \cdot \delta(t - nT) \right) * \operatorname{sinc}\left(\frac{t}{T}\right),$$

相当于采用一个低通滤波器滤波.

5.2.2.2.1.2 实验与分析

5.2.2.2.1.3 sinc 函数特性

5.2.2.2.1.4 实验代码

本书 Python 实现代码, 参见文件 `demo_SincProperties.py`.

5.2.2.2.1.5 实验结果

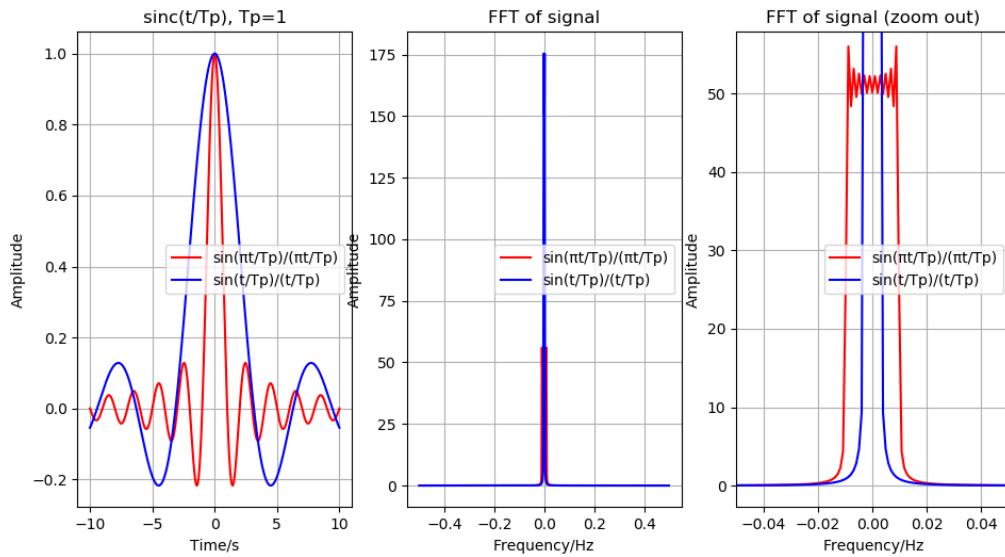


图 5.1: Properties of sinc, $T_p = 1s$

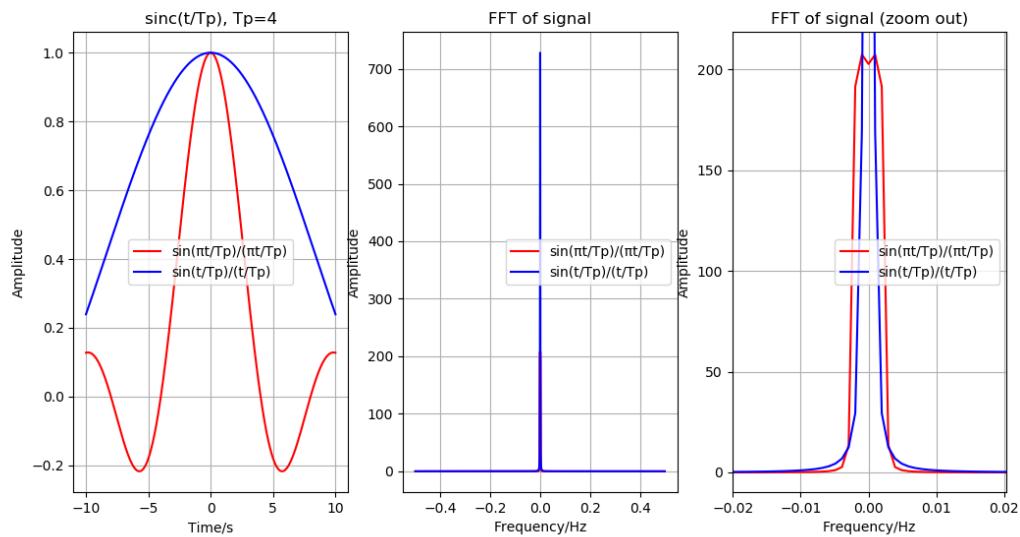


图 5.2: Properties of sinc, $T_p = 4s$

5.2.2.2.1.6 sinc 插值

5.2.2.2.1.7 实验代码

可参考代码:

- MATLAB code by Dr. John S. Loomis (http://www.johnloomis.org/ece561/notes/sinc_interp/sinc_interp.html)
- Time-domain Sinc Interpolation (Resampling) (<https://ww2.mathworks.cn/matlabcentral/fileexchange/59027-time-domain-sinc-interpolation-resampling>)

“MATLAB code by Dr. John S. Loomis” \rightarrow *sinc_interp.m*

```
function y = sinc_interp(x,u)

m = 0:length(x)-1;

for i=1:length(u)
    y(i) = sum(x.*sinc(m-u(i)));
end
```

demo_sinc_interp.m

```
a = 0.9;
N = 64;
n = 0:N-1;
x = n.*a.^n;

plot(n,x);

y = fft(x);
k= 0:N/2;
plot(k/N,abs(y(1:N/2+1)));

s = linspace(0,63,512);
x2 = sinc_interp(x,s);
plot(s(1:256), x2(1:256), '-r');
title('sinc interpolated')
hold
xi = interp1(n, x, s);
plot(s(1:256),xi(1:256),'k');
title('interp1 interpolated')
plot(n(1:N/2),x(1:N/2),'o');

legend({'sinc interpolated', 'interp1 interpolated', 'original'})
hold off
```

本书 Python 实现代码, 参见文件 [demo_SincInterpolation.py](#).

5.2.2.2.1.8 实验结果

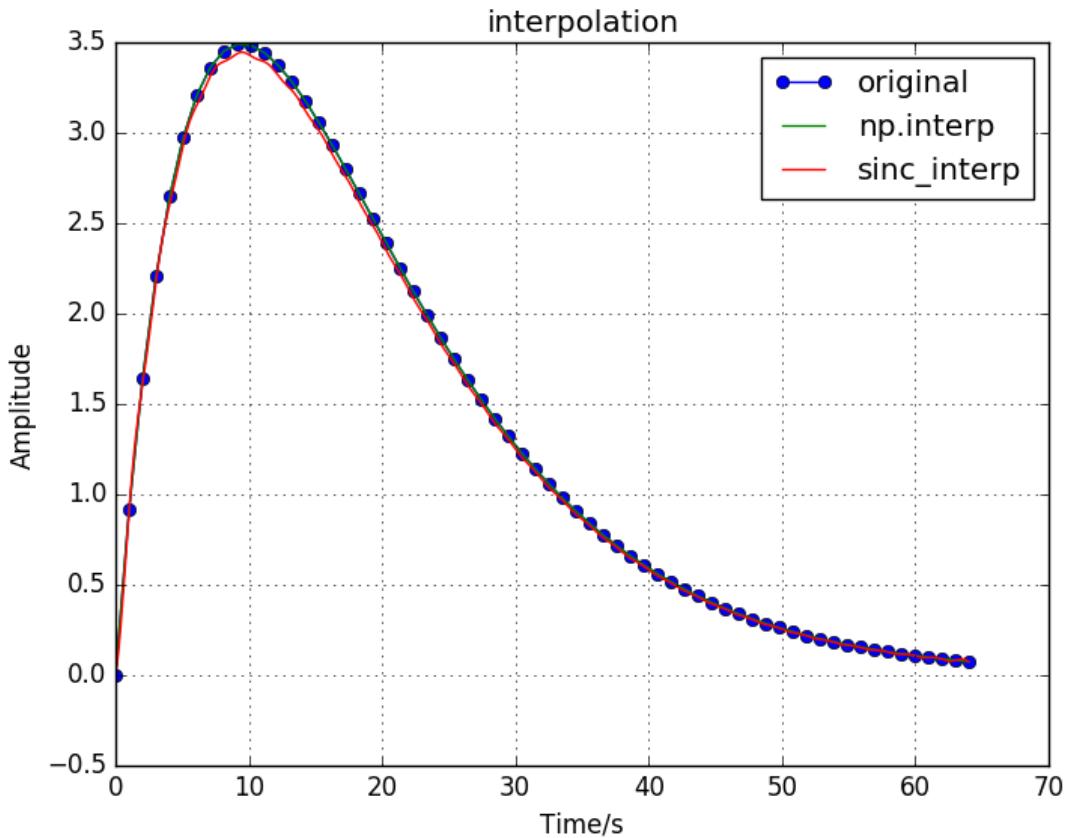


图 5.3: sinc 插值示例 1

5.2.3 谱估计理论

5.2.3.1 简介

5.2.3.1.1 What?

5.2.3.1.2 Why?

5.2.3.1.3 How?

5.2.3.2 经典谱估计

5.2.3.2.1 经典谱估计

5.2.3.2.1.1 谱估计

谱估计又称谱密度估计 (Spectral Density Estimation)，是一种频谱分析方法，分为经典谱估计 (*Classical Spectral Estimation*) 和现代谱估计 (*Modern Spectral Estimation*) 两大类。

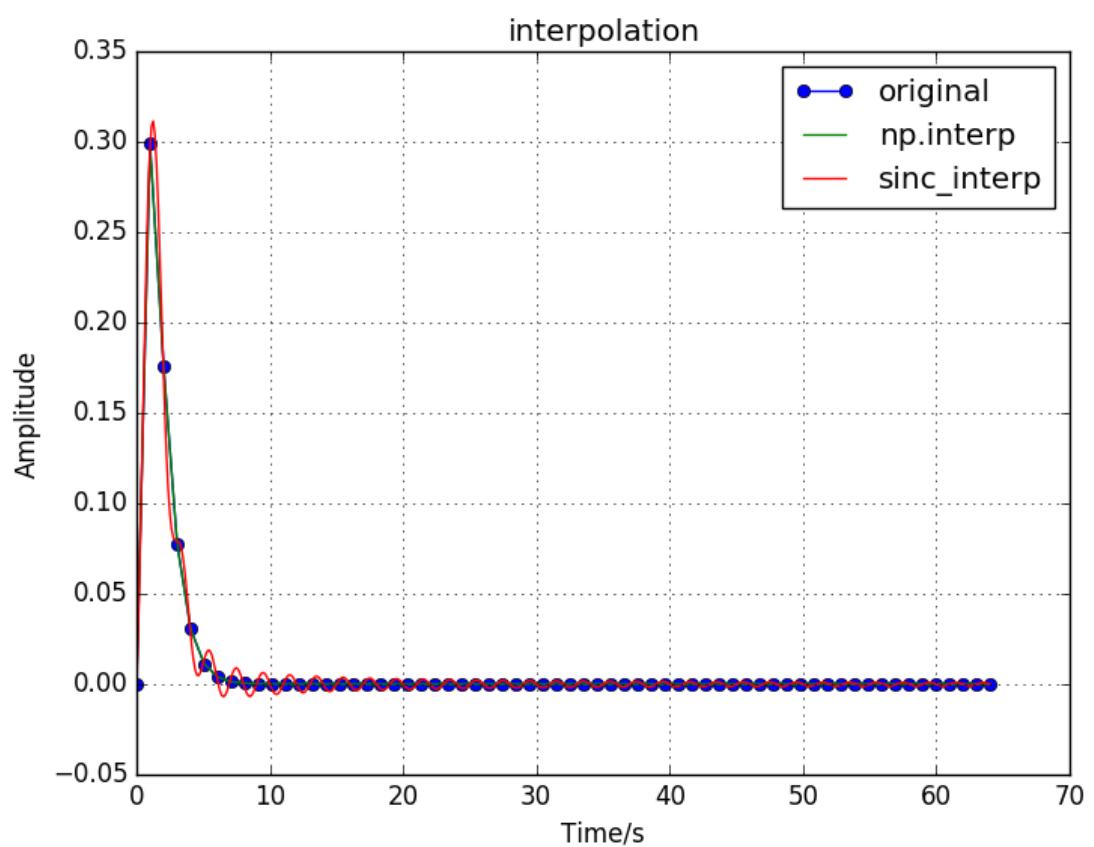


图 5.4: sinc 插值示例 2

- 经典谱估计(非参数化谱估计)
- 现代谱估计(参数化谱估计)

5.2.3.2.1.2 经典谱估计

5.2.3.2.2 窗函数及其性质

5.2.3.2.2.1 概念

5.2.3.2.2.2 类型

5.2.3.2.2.3 hamming

5.2.3.2.2.4 cosine

5.2.3.2.2.5 blackman

5.2.3.2.2.6 gaussian

5.2.3.2.2.7 tukey

5.2.3.2.2.8 kaiser

5.2.3.2.2.9 实例

5.2.3.2.2.10 窗函数比较

5.2.3.2.2.11 实验内容

产生 $f = 100Hz$ 的余弦信号 $x(t)$, 采样率为 $F_s = 1000Hz$, 采样时间 $T_s = 0.5s$, 对信号 $x(t)$ 施加六种窗函数, 窗函数持续时间 $T_w = T_s = 0.5s$, 并对加窗前后的数据进行快速傅里叶变换.

- hamming
- cosine
- blackman
- gaussian, std = 100
- tukey, = 10
- kaiser, = 10

5.2.3.2.2.12 实验代码

代码文件 [demo_windows_effect.py](#)

代码 5.1: demo_windows_effect.py

```

1 import numpy as np
2 from scipy import signal
3 import matplotlib.pyplot as plt
4
5
6 f = 100
7 Fs = 1000
8 T = 0.5
9 Tw = 0.5
10
11 Ns = int(T * Fs)
12
13 Nw = int(Tw * Fs)
14
15 t = np.linspace(0, T, Ns)
16 tw = np.linspace(0, Tw, Nw)
17
18 x = np.cos(2 * np.pi * f * t)
19
20
21 # windn = ('tukey', 10)
22 # windn = ('kaiser', 10)
23 # windn = ('gaussian', 100)
24 # windn = ('hamming')
25 # windn = ('cosine')
26 windn = ('blackman')
27
28 window = signal.get_window(windn, Nw)
29
30
31
32 print(Nw, len(window))
33
34 xfft = np.fft.fft(x)
35 ffft = np.fft.fftfreq(x.shape[-1])
36
37 idx = 0
38 y = x.copy()
39 y[idx:idx + Nw] = x[idx:idx + Nw] * window
40
41 print(y.shape, t.shape)
42

```

(下页继续)

(续上页)

```
43 plt.figure(figsize=(10, 10))
44 plt.subplot(221)
45 plt.grid()
46 plt.plot(t, x)
47 plt.title('signal')
48 plt.xlabel('Time/s')
49 plt.ylabel('Amplitude')

50
51 plt.subplot(222)
52 plt.grid()
53 plt.plot(tw, window)
54 plt.title('window: ' + str(windn))
55 plt.xlabel('Time/s')
56 plt.ylabel('Amplitude')

57
58 plt.subplot(223)
59 plt.grid()
60 plt.plot(t, y)
61 plt.title('applied window on signal')
62 plt.xlabel('Time/s')
63 plt.ylabel('Amplitude')

64
65 yfft = np.fft.fft(y)

66
67 plt.subplot(224)
68 plt.grid()
69 plt.plot(ffft * Fs, np.abs(xfft), '-b')
70 plt.plot(ffft * Fs, np.abs(yfft), '-r')
71 plt.title('FFT of signal')
72 plt.legend(['original', 'windowed'])
73 plt.xlabel('Frequency/Hz')
74 plt.ylabel('Amplitude')
75 plt.show()
```

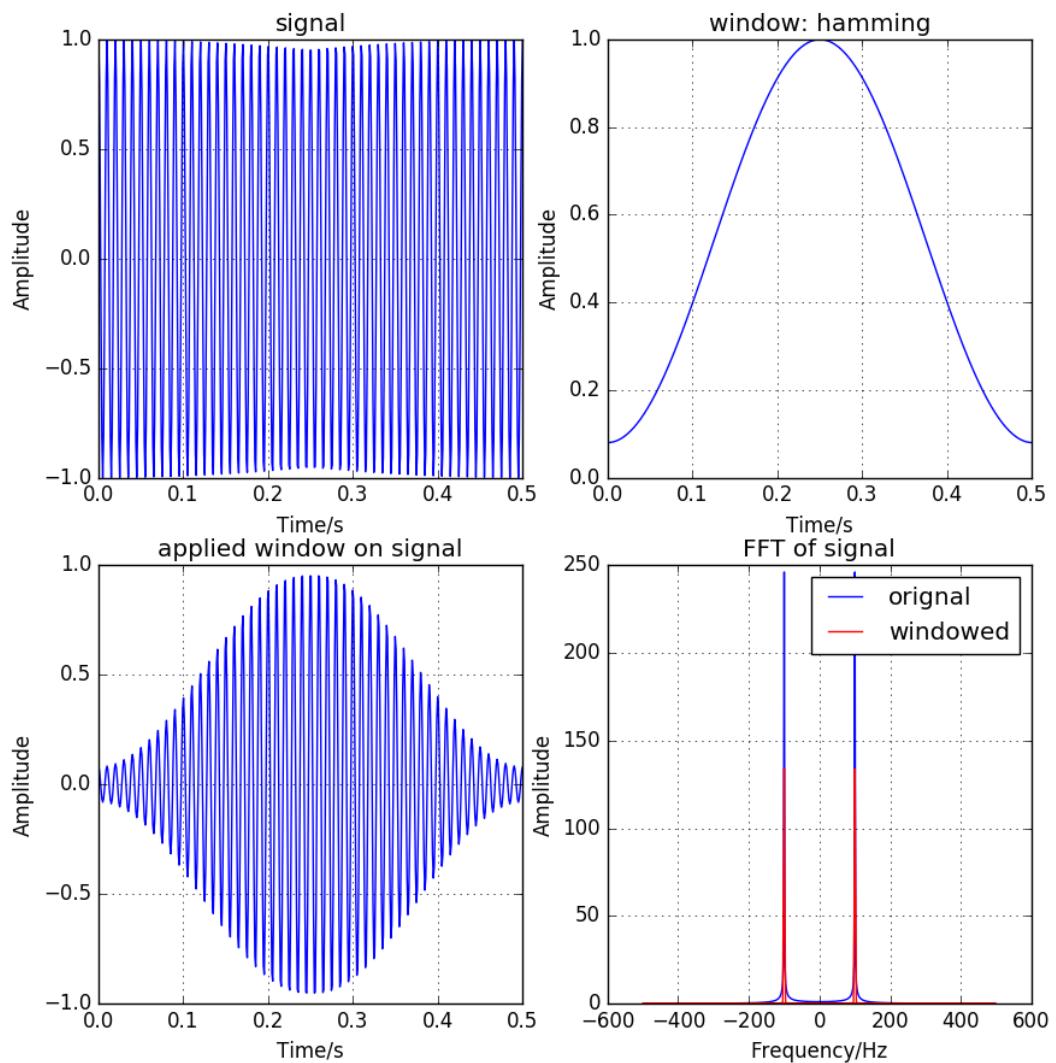


图 5.5: effect of hamming window function

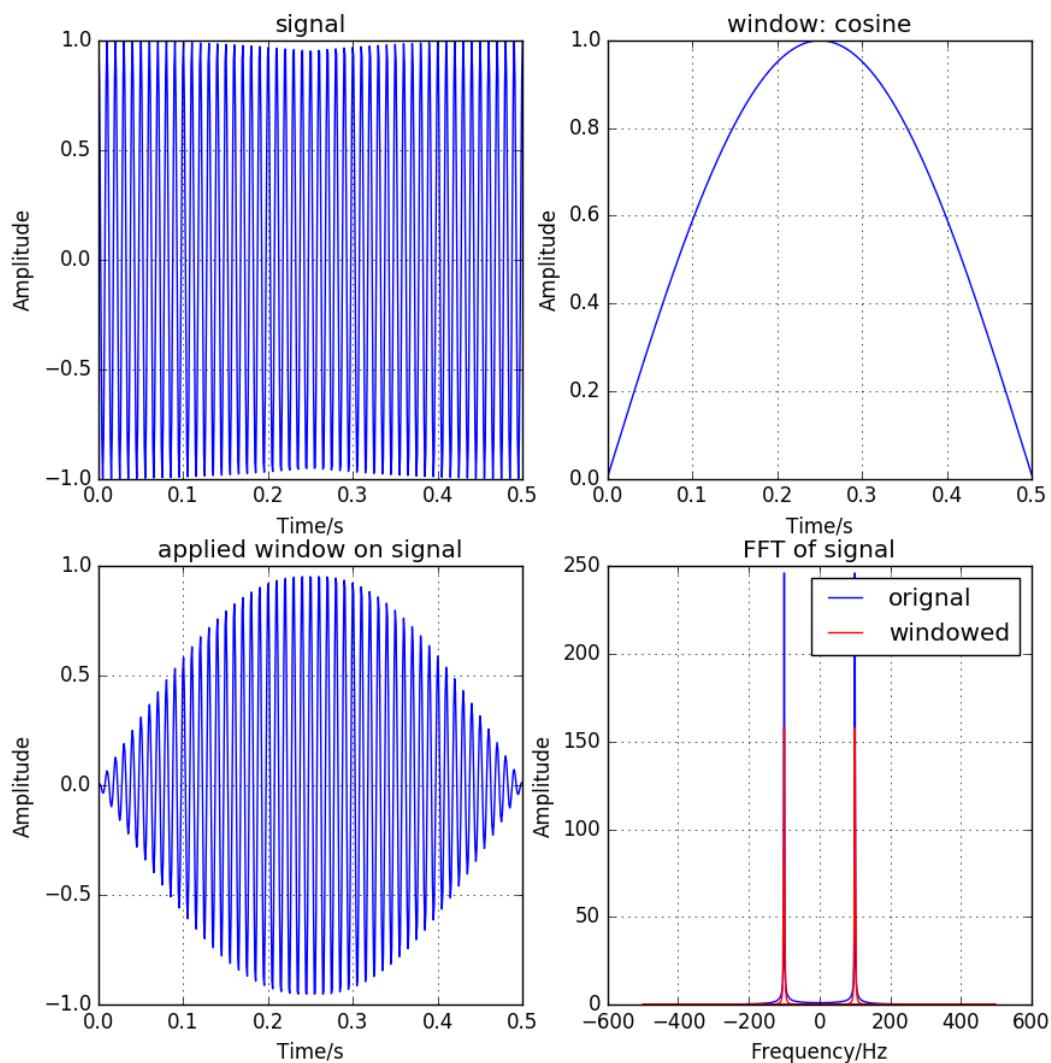


图 5.6: effect of cosine window function

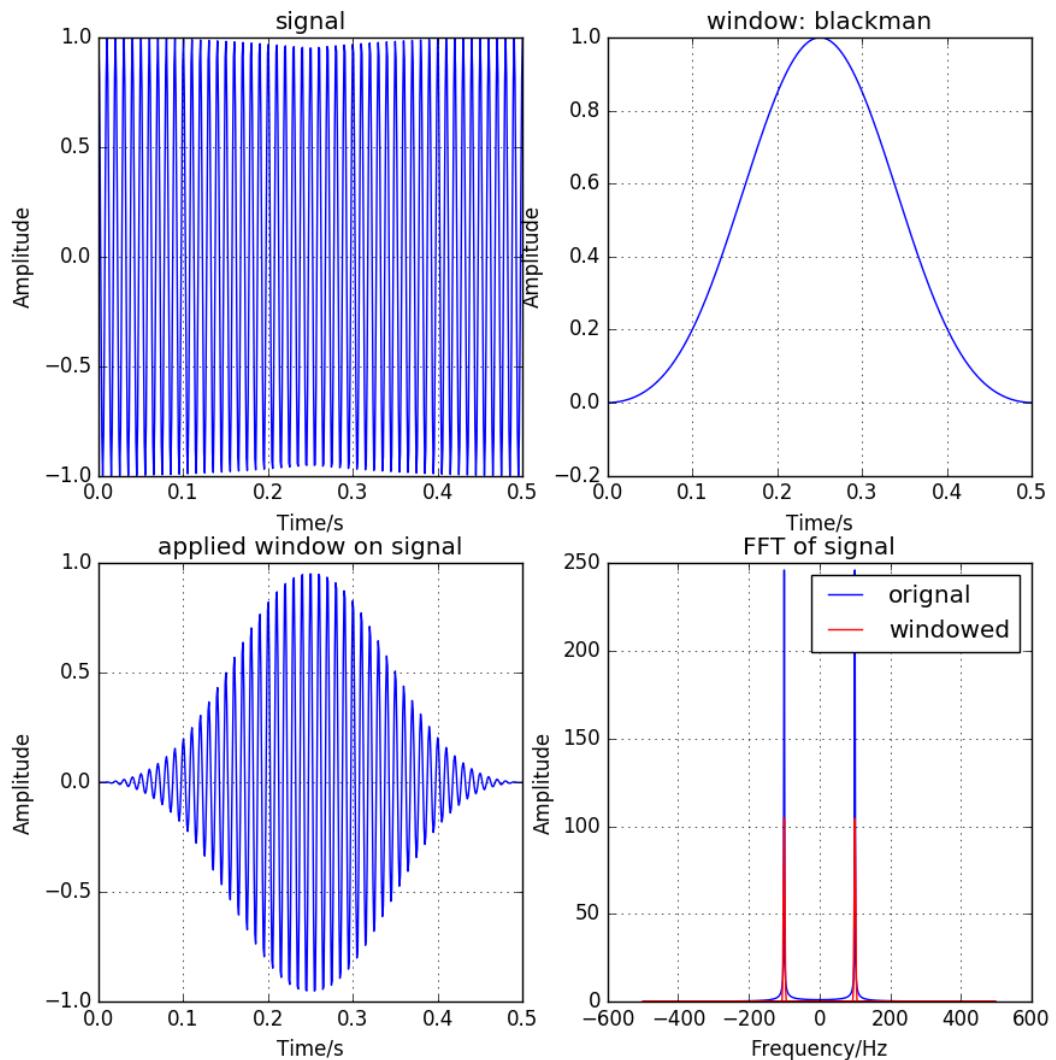
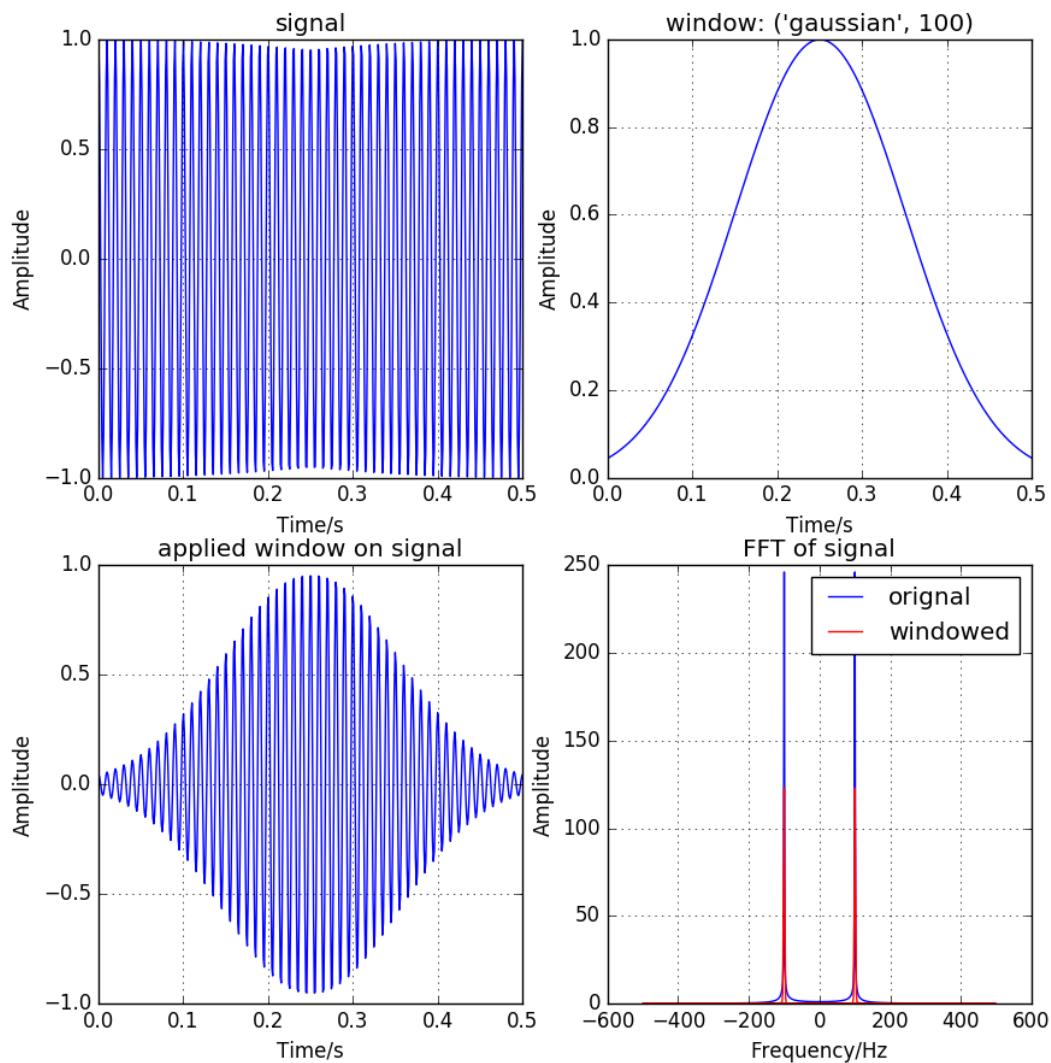


图 5.7: effect of blackman window function

图 5.8: effect of gauss window function, with $\text{std} = 100$

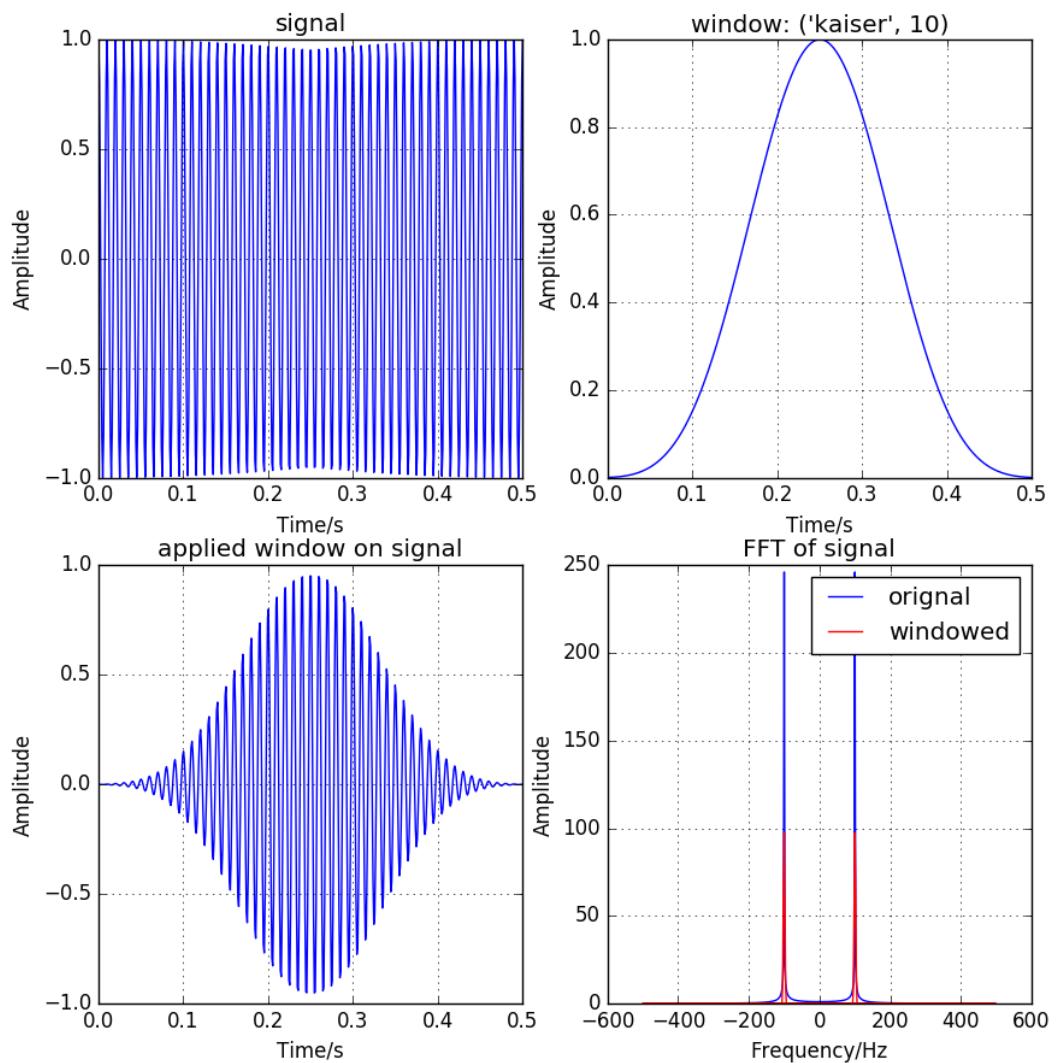
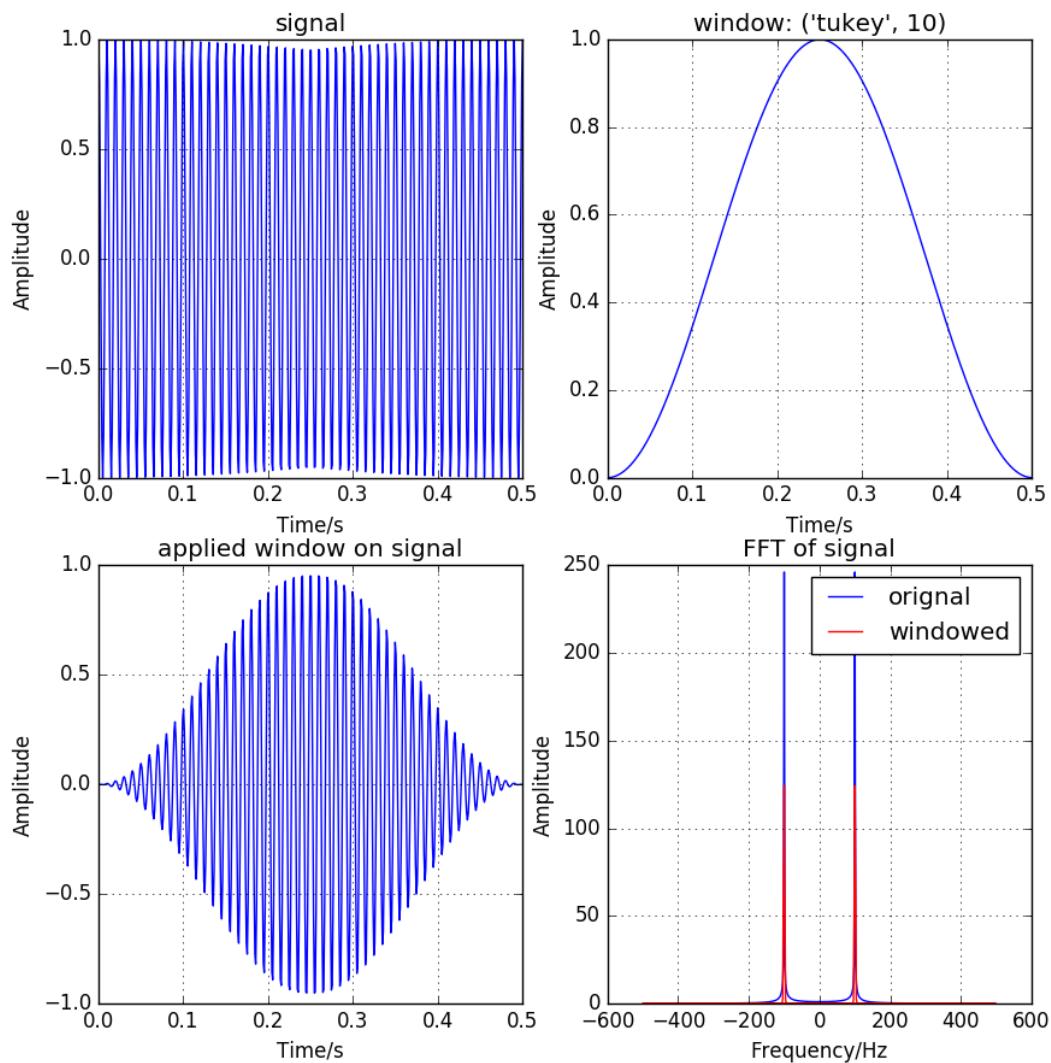


图 5.9: effect of kaiser window function, with $\beta = 10$

图 5.10: effect of tukey window function, with $\alpha = 10$

5.2.3.2.2.13 实验结果

5.2.3.2.3 傅立叶变换

5.2.3.2.3.1 概念与内涵

5.2.3.2.3.2 实验分析

5.2.3.2.3.3 仿真数据实验

5.2.3.2.3.4 实验代码

MATLAB 代码

[test_FFT.m](#).

PYTHON 代码

[test_FFT.py](#).

5.2.3.2.3.5 实验结果

5.2.3.2.3.6 真实数据实验

5.2.3.2.4 傅立叶变换与卷积

5.2.3.2.4.1 概念与内涵

5.2.3.2.4.2 FFT 实现卷积

通过补零将卷积核与数据补成相同大小, 然后对各自做 FFT,

注解: FFT 实现二维图像卷积(卷积步长为 1)

输入: 数据矩阵 $\mathbf{I} \in \mathbb{R}^{I_h \times I_w}$, 卷积核 $\mathbf{K} \in \mathbb{R}^{K_h \times K_w}$ 输出: 卷积结果 $\mathbf{O} = \mathbf{I} * \mathbf{K}$, 其中, $\mathbf{O} \in \mathbb{R}^{O_h \times O_w}$, $O_h = I_h + K_h/2 - 1$, $O_w = I_w + K_w/2 - 1$

Step 1: 对数据矩阵 \mathbf{I} 与卷积核 \mathbf{K} 周边进行补零, 补成 $O_h \times O_w$ 大小的矩阵

Step 2: 对数据矩阵 \mathbf{I} 做二维傅里叶变换, 得到其频域二维形式 $I = \text{FFT}(\mathbf{I})$

Step 3: 对卷积核矩阵 \mathbf{K} 做二维傅里叶变换, 得到其频域二维形式 $K = \text{FFT}(\mathbf{K})$

Step 4: 通过频域相乘(对应元素相乘)得到卷积后的频域结果 $O = G \odot H$

Step 5: 通过逆傅里叶变换得到卷积结果 $\mathbf{O} = \text{IFFT}(O)$

Step 6: \mathbf{O}

设有数据矩阵 $\mathbf{I} \in \mathbb{R}^{I_h \times I_w}$, 卷积核 $\mathbf{K} \in \mathbb{R}^{K_h \times K_w}$, 利用 FFT 实现图像二维卷积的步骤为

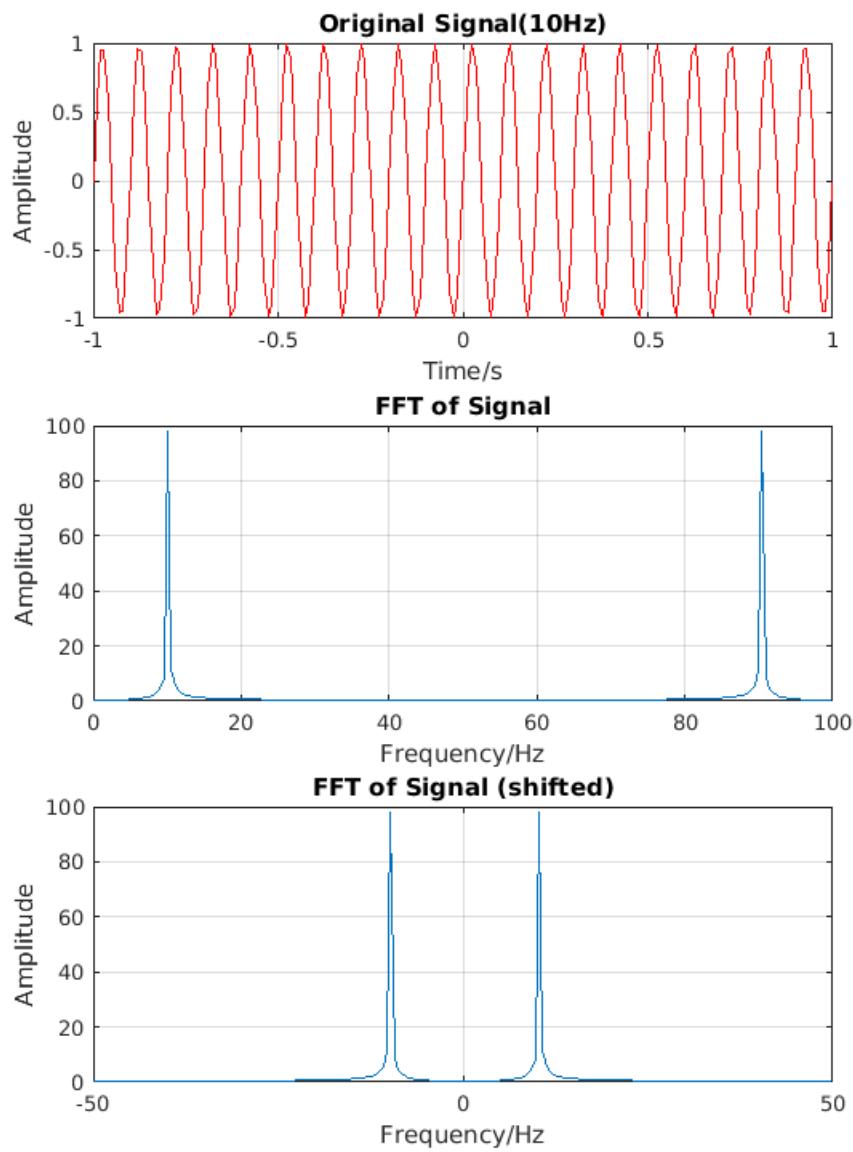


图 5.11: FFT matlab
FFT matlab

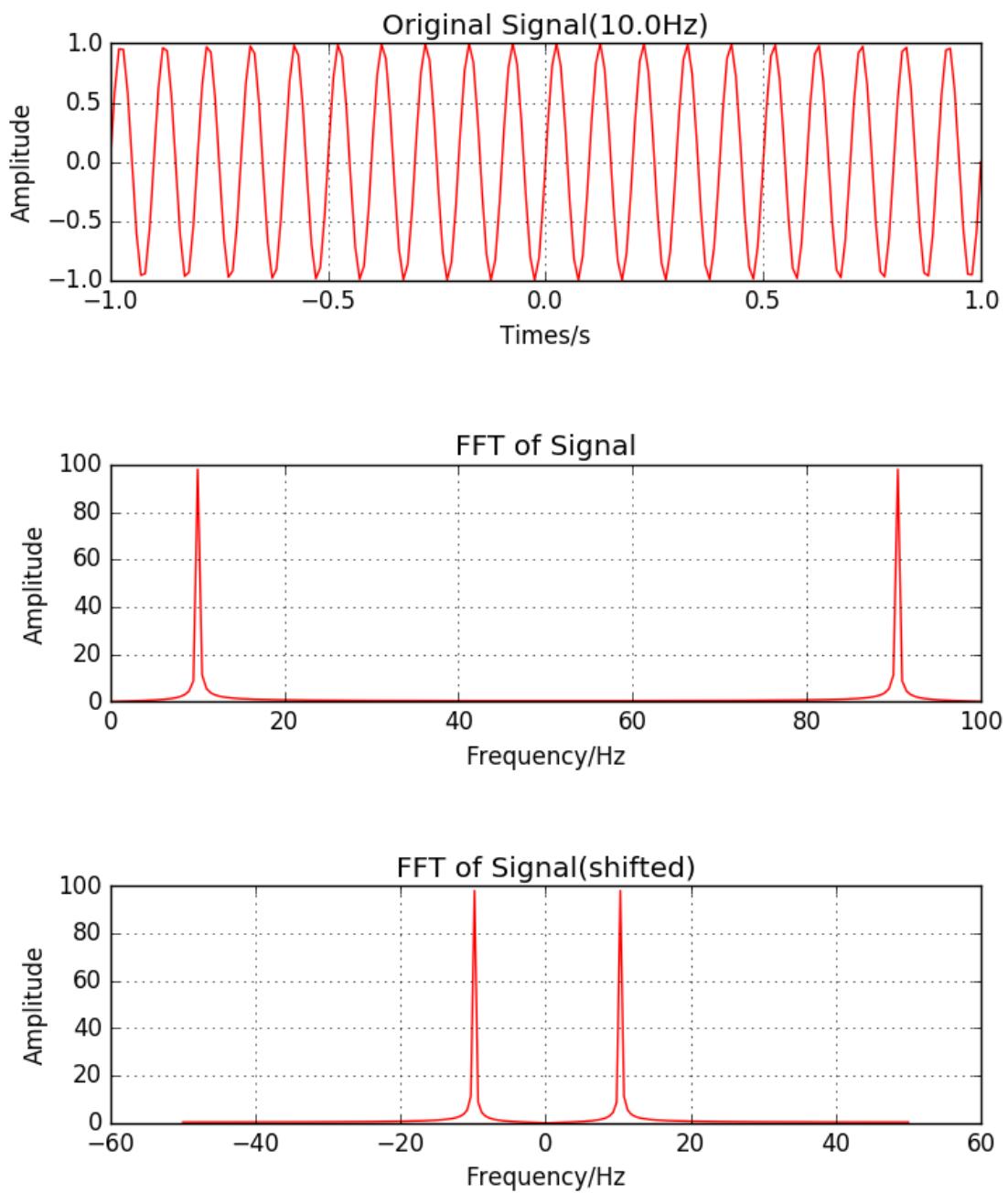


图 5.12: FFT python
FFT python

- 将数据矩阵 $I \in \mathbb{R}^{H \times W}$,

5.2.3.2.4.3 实验与分析

5.2.3.2.4.4 FFT 实现二维矩阵卷积

MATLAB 代码:

代码 5.2: demo_Array_FFT_Conv2d.m

```

1 clear all
2
3 % ---data
4 X = [1 2 3 4 5 6; 5 6 7 8 9 10; 9 10 11 12 13 14; 9 10 11 12 13 14;
5 %---convolution kernel
6 K = [0 1 1;-1 0 1;-1 -1 0];
7
8 [Kh, Xw, Xc] = size(X);
9
10 % ---FFT
11 [Kh, Kw] = size(K);
12
13 % ---pad (zeros) kernel to the same size of data X
14 KK = zeros(Xh, Xw);
15 KK(1:Kh, 1:Kw) = K;
16
17 % ---Conv2d valid
18 Y = fft2(X);
19 H = fft2(KK);
20 Z = Y .* H;
21 O1 = ifft2(Z);
22
23 % ---Conv2d same
24 O2 = conv2(X, K, 'valid');
25
26 % ---Conv2d same
27 O3 = conv2(X, K, 'same');
28
29 disp('---By FFT')
30 disp(O1)
31
32 disp('---By Conv(valid)')
33 disp(O2)
34
35 disp('---By Conv(same)')
36 disp(O3)
37

```

```
>> demo_Array_FFT_Conv2d
---By FFT
 -8.0000   -8.0000  -20.0000  -20.0000  -20.0000  -20.0000
      0         0  -12.0000  -12.0000  -12.0000  -12.0000
 24.0000   24.0000   12.0000   12.0000   12.0000   12.0000
 16.0000   16.0000    4.0000    4.0000    4.0000    4.0000
  8.0000    8.0000   -4.0000   -4.0000   -4.0000   -4.0000
  8.0000    8.0000   -4.0000   -4.0000   -4.0000   -4.0000

---By Conv(valid)
 12     12     12     12
   4      4      4      4
  -4     -4     -4     -4
  -4     -4     -4     -4

---By Conv(same)
   3      9     11     13     15     24
   0     12     12     12     12     30
  -12     4      4      4      4     30
  -20     -4     -4     -4     -4     26
  -20     -4     -4     -4     -4     26
  -29    -23    -25    -27    -29     -1
```

5.2.3.2.4.5 FFT 实现二维图像卷积

MATLAB 代码:

代码 5.3: demo_Image_FFT_Conv2d.m

```
1 clear all
2 close all
3
4 % ---image data
5 imgfile = 'trees.tif';
6 X = imread(imgfile);
7
8 X = X(:, :, 1);
9 [Xh, Xw, Xc] = size(X);
10
11 % ---convolution kernel
12 K = [0 1 1;-1 0 1;-1 -1 0];
13 [Kh, Kw] = size(K);
14
15 % ---pad (zeros) kernel to the same size of data X
16 KK = zeros(Xh, Xw);
17 KK(1:Kh, 1:Kw) = K;
18
```

(下页继续)

(续上页)

```

19 % ---FFT
20 Y = fft2(X);
21 H = fft2(KK);
22 Z = Y .* H;
23 O1 = ifft2(Z);
24
25 % ---Conv2d valid
26 O2 = conv2(X, K, 'valid');
27
28 % ---Conv2d same
29 O3 = conv2(X, K, 'same');
30
31 figure
32 subplot(221)
33 imshow(X, [])
34 title('Original')
35 subplot(222)
36 imshow(O1, [])
37 title('By FFT')
38 subplot(223)
39 imshow(O2, [])
40 title('By Conv2d (valid)')
41 subplot(224)
42 imshow(O3, [])
43 title('By Conv2d (same)')
44 axis tight
45
46
47
48

```

5.2.3.3 现代谱估计

5.2.3.3.1 现代谱估计

5.2.3.3.1.1 现代谱估计

5.2.3.3.1.2 概念与内涵

5.2.3.3.1.3 现代普估计方法

- 基于滤波器的谱估计
- 基于信号子空间的谱估计
- 基于噪声子空间的谱估计

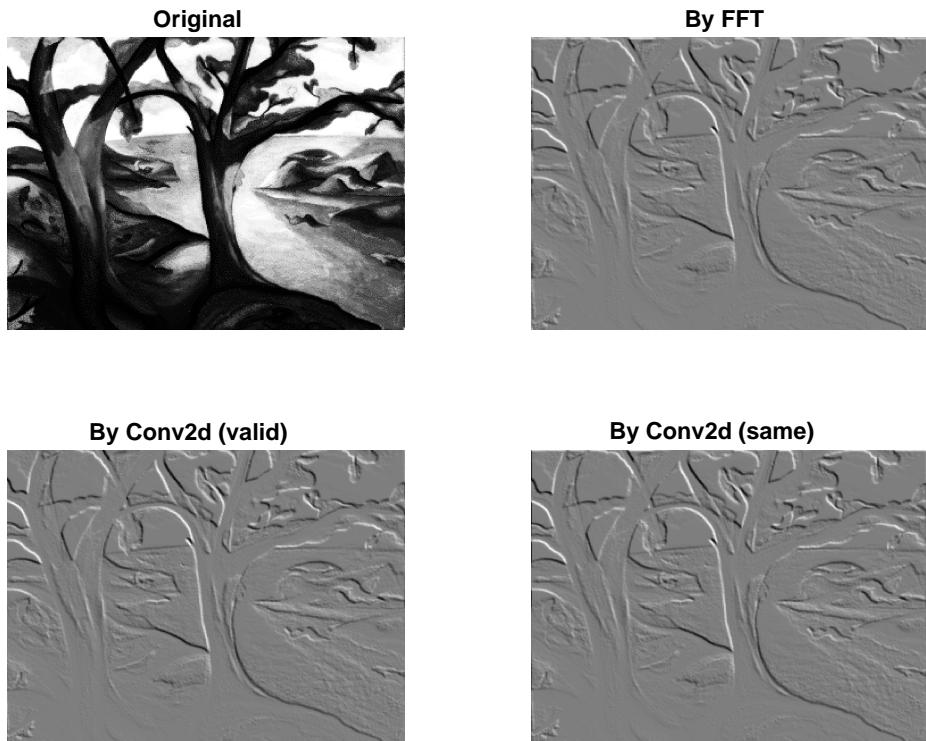


图 5.13: FFT 实现图像二维卷积与直接卷积结果对比
FFT 实现图像二维卷积与直接卷积结果对比.

5.2.3.3.2 基于滤波器的谱估计

5.2.3.3.2.1 简介

5.2.3.3.3 基于信号子空间的谱估计

5.2.3.3.3.1 简介

5.2.3.3.4 基于噪声子空间的谱估计

5.2.3.3.4.1 多重信号分类

在许多实际信号处理问题中, 目标是从测量中估计接收信号所依赖的一组常量参数. 对于这类问题, 有几种方法可以解决, 如 Capon 1969 年提出最大似然法和 Burg 提出的最大熵法. 虽然这些方法已经被广泛使用, 但它们有一定的局限性(尤其是参数估计中的偏差和敏感性), 这主要是因为它们使用了不正确的测量模型(例如, 使用自回归 AR 模型而不是特殊的 ARMA 模型). Pisarenko 于 1973 年首次利用数据模型结构来估计信号参数, 如使用协方差方法估计带有加性噪声的复杂正弦信号的参数. Schmidt 于 1977 年, 通过研究干净无噪声信号, 推导出一个完整的几何解, 然后巧妙地扩展几何概念, 在有噪声的情况下获得一个合理的近似解. 该算法被称为多信号分类(MUSIC), 并得到了广泛的研究. MUSIC 是目前公认的最有效的高分辨率算法之一, 尽管音乐的性能优势是实质性的, 但其计算与存储资源消耗大.

5.2.3.3.4.2 多信号分类用于频率估计

多重信号分类 (MUltiple SIgnal Classification , MUSIC)

5.2.3.3.4.3 实例分析

5.2.3.3.4.4 仿真信号

5.2.3.3.4.5 实验内容

5.2.3.3.4.6 实验代码

5.2.3.3.4.7 实验结果

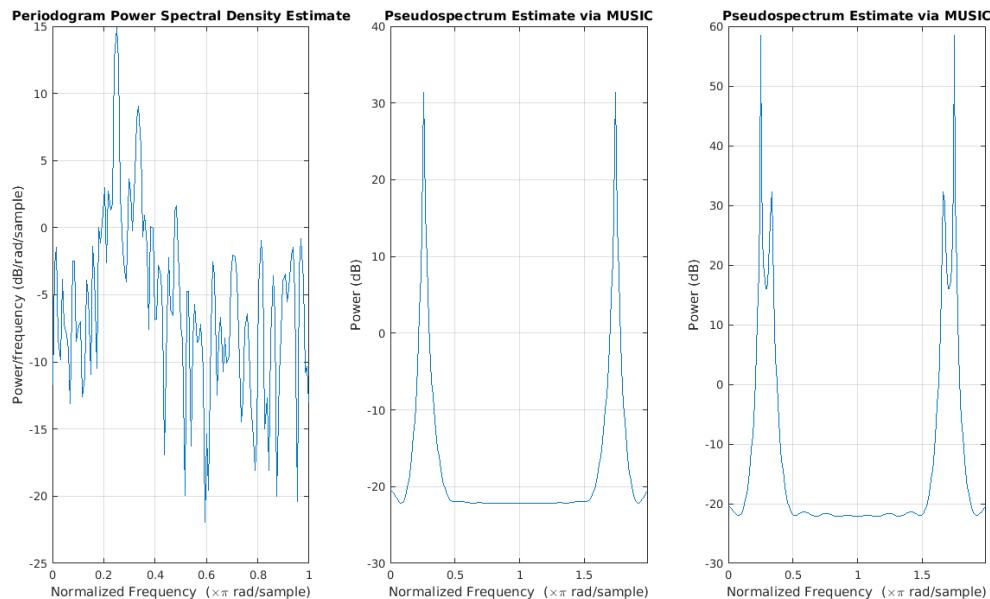


图 5.14: Synthetic signal
合成的信号, 包含四种频率成分.

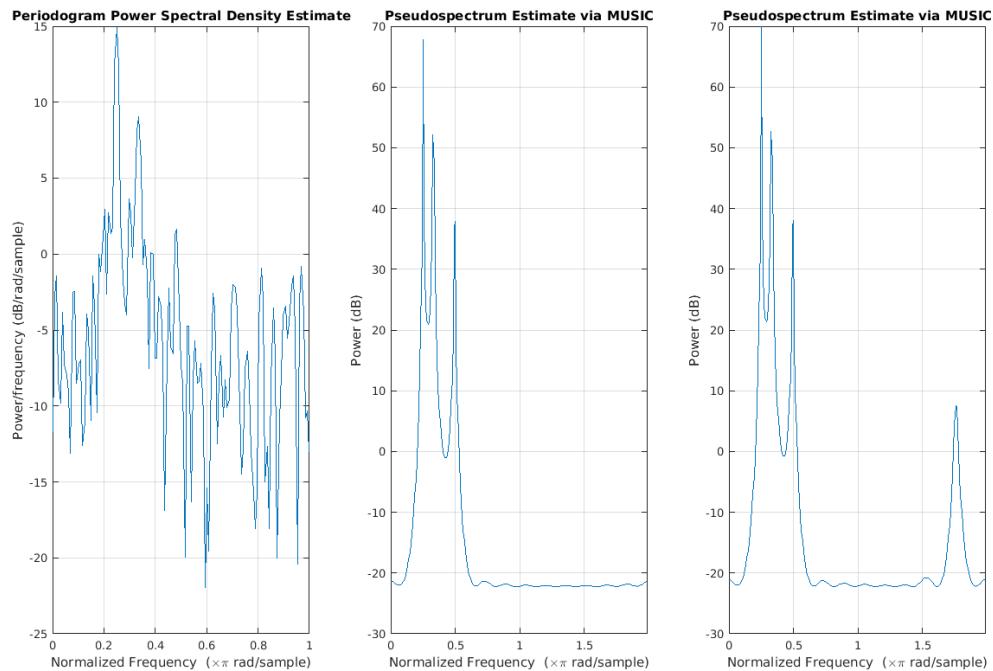


图 5.15: Synthetic signal
合成的信号, 包含四种频率成分.

5.2.3.4 智能谱估计

5.2.3.4.1 智能谱估计简介

5.2.3.4.1.1 什么是智能谱估计

5.2.4 时频分析

5.2.4.1 简介

5.2.4.1.1 What?

5.2.4.1.2 Why?

5.2.4.1.3 How?

5.2.4.2 经典时频分析

5.2.4.2.1 经典时频分析

5.2.4.2.1.1 经典时频分析

5.2.4.2.1.2 概念与内涵

5.2.4.2.1.3 经典时频分析方法

5.2.4.2.2 短时傅立叶变换

5.2.4.2.2.1 短时傅立叶变换

短时傅立叶变换 (*Short Time Fourier Transform*, STFT) 是和傅里叶变换相关的一种数学变换, 用以确定时变信号其局部区域正弦波的频率与相位。通过将长时间信号分成等长的短的片段, 计算每个片段的傅里叶变换, 将这些片段组成一个频率-时间矩阵得到。与傅里叶变换类似, 分为连续时间 (Continuous-time) STFT 和离散时间 (Discrete-time) STFT。

5.2.4.2.2.2 连续时间 STFT

设有连续时间信号 $x(t)$, 窗函数 $w(t)$, 在信号上滑动窗函数, 计算信号与窗函数乘积的傅里叶变换, 得到连续时间信号的二维时频 STFT 表示, 即

$$\text{STFT}\{x(t)\}(\tau, \omega) = X(\tau, \omega) = \int_{-\infty}^{+\infty} x(t)w(t - \tau)e^{-j\omega t} dt$$

其中, $t, \tau, \omega = 2\pi f$ 均为连续的。

提示: 信号 $x(t)$ 的傅里叶变换表示为

$$F(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t} dt$$

逆傅里叶变换表示为

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)e^{+j\omega t} d\omega$$

5.2.4.2.2.3 离散时间 STFT

设有离散时间信号 $x[n]$, 窗函数 $w[n]$, 在信号上滑动窗函数, 计算信号与窗函数乘积的傅里叶变换, 得到离散时间信号的二维时频 STFT 表示, 即

$$\text{STFT}\{x[n]\}(m, \omega) = X(m, \omega) = \sum_{n=-\infty}^{+\infty} x[n]w[n-m]e^{-j\omega n}$$

其中, n, m 是离散的, $\omega = 2\pi f$ 为连续的. 然而, 通常情况下傅里叶变换采用快速傅里叶变换 (FFT) 计算, 此时 ω 也是离散的.

提示: 通常情况下, 窗取为重叠的 (overlap),

5.2.4.2.2.4 频谱表示

信号的功率谱密度的频谱表示为 STFT 结果的平方

$$\text{spectrogram}\{x(t)\}(\tau, \omega) = |X(\tau, \omega)|^2$$

5.2.4.2.2.5 逆短时傅立叶变换

STFT 是可逆的, 即原始信号可以从变换后的数据中恢复出来.

5.2.4.2.2.6 连续时间逆 STFT

设有连续时间信号 $x(t)$ 的 STFT 表示 $X(\tau, \omega)$, 则

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X(\tau, \omega)e^{+j\omega t} d\tau d\omega$$

或

$$x(t) = \int_{-\infty}^{+\infty} \left[\frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\tau, \omega)e^{+j\omega t} d\omega \right] d\tau$$

5.2.4.2.2.7 离散时间逆 STFT

5.2.4.2.2.8 实现步骤

- 离散时间 STFT
 - 使用窗函数在信号上滑动, 将长时间信号分成等长的短的片段
 - 计算每个片段与窗函数乘积的傅里叶变换
 - 计算每个片段变换后的幅度或功率密度谱
 - 将这些片段组成一个频率-时间矩阵

5.2.4.2.2.9 实例分析

5.2.4.2.2.10 实验 1: 仿真信号

5.2.4.2.2.11 实验内容

生成四个子信号, 四种频率:

$$\begin{aligned}x_1 &= \cos(2f_1 t), (0\Delta T \leq t < 1\Delta T) \\x_2 &= \cos(2f_2 t), (1\Delta T \leq t < 2\Delta T) \\x_3 &= \cos(2f_3 t), (2\Delta T \leq t < 3\Delta T) \\x_4 &= \cos(2f_4 t), (3\Delta T \leq t < 4\Delta T)\end{aligned}$$

合成一个信号, 含上述四种频率:

$$x = x_1 + x_2 + x_3 + x_4$$

做以下实验

1. 不同窗
2. Gauss 窗, 不同 std
3. Gauss 窗, 不同窗大小
4. Gauss 窗, 不同重叠大小
5. Gauss 窗, 不同 FFT 大小

5.2.4.2.2.12 实验代码

核心函数: `scipy.signal.spectral.spectrogram`

1. 不同窗: [demo_stft_windows.py](#)
2. 不同窗, 不同窗大小: [demo_stft_gauss_windowsize.py](#)
3. Gauss 窗, 不同 std: [demo_stft_gauss_std.py](#)

4. Gauss 窗, 不同窗大小: [demo_stft_gauss_windowsize.py](#)
5. Gauss 窗, 不同重叠大小: [demo_stft_gauss_noverlap.py](#)
6. Gauss 窗, 不同 FFT 大小: [demo_stft_gauss_nfft.py](#)

5.2.4.2.2.13 实验结果

实验中, 设四种信号频率为: $f_1 = 10Hz$, $f_2 = 25Hz$, $f_3 = 50Hz$, $f_4 = 100Hz$, $\Delta T = 5s$, 采样率为 $400Hz$, 共 8000 个样本点, 生成的合成信号如下图

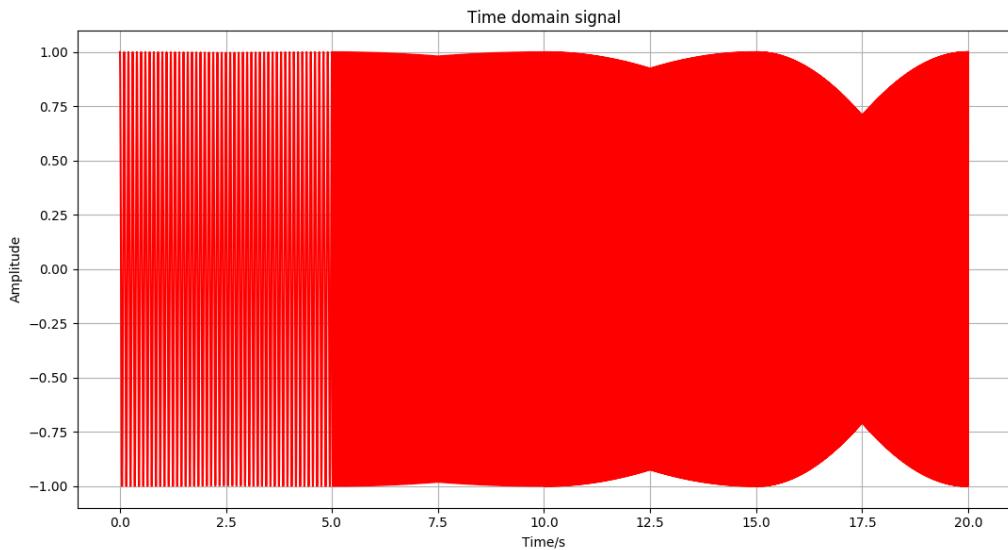


图 5.16: Synthetic signal
合成的信号, 包含四种频率成分.

1. 不同窗
2. 不同窗, 不同窗大小

对合成信号施加不同窗, 并改变窗大小, FFT 点数取 2048, 重叠点数取 1, 其它参数见图. 由图可见窗大小影响较大, 窗过小, 频率难分辨.

窗大小为 $10ms$ 的结果

窗大小为 $100ms$ 的结果

窗大小为 $1000ms$ 的结果

3. gauss 窗, 不同 std
4. gauss 窗, 不同窗大小

对于 Gauss 窗, 窗过小频率难区分, 窗过大时间难区分.

5. 不同重叠点数
6. 不同 FFT 点数

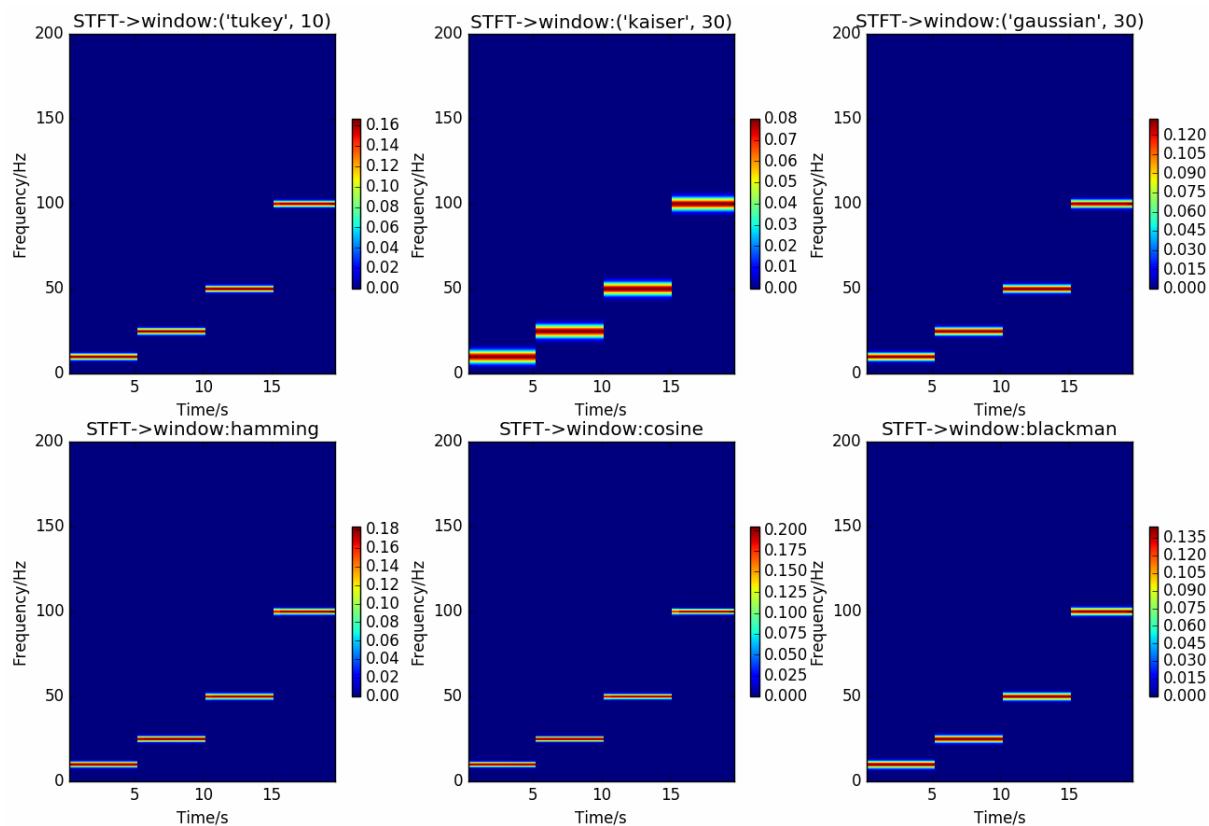


图 5.17: STFT 在不同窗下的结果

对合成信号施加六种不同的窗 (tukey, kaiser, gauss, hamming, cosine, blackman), 窗大小取 500ms , FFT 点数取 2048, 重叠点数取 1 , 然后进行 STFT 分析.

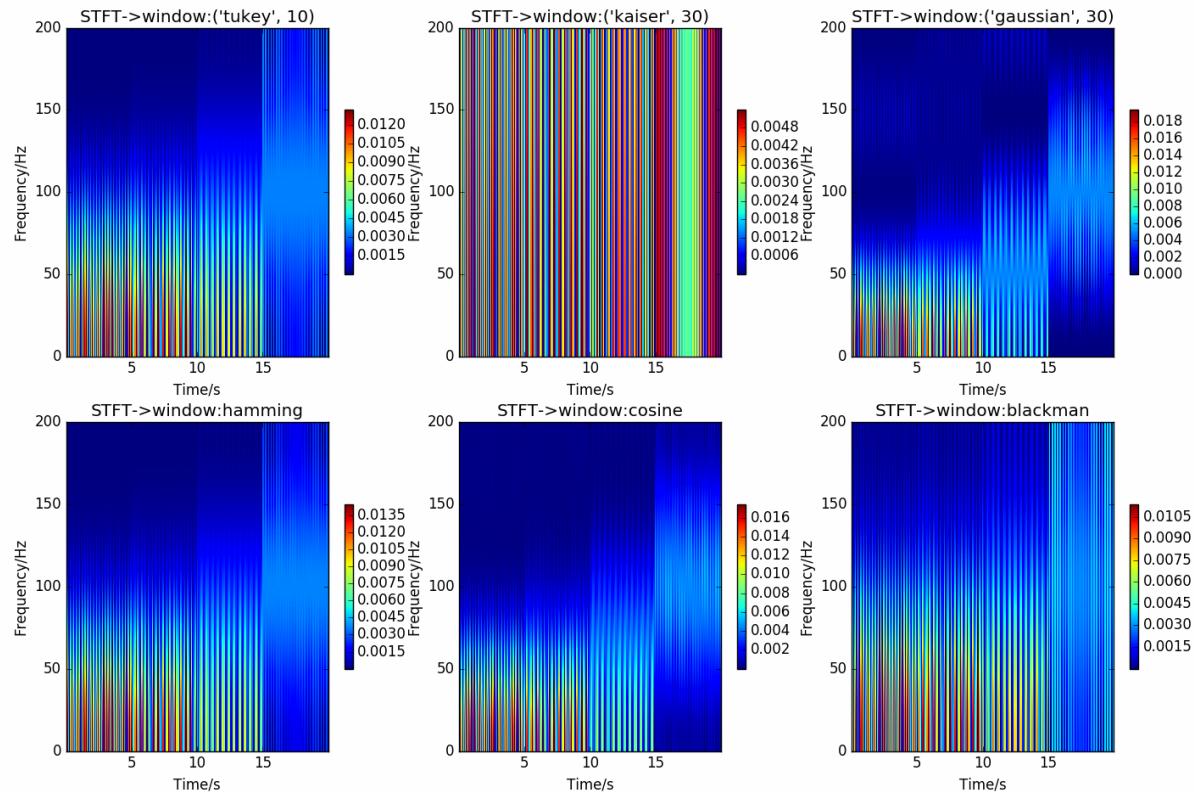


图 5.18: STFT 在窗大小为 10ms 时的结果

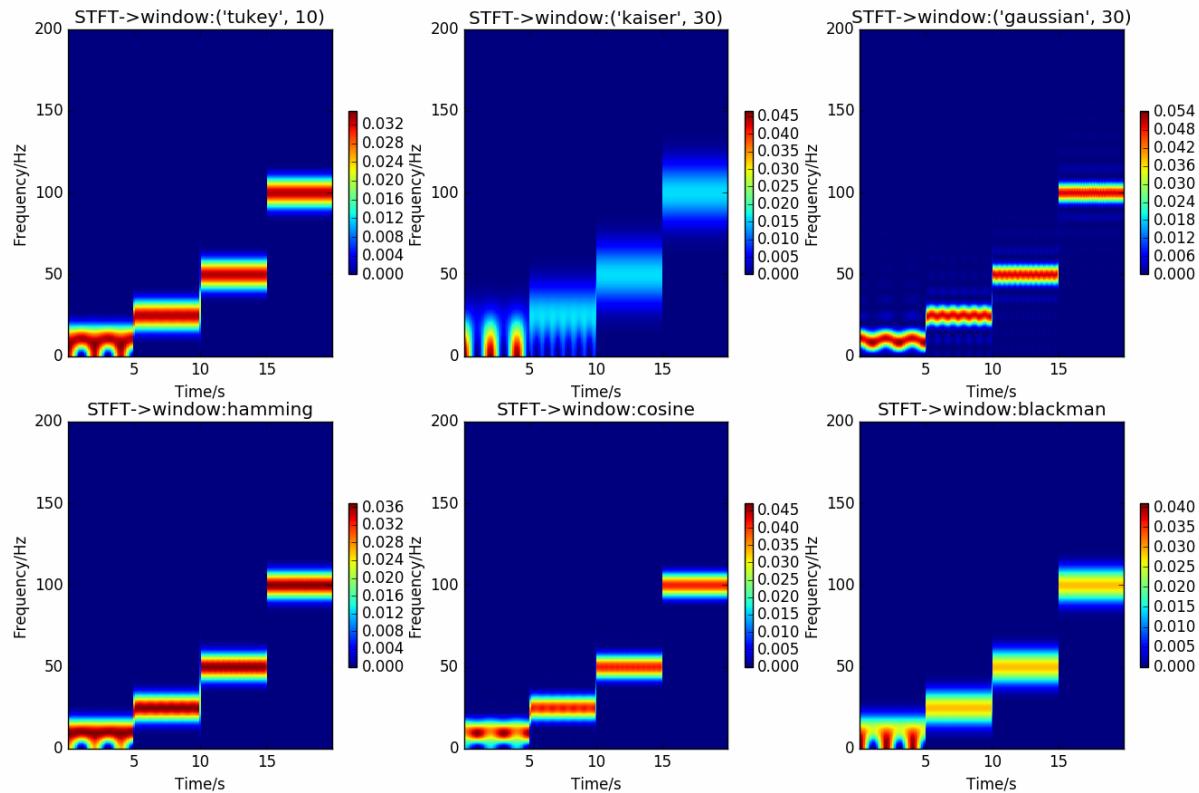


图 5.19: STFT 在窗大小为 100ms 时的结果

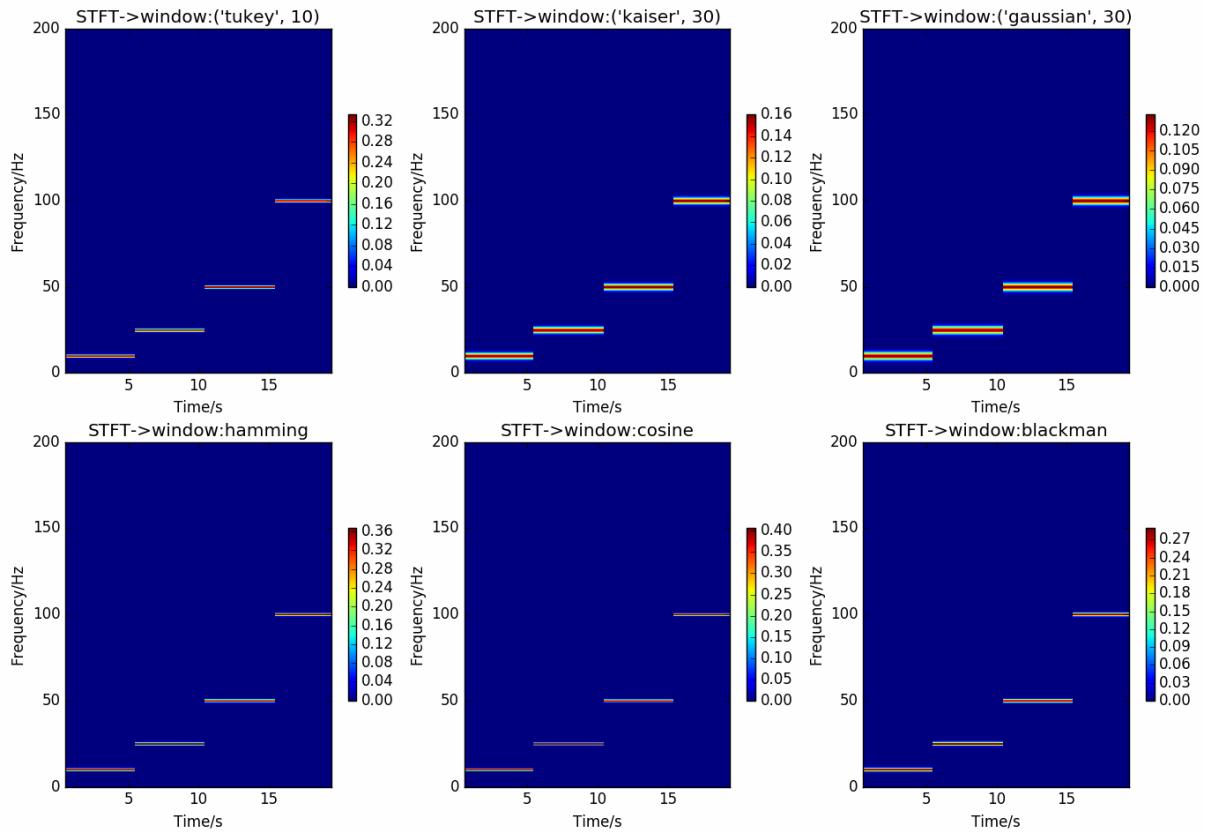


图 5.20: STFT 在窗大小为 1000ms 时的结果

5.2.4.2.2.14 实验分析

- 对于所有窗函数, 窗大小影响较大
- 重叠点数影响较小, 但不能过大
- FFT 点数影响较小, 但要大于分段长度
- 窗函数及其参数影响较大, 应仔细选取
- 对于 Gauss 窗, 窗过小频率难区分, 窗过大时间难区分
- 对于 Gauss 窗, 窗大小及标准差影响较大;
- 对于 tukey, 其参数影响较小;
- 对于 kaiser, 其参数影响较大.

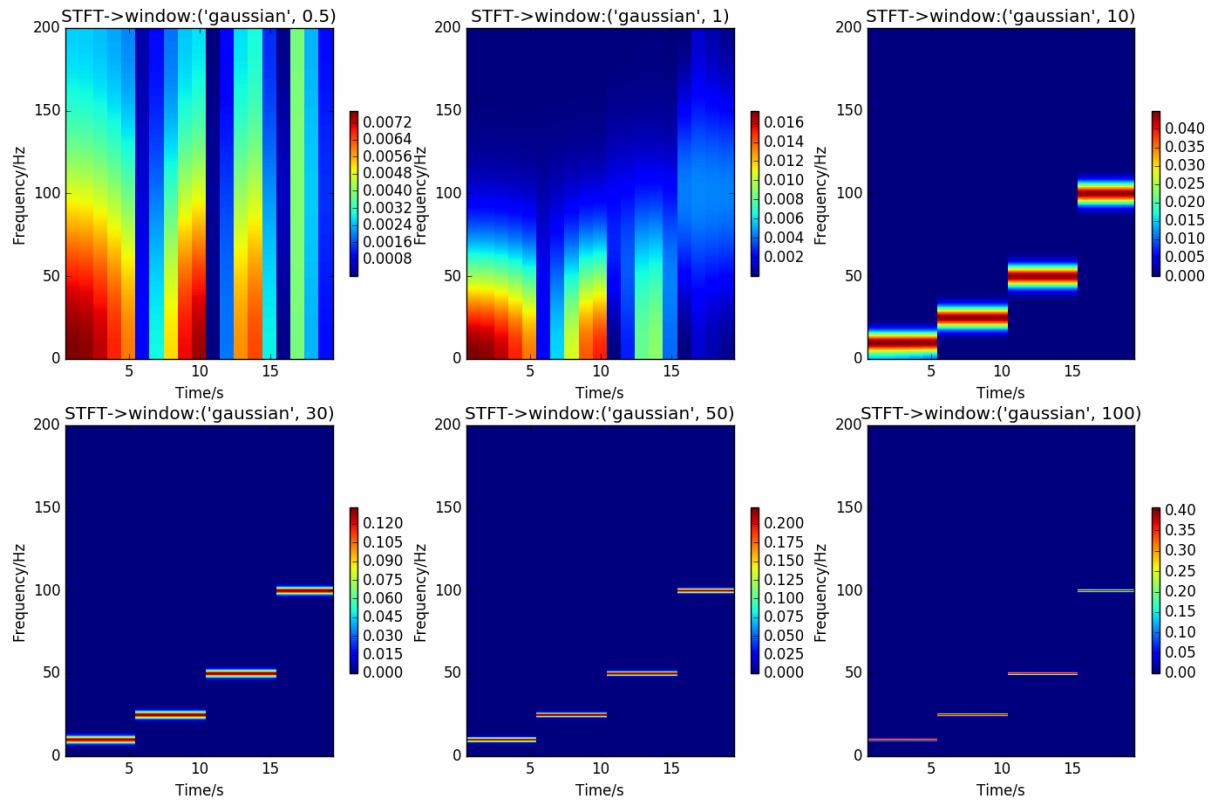


图 5.21: STFT 在 gauss 窗, 不同 std 下的结果

对合成信号施加 gauss 窗, 窗大小取 1000ms , FFT 点数取 2048, 重叠点数取 1 , 使用不同标准差 (0.5, 1.0, 10, 30, 50, 100), 然后进行 STFT 分析.

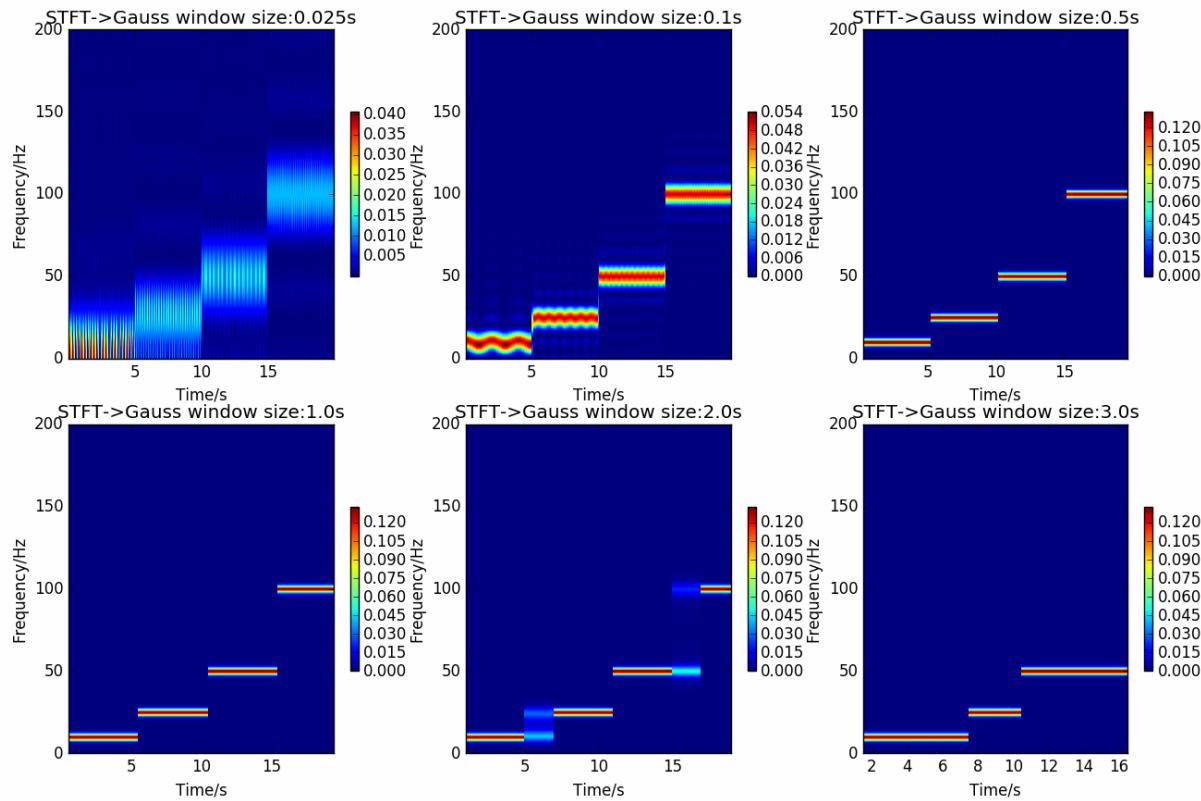


图 5.22: STFT 在 gauss 窗, 不同窗大小下的结果

对合成信号施加 gauss 窗, 标准差取 30 , FFT 点数取 2048 , 重叠点数取 1 , 使用不同窗大小 (25ms, 100ms, 500ms, 1000ms, 2000ms, 3000ms), 然后进行 STFT 分析.

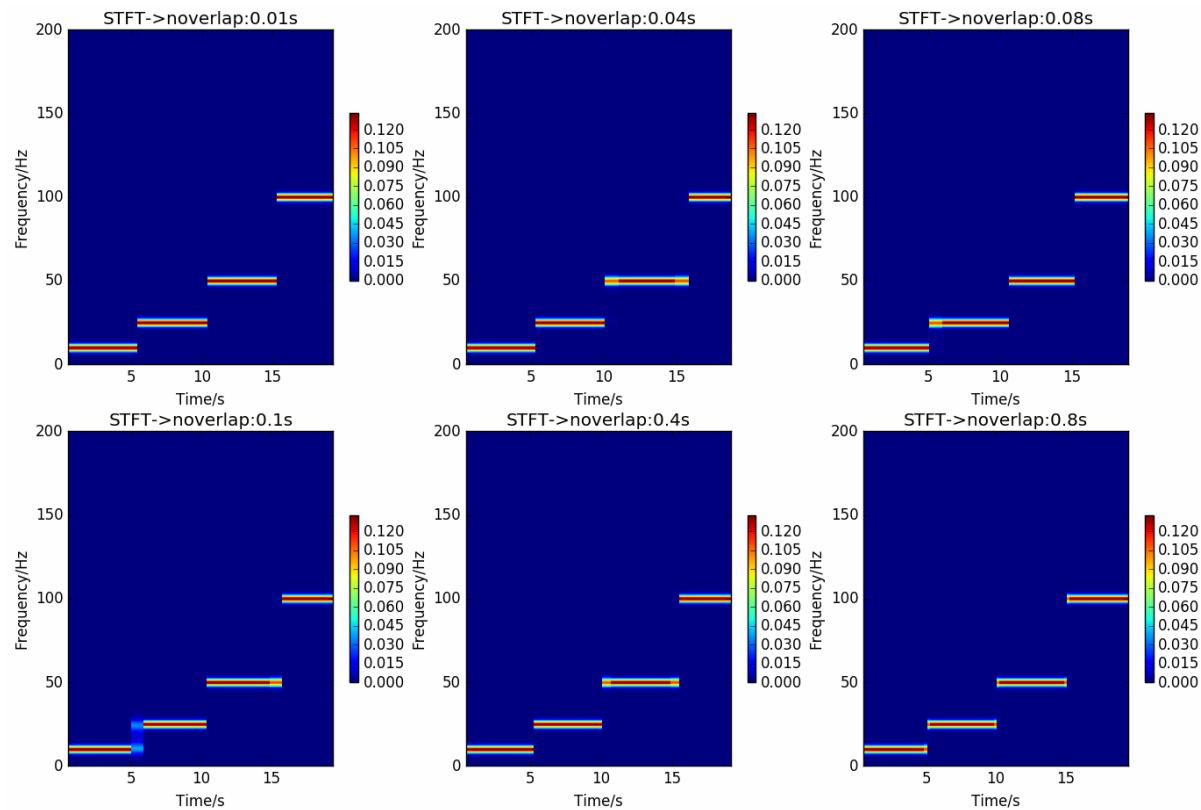


图 5.23: STFT 在不同重叠点数下的结果

对合成信号施加 gauss 窗, 标准差取 30 , 窗大小取 1000ms , FFT 点数取 2048 , 使用不同重叠点数, 然后进行 STFT 分析.

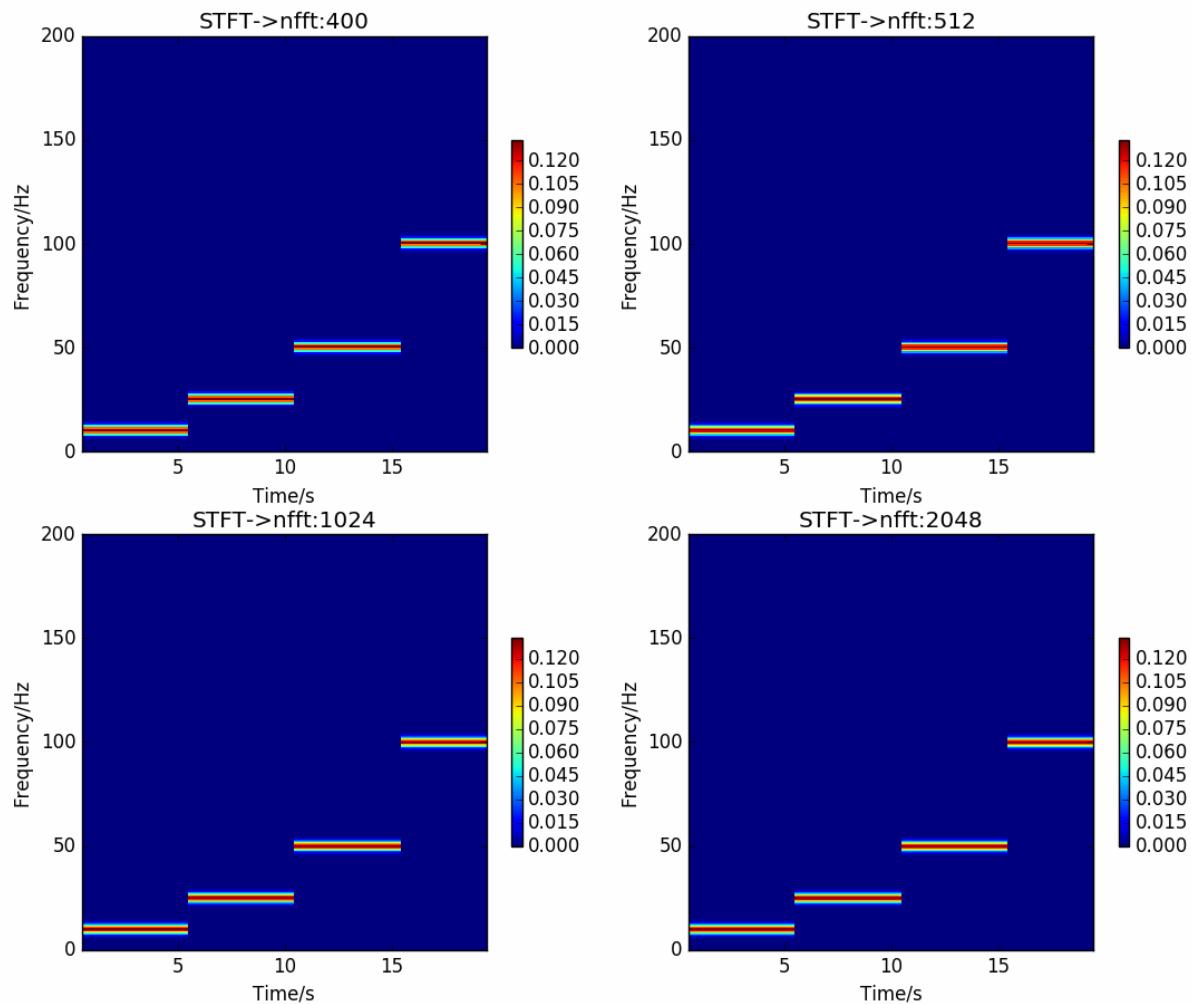


图 5.24: STFT 在不同 FFT 点数下的结果

对合成信号施加 gauss 窗, 标准差取 30, 窗大小取 1000ms, 重叠点数取 1, 使用不同 FFT 点数, 然后进行 STFT 分析.

5.2.4.2.2.15 实验 2: 真实信号

语音信号

5.2.4.2.3 小波变换

5.2.4.2.3.1 小波与小波变换

5.2.4.2.3.2 小波及其性质

小波 (*Wavelet*)，即小区域的波，是一种特殊的，长度有限的，平均值为零的波形。

特点：

- “小”：在时域具有紧支集或近似紧支集
- “波”：正负交替的“波动性”，直流分量为零
- 信号可分解为一系列由同一母小波函数经平移与尺度伸缩得到的小波函数的叠加

将小波母函数 $\psi(t)$ 进行平移与尺度伸缩，得到

$$\psi_{a,\tau}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-\tau}{a}\right), a, \tau \in \mathbb{R}, a > 0$$

由同一个小波母函数 $\psi(t)$ 进行平移与尺度伸缩后得到的一组函数序列称为 **小波函数基**。

5.2.4.2.3.3 小波变换

小波 (*Wavelet*) 是一种数学函数，用来将给定的函数或连续时间信号分解成不同的尺度分量。通常人们可以为每个尺度分量指定一个频率范围。**小波变换**是指用小波表示一个函数。小波是有限长度或快速衰减振荡波（称为母小波 *Mother Wavelet*）的缩放和平移后的波（称为子小波 *Daughter Wavelet*）。与传统的傅立叶变换相比，小波变换在表示具有不连续性和尖峰的函数以及精确解构和重构有限长，非周期或非平稳信号方面具有优势。

小波变换分为离散小波变换和连续小波变换。注意，离散小波变换和连续小波变换都是连续时间（模拟）变换。它们可以用来表示连续时间（模拟）信号。连续小波变换在所有可能的尺度和平移上执行，而离散小波变换使用尺度与平移参数的特定的量化后的子集。

5.2.4.2.3.4 尺度与频率的关系

设 a 为尺度， F_s 为采样频率， F_c 为小波中心频率，则 a 对应的实际频率 $F_a = \frac{F_c F_s}{a}$ 。

5.2.4.2.3.5 连续小波变换

连续小波变换 (*Continuous Wavelet Transform*, CWT) 提供了对信号的过完备表示, 通过取各种平移 (translation) 与缩放 (scale) 参数得到一组小波函数基.

函数 $x(t)$ 在尺度因子 $a, a > 0, a \in \mathbb{R}$, 平移因子 τ 下的连续小波变换表示为

$$\text{CWT}\{x(t)\}(a, \tau) = X(a, \tau) = \langle x(t), \psi_{a, \tau}(t) \rangle = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t) \bar{\psi} \left(\frac{t - \tau}{a} \right) dt$$

其中, ψ 为时域与频域的连续函数, 即小波母函数, $\bar{\psi}$ 表示复共轭.

5.2.4.2.3.6 连续小波种类

- Continuous wavelets

- Real-valued

- * Beta wavelet
 - * Hermitian wavelet
 - * Hermitian hat wavelet
 - * Meyer wavelet
 - * Mexican hat wavelet
 - * Poisson wavelet
 - * Shannon wavelet
 - * Spline wavelet
 - * Stromberg wavelet

- Complex-valued

- * Complex Mexican hat wavelet
 - * fbsp wavelet
 - * Morlet wavelet
 - * Shannon wavelet
 - * Modified Morlet wavelet

5.2.4.2.3.7 Mexh 小波

Mexican Hat 小波函数为 Gauss 函数的二阶导数, 即

$$\psi(t) = (1 - t^2)e^{-\frac{t^2}{2}}$$

$$\psi(\omega) = \sqrt{2\pi}\omega^2 e^{-\frac{\omega^2}{2}}$$

5.2.4.2.3.8 Morlet 小波

Morlet 小波为 Gauss 包络下的单频复正弦函数, 即

$$\psi(t) = Ce^{-\frac{t^2}{2}} \cos(5t)$$

5.2.4.2.3.9 算法步骤

- Step1: 选择小波函数及其尺度值 a
- Step2: 从信号起始位置开始, 将小波函数与信号进行比较, 计算小波系数
- Step3: 沿时间轴移动小波函数, 即改变平移参数 τ , 在新的位置计算小波系数, 直至信号终点
- Step4: 改变尺度参数 a 的值, 重复 Step2, Step3

5.2.4.2.3.10 离散小波变换

离散小波变换 (*Discrete Wavelet Transform*, DWT) 提供了对信号的过完备表示.

5.2.4.2.3.11 离散小波种类

- Discrete wavelets
 - Beylkin
 - BNC wavelets
 - Coiflet
 - Cohen-Daubechies-Feauveau wavelet (Sometimes referred to as CDF N/P or Daubechies biorthogonal wavelets)
 - Daubechies wavelet
 - Binomial-QMF (Also referred to as Daubechies wavelet)
 - Haar wavelet
 - Mathieu wavelet
 - Legendre wavelet
 - Villasenor wavelet
 - Symlet

5.2.4.2.3.12 Haar 小波

Haar 小波函数为:

$$\psi(t) = \begin{cases} 1, & 0 \leq t \leq 1/2 \\ -1, & 1/2 \leq t \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

5.2.4.2.3.13 连续逆小波变换

设有连续时间信号 $x(t)$ 的 CWT 变换表示 $X(a, \tau)$, 则

$$x(t) = \frac{1}{2\pi\hat{\psi}(1)} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{1}{a^2} X(a, \tau) e^{j\frac{t-\tau}{a}} d\tau da$$

其中, $\hat{\psi}$ 表示 ψ 的傅里叶变换.

由逆变换知小波函数应定义为

$$\psi(t) = w(t)e^{jt}$$

其中, $w(t)$ 是窗函数, 这种定义的小波称为分析小波, 因为它可以进行时频分析, 分析小波是不需要满足容许条件的.

5.2.4.2.3.14 实例分析

5.2.4.2.3.15 仿真信号

5.2.4.2.3.16 实验内容

生成四个子信号, 四种频率:

$$\begin{aligned} x_1 &= \cos(2f_1 t), (0\Delta T \leq t < 1\Delta T) \\ x_2 &= \cos(2f_2 t), (1\Delta T \leq t < 2\Delta T) \\ x_3 &= \cos(2f_3 t), (2\Delta T \leq t < 3\Delta T) \\ x_4 &= \cos(2f_4 t), (3\Delta T \leq t < 4\Delta T) \end{aligned}$$

合成一个信号, 含上述四种频率:

$$x = x_1 + x_2 + x_3 + x_4$$

做 CWT 分析

5.2.4.2.3.17 实验代码

MATLAB 代码:

PYTHON 代码:

5.2.4.2.3.18 实验结果

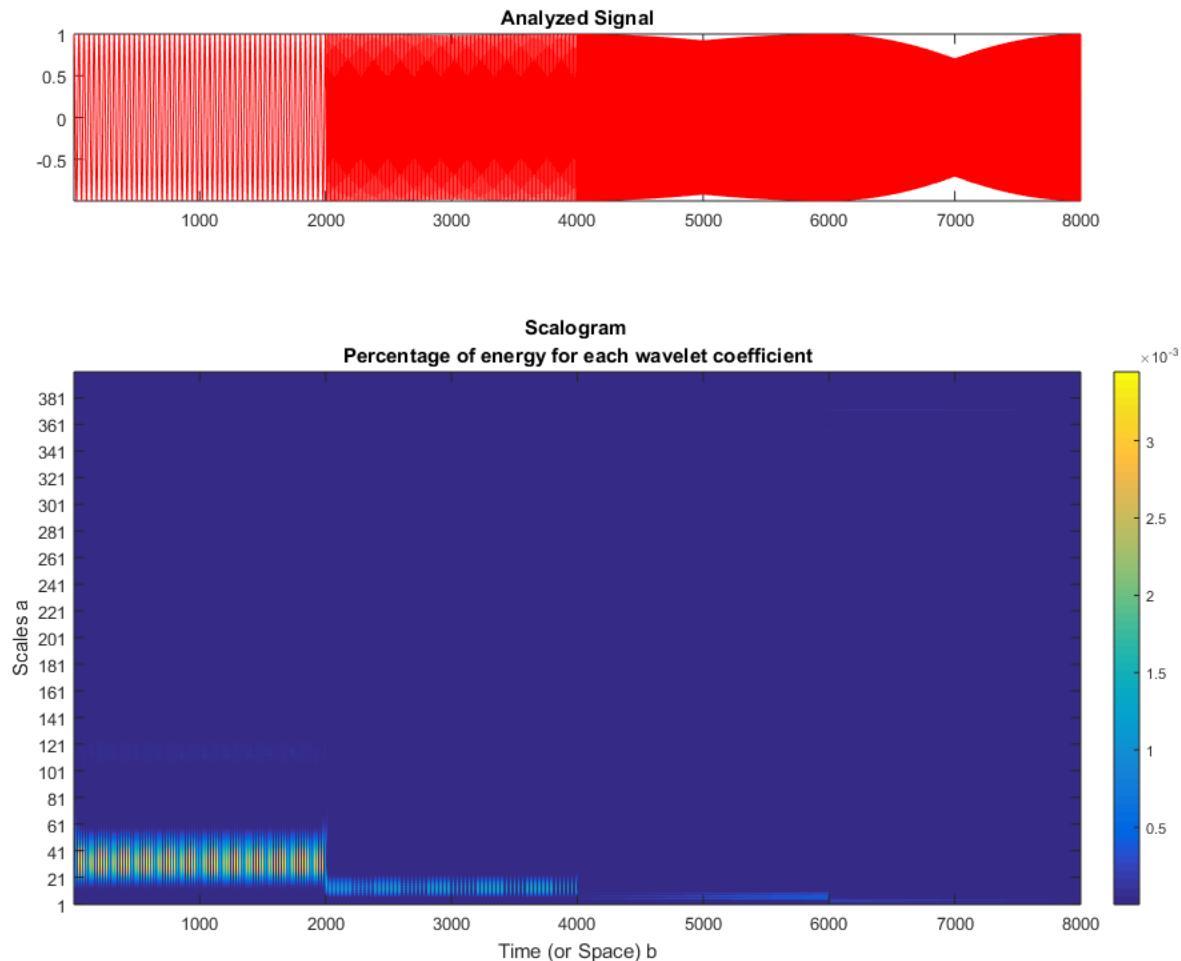


图 5.25: MATLAB CWT 结果
使用 MATLAB 对合成的信号进行 CWT 分析.

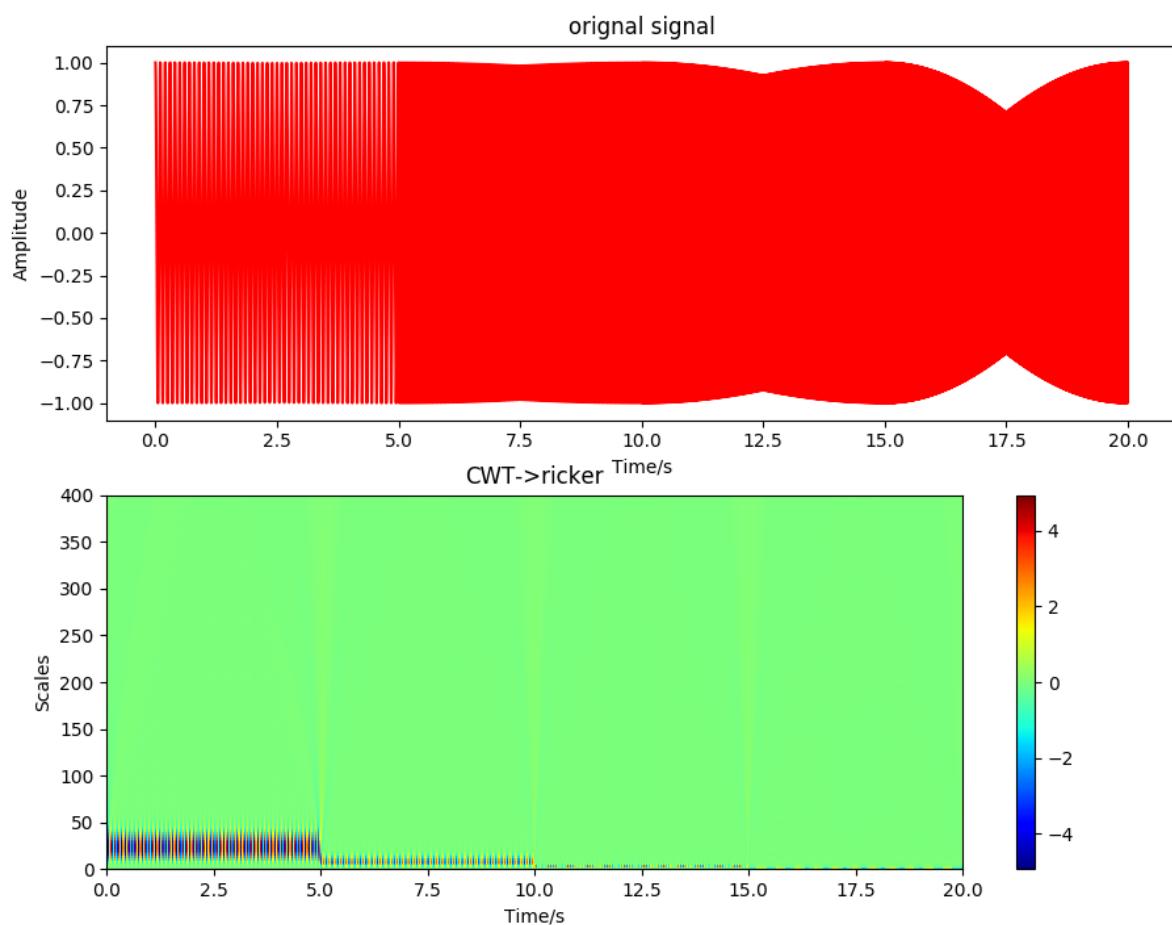


图 5.26: PYTHON CWT 结果
使用 PYTHON 对合成的信号进行 CWT 分析.

5.2.4.2.3.19 真实信号

5.2.4.2.4 S 变换

5.2.4.3 总结

5.2.4.3.1 不同时频变换间的关系

短时傅里叶变换，由于窗口大小是固定的，只适用于频率波动小的平稳信号，不适用于频率波动大的非平稳信号。而小波变换可以根据频率的高低自动调节窗口大小，是一种自适应的时频分析方法，可以进行多分辨率分析。

5.2.5 名词术语

Autoregressive Model 自回归模型 ([Autoregressive Model](http://en.volupedia.org/wiki/Autoregressive_model)) , 是统计上一种处理时间序列的方法, 是用同一变量之前各期的表现情况, 来预测该变量本期的表现情况, 并假设它们为线性关系. 因为这是从回归分析中的线性回归发展而来, 只是不是用来预测其他变量, 而是用来预测自己, 所以叫做自回归. 自回归模型被广泛运用在经济学、信息学、自然现象的预测上.

Classical Spectral Estimation 经典谱估计 (Classical Spectral Estimation), 又称非参数化谱估计.

Compact Set 紧集

Compact Support 紧支撑 ([Compact support](http://en.volupedia.org/wiki/Support_(mathematics)#Compact_support)) , 在数学中, 如果函数 f 的支撑集是拓扑空间 \mathbb{X} 的紧集, 则称函数 f 紧支撑于空间 \mathbb{X} .

Continuous Wavelet Transform 连续小波变换 ([Continuous wavelet transform](http://en.volupedia.org/wiki/Continuous_wavelet_transform)) , CWT , 由于短时傅里叶变换窗口大小是固定的, 只适用于频率波动小的平稳信号, 不适用于频率波动大的非平稳信号。而小波变换可以根据频率的高低自动调节窗口大小, 是一种自适应的时频分析方法, 可以进行多分辨率分析。

Daughter Wavelet 子小波 ([Daughter Wavelet](http://en.volupedia.org/wiki/Wavelet#Daughter_wavelet)) 由小波母函数通过平移与尺度缩放得到。

Discrete Wavelet Transform 连续小波变换 ([Discrete wavelet transform](http://en.volupedia.org/wiki/Discrete_wavelet_transform)) , DWT , 短时傅里叶变换, 但由于窗口大小是固定的, 只适用于频率波动小的平稳信号, 不适用于频率波动大的非平稳信号。而小波变换可以根据频率的高低自动调节窗口大小, 是一种自适应的时频分析方法, 可以进行多分辨率分析。

Modern Spectral Estimation 现代谱估计 (Modern Spectral Estimation), 又称参数化谱估计.

Mother Wavelet 母小波 ([Mother Wavelet](http://en.volupedia.org/wiki/Wavelet#Mother_wavelet)) 也称小波母函数.

Multiple Signal Classification 多重信号分类 ([Multiple Signal Classification](http://en.volupedia.org/wiki/MUSIC_(algorithm))) , MUSIC, 是统计上一种处理时间序列的方法, 是用同一变量之前各期的表现情况, 来预测该变量本期的表现情况, 并假设它们为线性关系. 因为这是从回归分析中的线性回归发展而来, 只是不是用来预测其他变量, 而是用来预测自己, 所以叫做自回归. 自回归模型被广泛运用在经济学、信息学、自然现象的预测上.

Nonparametric Spectral Estimation 非参数化谱估计 (Nonparametric Spectral Estimation), 又称经典谱估计.

Parametric Spectral Estimation 参数化谱估计 (Parametric Spectral Estimation), 又称现代谱估计.

Short Time Fourier Transform 短时傅里叶变换 (Short Time Fourier Transform (http://en.volupedia.org/wiki/Short-time_Fourier_transform), STFT) 是和傅里叶变换相关的一种数学变换, 用以确定时变信号其局部区域正弦波的频率与相位.

Sinc Interpolation Sinc 插值 ([Sinc Interpolation](http://en.volupedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula) (http://en.volupedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula)) 也叫 Whittaker–Shannon interpolation, Shannon’s interpolation 或 Whittaker’s interpolation.

Spectral Estimation 谱估计 (Spectral Density Estimation ([Spectral Density Estimation](http://en.volupedia.org/wiki/Spectral_density_estimation) (http://en.volupedia.org/wiki/Spectral_density_estimation))), 分为经典谱估计(非参数化谱估计), 现代谱估计(参数化谱估计).

Support 支撑 ([Support](http://en.volupedia.org/wiki/Support_(mathematics)#Compact_support) ([http://en.volupedia.org/wiki/Support_\(mathematics\)#Compact_support](http://en.volupedia.org/wiki/Support_(mathematics)#Compact_support))), 在数学中, 实值函数 f 的支撑是包含那些未映射到零的元素的域的子集. 如果 f 的域是拓扑空间, 则 f 的支撑被定义为包含所有未映射到零点的最小闭集. 这个概念在数学分析中应用非常广泛.

Time Frequency Analysis 时频分析 ([Time–frequency analysis](http://en.volupedia.org/wiki/Time–frequency_analysis) (http://en.volupedia.org/wiki/Time–frequency_analysis)), 在信号处理中, 时频分析研究时间域与频域的时频表示, 通过时频变换得到信号在时间与频率二维平面中的分布.

Wavelet 小波 ([Wavelet](http://en.volupedia.org/wiki/Wavelet) (<http://en.volupedia.org/wiki/Wavelet>)) 顾名思义, “小波”就是小的波形. 所谓“小”是指它具有衰减性; 而称之为“波”则是指它的波动性, 其振幅正负相间的震荡形式. 与 Fourier 变换相比, 小波变换是时间(空间)频率的局部化分析, 它通过伸缩平移运算对信号(函数)逐步进行多尺度细化, 最终达到高频处时间细分, 低频处频率细分, 能自动适应时频信号分析的要求, 从而可聚焦到信号的任意细节, 解决了 Fourier 变换的困难问题, 成为继 Fourier 变换以来在科学方法上的重大突破. 有人把小波变换称为“数学显微镜”.

Wavelet Transform 连续小波变换 ([Wavelet transform](http://en.volupedia.org/wiki/Wavelet_transform) (http://en.volupedia.org/wiki/Wavelet_transform), CWT), 由于短时傅里叶变换窗口大小是固定的, 只适用于频率波动小的平稳信号, 不适用于频率波动大的非平稳信号. 而小波变换可以根据频率的高低自动调节窗口大小, 是一种自适应的时频分析方法, 可以进行多分辨率分析.

5.3 非线性信号处理

5.4 统计信号处理

5.4.1 引言

涉及估计, 滤波等.

5.4.2 估计理论

5.4.2.1 简介

5.4.3 滤波理论

5.4.4 名词术语

Markup Language 嘿嘿嘿

5.5 稀疏信号处理

5.5.1 引言

涉及稀疏性, 压缩感知等理论.

5.5.1.1 资源库

- [spams](http://spams-devel.gforge.inria.fr/) (<http://spams-devel.gforge.inria.fr/>) : a SPArse Modeling Software, Support :label:'Matlab', :label:'Python'
- [KSVD](https://www.cnblogs.com/endlesscoding/p/10090866.html) (<https://www.cnblogs.com/endlesscoding/p/10090866.html>)

5.5.2 稀疏采样与重构

5.5.2.1 稀疏性分析

5.5.2.1.1 稀疏信号

Definition 49 (稀疏信号 (Sparse Signal)) 若信号 x 仅含有 k 个非零元素, 即 $\|x\|_0 \leq k$, 则称该信号为 k 稀疏的. 对于非稀疏信号 x , 若存在一组基 Ψ , 使得 $x = \Psi\alpha$, 且 $\|\alpha\|_0 \leq k$, 此时, 仍称该信号为 k 稀疏的. 进一步地, 定义 k 稀疏信号集合

$$\mathbb{V}_k = \{x | \|x\|_0 \leq k\}.$$

警告: 在数字信号处理中, 信号常常由向量表示, 是否有其它表示方法?

5.5.2.2 采样

5.5.2.3 重构

5.5.2.3.1 有限等距性

Candes 等人于 2005 年提出 **有限等距常数** (Restricted Isometry Constant, RIC) [12]，文中给出的定义如下

Definition 50 (有限等距常数 (Restricted Isometry Constant)) 设矩阵 \mathbf{A} 由向量集合 $\{\mathbf{a}_i \in \mathbb{R}^p, i \in \mathbb{I}\}$ 构成。对于每个整数 $1 \leq s \leq |\mathbb{I}|$ ，定义 \mathbf{A} 的 s 阶有限等距常数 δ_s 为使得 $\mathbf{A}_{\mathbb{T}}$ 满足如下条件的最小正数

$$(1 - \delta_s) \|\mathbf{x}\| \leq \|\mathbf{A}_{\mathbb{T}} \mathbf{x}\| \leq (1 + \delta_s) \|\mathbf{x}\| \quad \forall \mathbb{T} \subset \mathbb{I} \quad (5.1)$$

其中， $\mathbf{x} \in \mathbb{R}_k^n$ 为 k 稀疏信号。 $|\mathbb{I}|$ 为集合 \mathbb{I} 的势 (Cardinality)。

RIP 被广泛用于压缩感知 (Compressive Sensing, CS) 中 [7]。在 CS 中，感知矩阵 ($\mathbf{A} = \Phi \Psi$) 是否满足 RIP 条件直接决定了重构信号质量。在 CS 中，矩阵 \mathbf{A} 的有限等距常数定义为

Definition 51 (有限等距常数 (Restricted Isometry Constant)) 设 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, $\mathbf{a}_i \in \mathbb{R}^m, i = 1, 2, \dots, n$. 对于每个整数 $s \in [1, n]$, 定义矩阵 \mathbf{A} 的有限等距常数为满足下式的最小的数 δ_s

$$(1 - \delta_s) \|\mathbf{x}\| \leq \|\mathbf{A} \mathbf{x}\| \leq (1 + \delta_s) \|\mathbf{x}\| \quad (5.2)$$

其中， $\mathbf{x} \in \mathbb{R}_k^n$ 为 k 稀疏信号。

矩阵 \mathbf{A} 具有有限等距特性 (Restricted Isometry Property, RIP) [6] 是指对于充分小的 δ_s ，矩阵 \mathbf{A} 满足式 5.2。式 5.2 表明，矩阵 \mathbf{A} 对信号 \mathbf{x} 投影前后的长度仅有微小的改变。即 RIP 特性保证了来自矩阵 \mathbf{A} 的投影可以保持两个信号 $\mathbf{x}_1, \mathbf{x}_2$ 之间的距离，式 5.2 可以等价地表示为

$$(1 - \delta_s) \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\| \leq (1 + \delta_s) \|\mathbf{x}_1 - \mathbf{x}_2\|. \quad (5.3)$$

RIP 保证了稀疏信号 \mathbf{x} 可以被以高概率重构。

5.5.3 表示字典

5.5.3.1 字典的类型

5.5.3.1.1 什么是字典

5.5.3.1.2 字典的种类

5.5.3.1.2.1 完备与过完备

对于给定的信号 $\mathbf{y} \in \mathbb{R}^{M \times N}$ ，期望找到一组过完备基 $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N], \mathbf{d}_i \in \mathbb{R}^{M \times 1}$ ，和一组系数 \mathbf{x} ，满足

$$\mathbf{y} = \mathbf{D} \mathbf{x} = x_1 \mathbf{d}_1 + x_2 \mathbf{d}_2 + \dots + x_N \mathbf{d}_N$$

当基的个数 N 大于信号 \mathbf{y} 的维度 M 时，称基/字典 \mathbf{D} 是 **过完备字典** (Overcomplete Dictionary)；当基的个数 N 等于信号 \mathbf{y} 的维度 M 时，称基/字典 \mathbf{D} 是 **完备字典** (Complete Dictionary)。

5.5.3.1.2.2 冗余与非冗余

5.5.3.1.2.3 正交与非正交

5.5.3.1.2.4 总结

注解:

- 完备字典是指字典中的原子张成的空间可以覆盖整个信号空间.
- 冗余字典是指字典中存在线性相关的原子.
- 冗余字典可能不完备, 完备字典有可能冗余.

5.5.3.2 离散余弦变换类字典

5.5.3.2.1 什么是离散余弦变换

离散余弦变换 (*Discrete Cosine Transform*, DCT) 将有限长序列信号表示成不同频率的余弦函数之和, 被广泛用于音频与图像的压缩. 目前共有 DCT-I, DCT-II, DCT-III, DCT-IV, DCT-V-VIII 五种变种². DCT 在图像压缩上的应用内容参见¹. 最常见的是 DCT-II 型, 下面进行介绍.

5.5.3.2.2 一维离散余弦变换

[link text](https://blog.csdn.net/wyw921027/article/details/52102264) (<https://blog.csdn.net/wyw921027/article/details/52102264>)

[link text](http://fourier.eng.hmc.edu/e161/lectures/dct/node1.html) (<http://fourier.eng.hmc.edu/e161/lectures/dct/node1.html>)

离散余弦变换 (*Discrete Cosine Transform*, DCT) 最早由 Ahmed 等人 [8] 于 1974 年提出, 他们发现 DCT 的基提供了对 Toeplitz 矩阵特征向量的良好近似. 序列信号 $x[k], k = 0, \dots, N - 1$ 的 DCT-II 的离散余弦变换定义如下

$$y[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{(2n+1)k\pi}{2N}, \quad (5.4)$$

其中, $y[k]$ 为变换后的系数序列. 对 equ-DCTII-1D 中的结果分别乘以 $1/\sqrt{2}$ ($k = 0$) 和 $\sqrt{2/N}$ ($k = 1, \dots, N - 1$) 可以得到正交版本的 DCT 变换.

Definition 52 (离散余弦变换 (Discrete Cosine Transformation)) 离散序列 $x[n], n = 0, 1, \dots, N - 1$ 的 DCT 变换定义为

$$y[k] = \text{DCT}(x[n]) = \begin{cases} \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x[n] \frac{1}{\sqrt{2}}, & k = 0 \\ \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x[n] \cos \frac{(2n+1)k\pi}{2N}, & k = 1, \dots, N - 1 \end{cases} \quad (5.5)$$

用矩阵表示为

$$\mathbf{y} = \mathbf{D}\mathbf{x} \quad (5.6)$$

² http://en.wikipedia.org/wiki/Discrete_cosine_transform

¹ http://c.csie.org/~itct/slide/DCT_larry.pdf

其中, $\mathbf{x} = (x_n)_{N \times 1}$, $x_n = x[n]$, $\mathbf{D} = (d_{ij})_{N \times N}$ 表示为

$$\mathbf{D} = \sqrt{\frac{2}{N}} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & \cdots & 1/\sqrt{2} \\ \cos \frac{\pi}{2N} & \cos \frac{3\pi}{2N} & \cos \frac{5\pi}{2N} & \cdots & \cos \frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(N-1)\pi}{2N} & \cos \frac{3(N-1)\pi}{2N} & \cos \frac{5(N-1)\pi}{2N} & \cdots & \cos \frac{(2N-1)(N-1)\pi}{2N} \end{bmatrix} \quad (5.7)$$

向量 \mathbf{x} 为信号 $\mathbf{y} = (y_k)_{N \times 1}$, $y_k = y[k]$ 在基 $\{1/\sqrt{2}, \cos \frac{(2n+1)k\pi}{2N}\}$ 下的坐标, 亦即离散余弦变换系数, 代表频率成分(与傅立叶变换一致). 变换前后的数据维度不变.

易知, 一维离散余弦变换矩阵 \mathbf{D} 为正交矩阵 ($\mathbf{DD}^T = \mathbf{D}^T \mathbf{D} = \mathbf{I}$), 故逆一维离散余弦变换可以通过下式计算

$$\mathbf{x} = \mathbf{D}^{-1} \mathbf{y} = \mathbf{D}^T \mathbf{y} \quad (5.8)$$

5.5.3.2.3 多维离散余弦变换

根据一维离散余弦变换的定义, 很容易将其扩展成多维形式, 记一个 L 维序列信号为 x_{k_1, k_2, \dots, k_L} , 其中 $k_l = 0, 1, \dots, N_l - 1$, $l = 1, 2, \dots, L$, N_l 为信号在第 l 维的样点数. 则扩展后的多维 DCT-II 变换可表示为

$$\begin{aligned} y_{k_1, k_2, \dots, k_L} &= \sum_{n_1=0}^{N_1-1} \cdots \left(\sum_{n_L=0}^{N_L-1} x_{n_1, \dots, n_L} \cos \frac{(2n_L + 1)k_L \pi}{2N_L} \right) \cdots \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \\ &= \sum_{n_1=0}^{N_1-1} \cdots \sum_{n_L=0}^{N_L-1} x_{n_1, \dots, n_L} \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \cdots \cos \frac{(2n_L + 1)k_L \pi}{2N_L}. \end{aligned} \quad (5.9)$$

如二维的 DCT 变换可表示为

$$\begin{aligned} y_{k_1, k_2} &= \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos \frac{(2n_2 + 1)k_2 \pi}{2N_2} \right) \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \\ &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \cos \frac{(2n_2 + 1)k_2 \pi}{2N_2} \\ &= \sum_{n_2=0}^{N_2-1} \left(\sum_{n_1=0}^{N_1-1} x_{n_1, n_2} \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \right) \cos \frac{(2n_2 + 1)k_2 \pi}{2N_2} \\ &= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \cos \frac{(2n_2 + 1)k_2 \pi}{2N_2} \cos \frac{(2n_1 + 1)k_1 \pi}{2N_1} \end{aligned} \quad (5.10)$$

记 $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2}$, $\mathbf{D}_1 \in \mathbb{R}^{N_1 \times N_1}$ 为第一维上的 DCT 变换, $\mathbf{D}_2 \in \mathbb{R}^{N_2 \times N_2}$ 为第二维上的 DCT 变换, $\mathbf{D}_1, \mathbf{D}_2$ 均由式.?? 生成, $\mathbf{Y} \in \mathbb{R}^{N_1 \times N_2}$ 为二维 DCT 变换后的系数矩阵, 则

$$\mathbf{Y} = \mathbf{D}_1 \mathbf{X} \mathbf{D}_2^T \quad (5.11)$$

设 \mathbf{X} 为一二维矩阵, 记 $\text{DCT}(\mathbf{X}, 1)$ 表示对 \mathbf{X} 的第一维(列)进行 DCT 变换, $\text{DCT}(\mathbf{X}, 2)$ 表示对 \mathbf{X} 的第二维(行)进行 DCT 变换, 则矩阵 \mathbf{X} 的二维 DCT 变换可以表示为

$$\mathbf{Y} = \text{DCT2}(\mathbf{X}) = \text{DCT}(\text{DCT}(\mathbf{X}, 1), 2) = \text{DCT}(\text{DCT}(\mathbf{X}, 2), 1) \quad (5.12)$$

逆二维离散余弦变换可以表示为

$$\mathbf{X} = \text{IDCT2}(\mathbf{Y}) = \text{IDCT}(\text{IDCT}(\mathbf{Y}, 1), 2) = \text{IDCT}(\text{IDCT}(\mathbf{Y}, 2), 1) \quad (5.13)$$

5.5.3.2.4 过完备 DCT 字典

在过完备字典 (参见 Section-DictionaryType) 中, 原子 (列) 间是线性相关的, 存在冗余原子, 原子的数目 N 大于原子的维度 M . 式.?? 所表示的 DCT 基矩阵是一个完备字典, 那么如何得到过完备的 DCT 字典呢? 可以通过对对其进行上采样和调制得到.

对于给定的信号 $\mathbf{y} \in \mathbb{R}^{M \times 1}$, 期望找到一组过完备基 $\mathbf{D} = [\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{N-1}]$ 和一组系数 $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]$, 满足

$$\mathbf{y} = \mathbf{D}\mathbf{x} = x_0\mathbf{d}_0 + x_1\mathbf{d}_1 + \dots + x_{N-1}\mathbf{d}_{N-1}$$

其中, $\mathbf{d}_i = \{1/\sqrt{2}, \dots, \cos \frac{(2i+1)k\pi}{2N}, \dots, \cos \frac{(2i+1)(M-1)\pi}{2N}\} \in \mathbb{R}^{M \times 1}$ 为 DCT 基向量, $i = 0, 1, \dots, N-1$, $k = 1, \dots, M-1$, 过完备 DCT 字典 $\mathbf{D} \in \mathbb{R}^{M \times N}$, 可表示为

$$\mathbf{D} = \sqrt{\frac{2}{N}} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & \cdots & 1/\sqrt{2} \\ \cos \frac{\pi}{2N} & \cos \frac{3\pi}{2N} & \cos \frac{5\pi}{2N} & \cdots & \cos \frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(M-1)\pi}{2N} & \cos \frac{3(M-1)\pi}{2N} & \cos \frac{5(M-1)\pi}{2N} & \cdots & \cos \frac{(2N-1)(M-1)\pi}{2N} \end{bmatrix} \quad (5.14)$$

通常, 会对字典的列进行归一化, 归一化后的字典可以表示为

$$\mathbf{D} = \left[\frac{\mathbf{d}_0}{\|\mathbf{d}_0\|_2}, \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|_2}, \dots, \frac{\mathbf{d}_{N-1}}{\|\mathbf{d}_{N-1}\|_2} \right] \quad (5.15)$$

5.5.3.2.4.1 一维过完备 DCT 字典

一维过完备 DCT 字典的构建较为简单, 只需按照式.?? 生成矩阵, 再按照式.?? 对字典的每一列 (原子) 进行归一化即可.

5.5.3.2.4.2 多维过完备 DCT 字典

多维过完备 DCT 字典的构建以一维过完备 DCT 字典为基础, 设矩阵 $\mathbf{D}_{1d} \in \mathbb{R}^{M \times N}$ 为一维归一化过完备 DCT 字典, 则二维 DCT 字典 \mathbf{D}_{2d} 可以通过下式计算

$$\mathbf{D}_{2d} = \mathbf{D}_{1d} \times \mathbf{D}_{1d}, \quad (5.16)$$

其中, \times 表示 Kronecker 积 (页 99) (Kronecker Product). 依次类推可以得到多维过完备 DCT 字典

$$\mathbf{D}_{nd} = \mathbf{D}_{(n-1)d} \times \mathbf{D}_{(n-1)d}. \quad (5.17)$$

提示: 按此规则生成的字典 \mathbf{D}_{nd} 大小为 $M^n \times N^n$, 每一列代表一个原子, 将原子重新排成 n 维的矩阵即可.

5.5.3.2.5 实验与分析

5.5.3.2.5.1 一维离散余弦变换

5.5.3.2.5.2 原始定义

Python 实现

```

1 def dct1(x):
2     x = np.array(x)
3     N = np.size(x)
4
5     y = np.zeros(N)
6
7     y[0] = np.sum(x) / np.sqrt(N)
8
9     for k in range(1, N):
10         d = np.cos(
11             np.linspace(0, N, N, endpoint=False) * 2 + 1) * k * np.pi / (2.0 * N)
12         y[k] = np.sqrt(2.0 / N) * np.sum(x * d)
13
14 return y

```

设置序列 $x[n] = n, n = 0, 1, 2, \dots, 5$, 调用上述函数以及 Matlab 中的 `dct` 函数, 得实验结果

Matlab 结果

```

>> x=[0,1,2,3,4,5]

x =
0     1     2     3     4     5

>> dct(x)

ans =
6.1237   -4.1626      0    -0.4082      0    -0.0801

```

Python 结果

```

x = [0, 1, 2, 3, 4, 5]
y = dct(x)
print(y)

>> [ 6.12372436e+00 -4.16256180e+00 -1.53837015e-15 -4.08248290e-01 -1.53837015e-
    ↪15 -8.00788912e-02]

```

5.5.3.2.5.3 矩阵法实现

Matlab 实现构造 DCT 变换矩阵

```
1 x = [0, 1, 2, 3, 4, 5]
2
3 n = length(x)
4
5 [cc,rr] = meshgrid(0:n-1);
6
7 T = sqrt(2 / n) * cos(pi * (2*cc + 1) .* rr / (2 * n));
8 T(1,:) = T(1,:) / sqrt(2);
9
10 y = T * x;
11 disp(y)
```

Python 实现构造 DCT 变换矩阵

```
1 x = [0, 1, 2, 3, 4, 5]
2
3 N = np.size(x)
4
5 r, c = np.mgrid[0: N, 0: N]
6
7 T = np.sqrt(2 / N) * np.cos(np.pi * (2 * c + 1) * r / (2 * N))
8 T[0, :] = T[0, :] / np.sqrt(2)
9
10 y = np.matmul(T, x)
11 print(y)
```

5.5.3.2.5.4 图像的 1 维 DCT 变换

5.5.3.2.5.5 二维离散余弦变换

5.5.3.2.5.6 仿真数据

Matlab 结果

```
1 a =
2
3     0      1      2
4     3      4      5
5
6 >> dct2(a)
7
8 ans =
```

(下页继续)

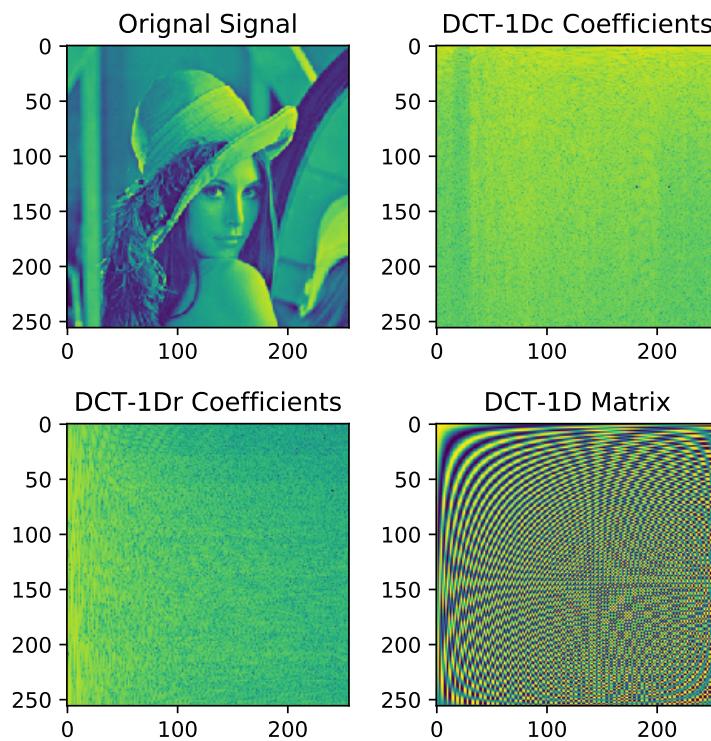


图 5.27: lena 图像的 1 维 DCT 变换

(续上页)

```

10      6.1237   -2.0000   -0.0000
11     -3.6742        0        0

```

```

1 def dct1(x, axis=0):
2     x = np.array(x)
3     if np.ndim(x) > 1:
4         N = np.size(x, axis=axis)
5         T = dctmat(N)
6         if axis is 0:
7             return np.matmul(T, x)
8         if axis is 1:
9             return np.matmul(T, x.transpose()).transpose()
10    if np.ndim(x) is 1:
11        N = np.size(x)
12        T = dctmat(N)
13        return np.matmul(T, x)
14
15 def dct2(X):
16     return dct1(dct1(X, axis=0), axis=1)
17
18 X = [[0, 1, 2], [3, 4, 5]]
19 Y = dct2(X)
20 print(Y)

```

(下页继续)

(续上页)

```

21
22 >> [[ 6.12372436e+00 -2.00000000e+00  5.32490308e-16]
23      [-3.67423461e+00 -3.44458101e-16 -2.51493529e-16]]

```

5.5.3.2.5.7 真实图像数据

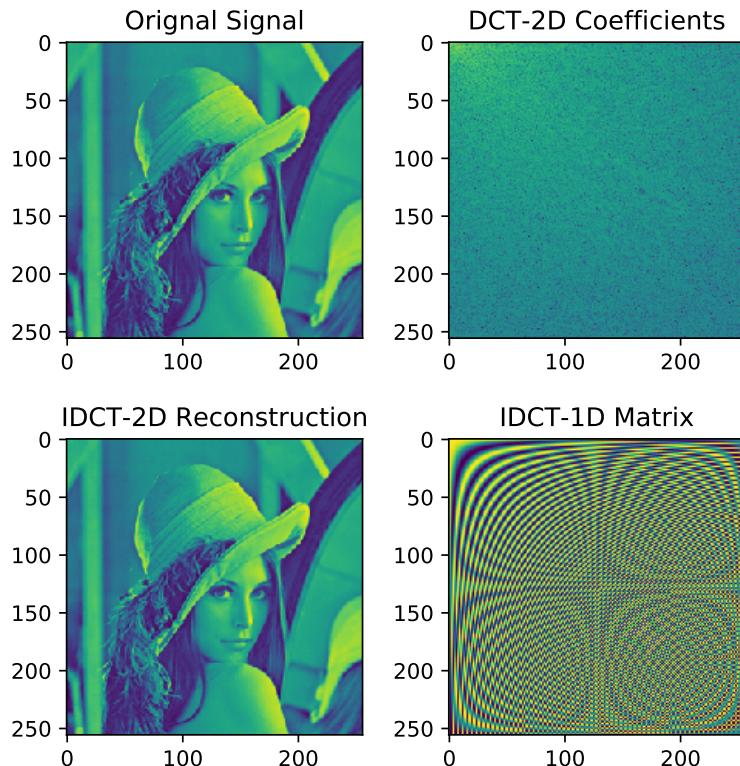


图 5.28: lena 图像二维 DCT 变换及 IDCT 变换

5.5.3.2.5.8 DCT 过完备字典

Python 实现参见文件 [demo_odctdict.py](https://github.com/antsfamily/pysparse/tree/master/examples/representation/dct/demo_odctdict.py) (https://github.com/antsfamily/pysparse/tree/master/examples/representation/dct/demo_odctdict.py)

生成大小分别为 $16 \times 8, 16 \times 16, 16 \times 32, 16 \times 64$ 的一维过完备 DCT 字典, 基于这些字典, 根据式.?? 生成的二维过完备字典.

依次将由式.?? 生成的二维过完备字典的每一列 (原子) $d_i \in \mathbb{R}^{256 \times 1}$ reshape 成 16×16 大小的矩阵, 并按顺序从左至右从上至下排列成方阵

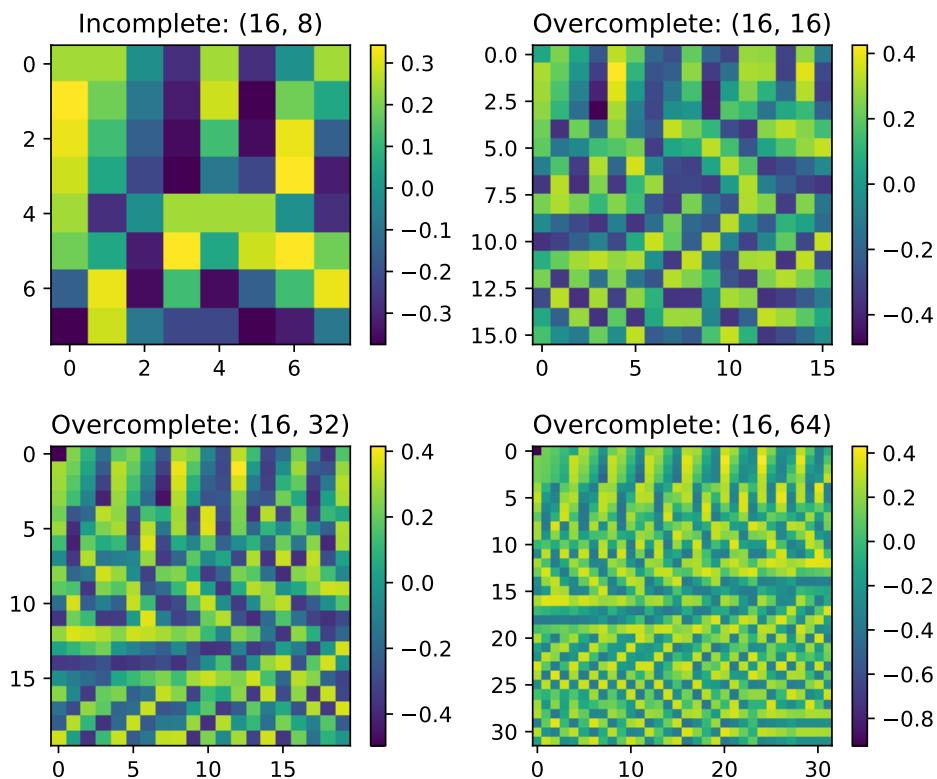


图 5.29: 1-dimension overcomplete DCT dictionary. These four dictionary are generated from 1-dimension DCT dictionary with size of 16×8 , 16×16 , 16×32 , 16×64 .

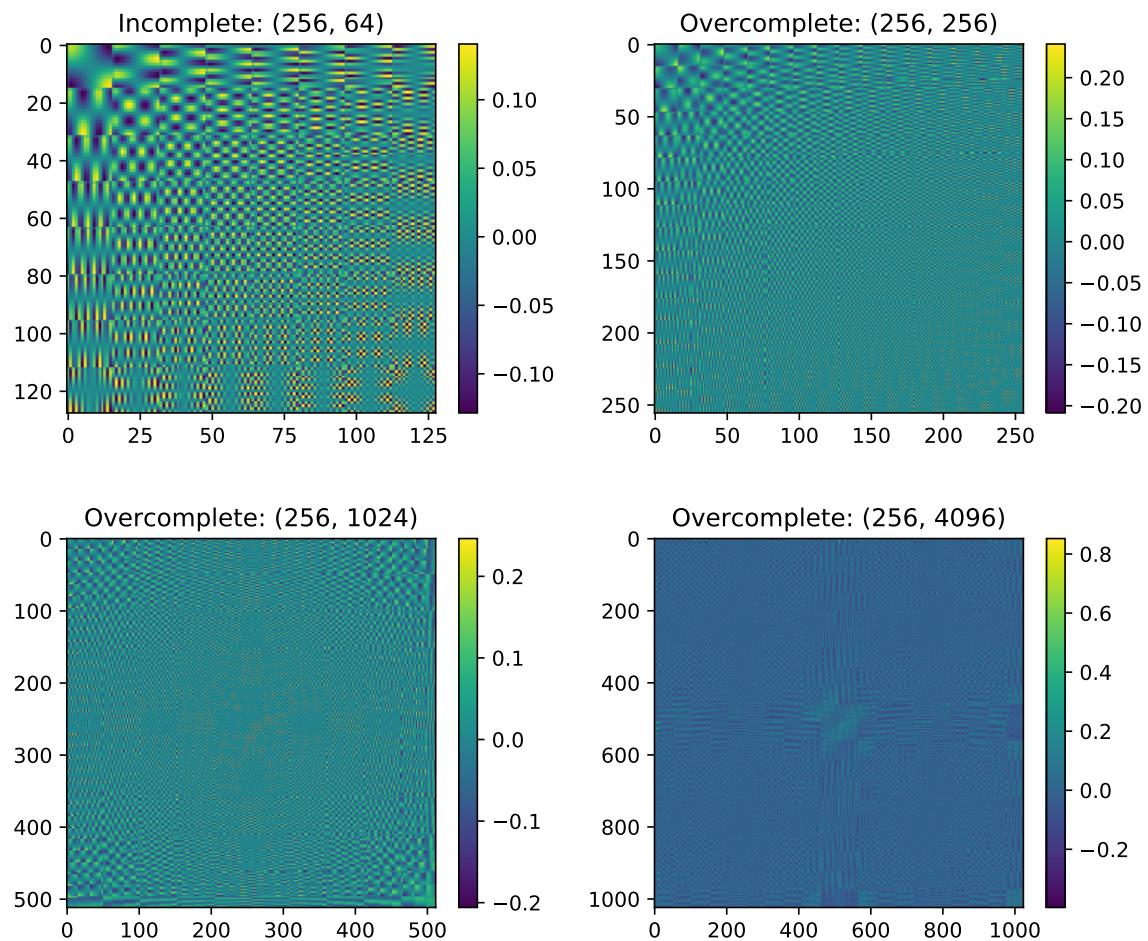


图 5.30: 2-dimension overcomplete DCT dictionary. These four dictionary are generated from 1-dimension DCT dictionary with size of 16×8 , 16×16 , 16×32 , 16×64 using formula $\text{式.}??$ respectively.

5.5.4 信号稀疏分解

5.5.4.1 简介

5.5.4.1.1 什么是信号稀疏分解

与频谱估计类似, 信号分解 (Signal Decomposition) 目的在于将信号分解成原子 (atom) 或基 (basis) 的组合 (线性或非线性), 信号稀疏分解 (Signal Sparse Decomposition) 目的在于将信号分解成少量原子 (atom) 基 (basis) 的组合 (线性或非线性)

假设有过完备字典 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, 其中每一列称为一个原子, 信号稀疏分解是指将信号 \mathbf{y} 分解为过完备字典 \mathbf{A} 中的尽可能少 (\mathbf{x} 稀疏) 的原子的线性组合

$$\mathbf{y} = \mathbf{Ax} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \dots + x_n\mathbf{a}_n, \quad (5.18)$$

其中, \mathbf{x} 仅有少量元素非零 (稀疏), 称为稀疏分解系数. 对于稀疏信号恢复问题, \mathbf{x} 被当作稀疏信号, \mathbf{y} 被当作观测信号, 字典相当于观测矩阵. 若考虑噪声成份, 则有

$$\mathbf{y} = \mathbf{Ax} + \mathbf{n},$$

其中, $\mathbf{n} \in \mathbb{R}^{m \times 1}$ 为噪声向量.

提示: 对于字典 $\mathbf{D} \in \mathbb{C}^{m \times n}$, 完备字典是指字典 \mathbf{D} 中的所有原子恰好能够张成 m 维空间, 当 $n >> m$ 且 \mathbf{D} 中的原子可以张成 m 维空间时, 称 \mathbf{D} 是过完备的.

上述分解中的过完备字典也可以是完备字典, 比如 DCT 逆变换对应的矩阵, 那么稀疏系数 \mathbf{x} 对应信号 \mathbf{y} 的频率成分.

5.5.4.1.2 常见稀疏分解算法

匹配追踪, 正交匹配追踪

5.5.4.2 匹配追踪

5.5.4.2.1 什么是匹配追踪

匹配追踪 (*Matching Pursuit*, MP) 最初被提出来是用于信号的稀疏分解 [9], 匹配追踪在很多领域中有应用, 匹配追踪用于量子模拟 [1].

5.5.4.2.2 匹配追踪算法

匹配追踪算法步骤见算法 1.

注解: 算法 1: 匹配追踪算法步骤

输入: 列归一化过完备字典矩阵 $\mathbf{A} \in \mathbb{R}^{m \times n}$, 待分解信号 $\mathbf{y} \in \mathbb{R}^{m \times 1}$, 稀疏度 K , 容忍残差模值 ϵ

输出: 稀疏分解系数 $\hat{\mathbf{x}} \in \mathbb{R}^{n \times 1}$, 索引集合 $\mathbb{I} \subset \mathbb{U} = \{1, 2, \dots, n\}$, 待分解信号近似 $\hat{\mathbf{y}}$ 和残差向量 $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$

Step1: 初始化稀疏信号 $\mathbf{x}_0 = \mathbf{0}$, 残差 $\mathbf{r}_0 = \mathbf{y} - \mathbf{A}\mathbf{x}_0 = \mathbf{y}$, 索引集合(支撑) $\mathbb{I}_0 = \emptyset$, 迭代计数器 $k = 1$.

Step2: 选择与残差最相关的原子, 即求解索引 λ_k 满足优化问题

$$i_k = \arg \max_{i \in \mathbb{U}} |\mathbf{A}_i^T \mathbf{r}_{k-1}|.$$

Step3: 更新索引集 $\mathbb{I}_k = \mathbb{I}_{k-1} \cup i_k$

Step4: 更新稀疏分解系数 $x[i_k] = \langle \mathbf{A}_{i_k}, \mathbf{r}_{k-1} \rangle = \mathbf{A}_{i_k}^T \mathbf{r}_{k-1}$

Step5: 计算新的残差 $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{A}_{i_k} x[i_k]$

Step6: 更新迭代计数器 $k = k + 1$ 若 $k < K$ 且 $\|\mathbf{r}\|_2 < \epsilon$, 重复 Step2 至 Step6, 否则停止迭代, 转 Step7.

Step7: 输出 $\hat{\mathbf{x}} = \mathbf{x}$, $\hat{\mathbf{y}} = \hat{\mathbf{y}}_k$, $\mathbf{r} = \mathbf{r}_k$. 其中, $\hat{\mathbf{x}}$ 在位置 \mathbb{I}_k 处非零.

5.5.4.2.3 实验与分析

5.5.4.3 正交匹配追踪

5.5.4.3.1 什么是正交匹配追踪

正交匹配追踪 (*Orthogonal Matching Pursuit*, OMP) 是一种信号稀疏分解方法, 由 Pati 等人 [10] 于 1993 年提出, OMP 被广泛用于稀疏信号恢复 [11]. 稀疏分解旨在将信号分解为过完备字典中的尽可能少(稀疏)的原子的线性组合, 从而获得信号的简洁有效的表示.

good (<https://blog.csdn.net/alec1987/article/details/79413055>)

5.5.4.3.2 正交匹配追踪算法

正交匹配追踪算法步骤见算法 1 [11].

注解: 算法 1: 正交匹配追踪算法步骤

输入: 列归一化过完备字典矩阵 $\mathbf{A} \in \mathbb{R}^{m \times n}$, 观测向量 $\mathbf{y} \in \mathbb{R}^{m \times 1}$, 待恢复信号稀疏度 K , 容忍残差模值 ϵ

输出: 待估计稀疏信号 $\hat{\mathbf{x}} \in \mathbb{R}^{n \times 1}$, 索引集合 $\mathbb{I} \subset \mathbb{U} = \{1, 2, \dots, n\}$, 观测向量近似 $\hat{\mathbf{y}}$ 和残差向量 $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$

Step1: 初始化稀疏信号 $\mathbf{x}_0 = \mathbf{0}$, 残差 $\mathbf{r}_0 = \mathbf{y} - \mathbf{A}\mathbf{x}_0 = \mathbf{y}$, 索引集合(支撑) $\mathbb{I}_0 = \emptyset$, 迭代计数器 $k = 1$.

Step2: 选择与残差最相关的原子, 即求解索引 i_k 满足优化问题

$$i_k = \arg \max_{i \in \mathbb{U}} |\mathbf{A}_i^T \mathbf{r}_{k-1}|.$$

Step3: 更新索引集 $\mathbb{I}_k = \mathbb{I}_{k-1} \cup i_k$.

Step4: 求解新的最小二乘优化问题

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}_{\mathbb{I}_k} \mathbf{x}\|_2^2$$

求得的稀疏系数解为 $\mathbf{x}_k = (\mathbf{A}_{\mathbb{I}_k}^T \mathbf{A}_{\mathbb{I}_k})^{-1} \mathbf{A}_{\mathbb{I}_k}^T \mathbf{y}$

Step5: 计算新的投影近似, 残差

$$\begin{aligned}\hat{\mathbf{y}}_k &= \mathbf{A}_{\mathbb{I}_k} \mathbf{x}_k \\ \mathbf{r}_k &= \mathbf{y} - \hat{\mathbf{y}}_k\end{aligned}$$

Step6: 更新迭代计数器 $k = k + 1$ 若 $k < K$ 且 $\|\mathbf{r}\|_2 < \epsilon$, 重复 Step2 至 Step5, 否则停止迭代, 转 Step7.

Step7: 输出 $\hat{\mathbf{x}} = \mathbf{x}_k$, $\hat{\mathbf{y}} = \hat{\mathbf{y}}_k$, $\mathbf{r} = \mathbf{r}_k$. 其中, $\hat{\mathbf{x}}$ 在位置 \mathbb{I}_k 处非零.

警告: 待查证是否有相关文献, 没有的话, 是否可以发表实际迭代过程中, 在求解逆矩阵 $(\mathbf{A}_{\mathbb{I}_k}^T \mathbf{A}_{\mathbb{I}_k})^{-1}$ 时容易出现其奇异的情况, 可以通过求解 $(\mathbf{A}_{\mathbb{I}_k}^T \mathbf{A}_{\mathbb{I}_k} + \alpha \mathbf{I})^{-1}$ 来代替, 其中 $\alpha > 0$.

提示: 记 $\mathbf{P}_k = \mathbf{A}_{\mathbb{I}_k} (\mathbf{A}_{\mathbb{I}_k}^T \mathbf{A}_{\mathbb{I}_k})^{-1} \mathbf{A}_{\mathbb{I}_k}^T$, 易知 \mathbf{P}_k 为正交投影矩阵(线性投影), 将向量投影到由 $\mathbf{A}_{\mathbb{I}_k}$ 的元素(类比坐标系)张成的线性空间中, 从而 $\hat{\mathbf{y}}_k = \mathbf{A}_{\mathbb{I}_k} \mathbf{x}_k = \mathbf{P}_k \mathbf{y}$, 则投影后的误差可以表示为

$$\begin{aligned}\mathbf{r}_k &= \mathbf{y} - \hat{\mathbf{y}}_k \\ &= (\mathbf{I} - \mathbf{P}_k) \mathbf{y} \\ &= (\mathbf{I} - \mathbf{P}_k) \mathbf{A} \mathbf{x} + (\mathbf{I} - \mathbf{P}_k) \mathbf{n},\end{aligned}$$

其中, $(\mathbf{I} - \mathbf{P}_k) \mathbf{A} \mathbf{x}$ 为残差中的信号成份, $(\mathbf{I} - \mathbf{P}_k) \mathbf{n}$ 为残差中的噪声成份.

5.5.4.3.3 实验与分析

主要分析 OMP 算法在信号稀疏分解上的性能, 有关 OMP 在压缩感知中的应用参见稀疏信号实验(页 281)小节.

5.5.4.3.3.1 仿真数据实验

实验中通过仿真生成含有三个频率成分的信号 $y = \sin 2f_1 t + \sin 2f_2 t + \sin 2f_3 t$, 仿真参数设置如下:

- 频率成分: $f_1 = 10\text{Hz}, f_2 = 20\text{Hz}, f_3 = 70\text{Hz}$
- 采样频率: $F_s = 256\text{Hz}$
- 采样时间: $T_s = 1\text{s}$
- 采样点数: $N_s = F_s T_s = 256$

构造过完备 DCT 字典 (参见 SubSection_ODCTDICT), 字典大小为 $M \times N$, 其中 $M = N_s, N = R * M, R$ 为一可调变量, 通过 OMP 求解稀疏系数 x , 并利用式.5.18 恢复信号 y .

实现代码, 参见文件 [demo_omp_sin3_decomposition.py](https://github.com/antsfamily/pysparse/tree/master/examples/decomposition) (<https://github.com/antsfamily/pysparse/tree/master/examples/decomposition>)

仿真生成的信号 y 如下图 (左) 所示, 其 DCT 变换后的系数如下图 (右) 所示

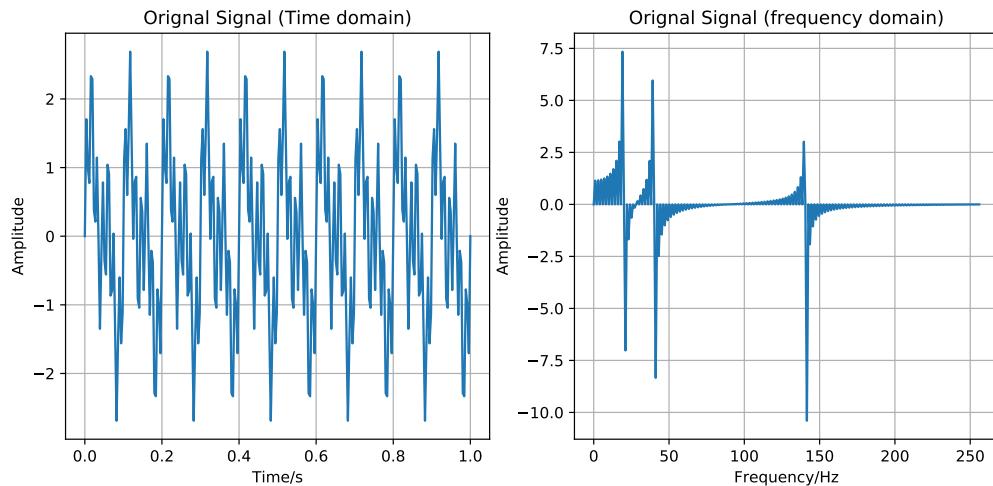


图 5.31: Signal in Time (left, y) and Frequency (right, x) domains.

如图中所示, 三种频率成分对应六个峰, 峰的位置为频率的二倍 (这是因为 DCT 变换中分母为 $2N$, 而不是 N), 峰的上下两个方向对应正频和负频.

$R = 2$ 即 $N = 2M$ 时, 设置不同稀疏度下的信号重构均方误差结果如下:

```
--MSE(y, y1) with k = 2: 0.937048086151075
--MSE(y, y2) with k = 4: 0.5420001800721357
--MSE(y, y3) with k = 6: 0.25283463530586237
--MSE(y, y4) with k = 100: 0.002307660406098402
```

求解的频域稀疏系数为

恢复的信号为

$R = 1$ 即 $N = M$ 时, 设置不同稀疏度下的信号重构均方误差结果如下:

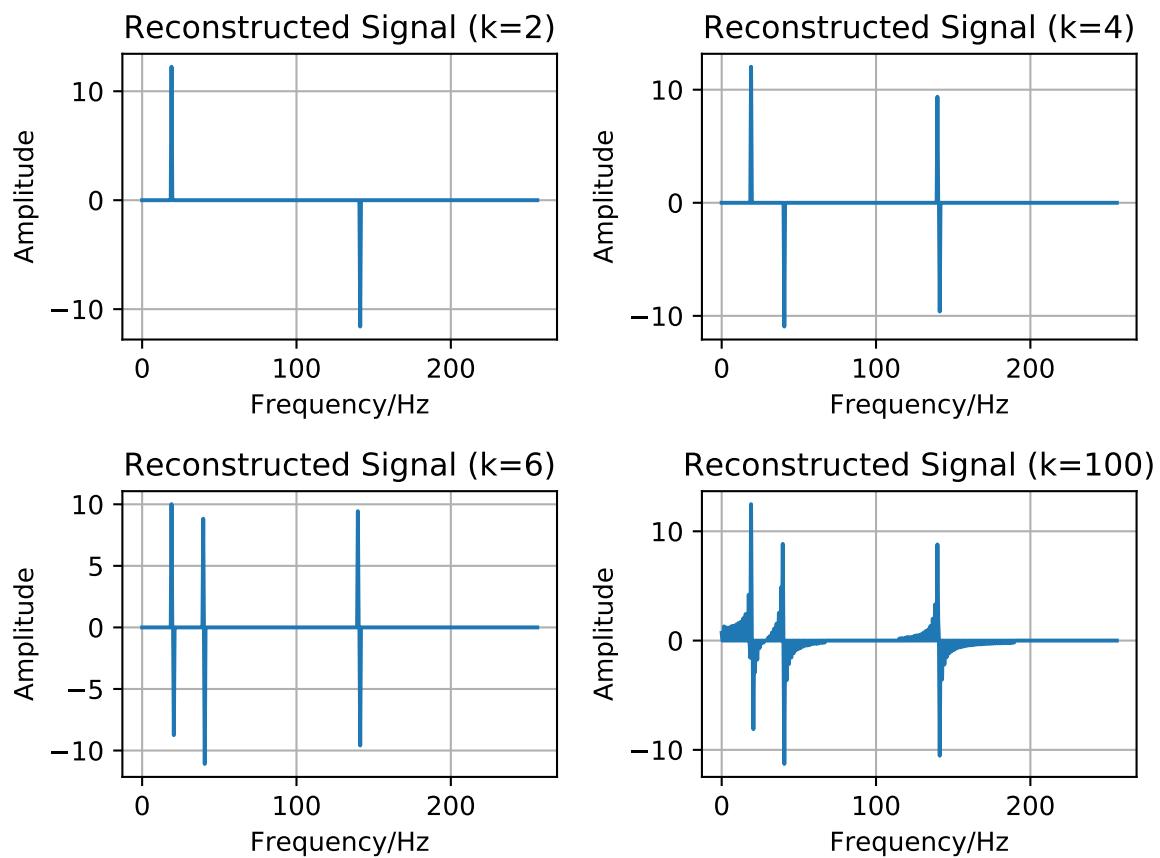


图 5.32: Recovered signal in frequency domain with different sparse degree k .

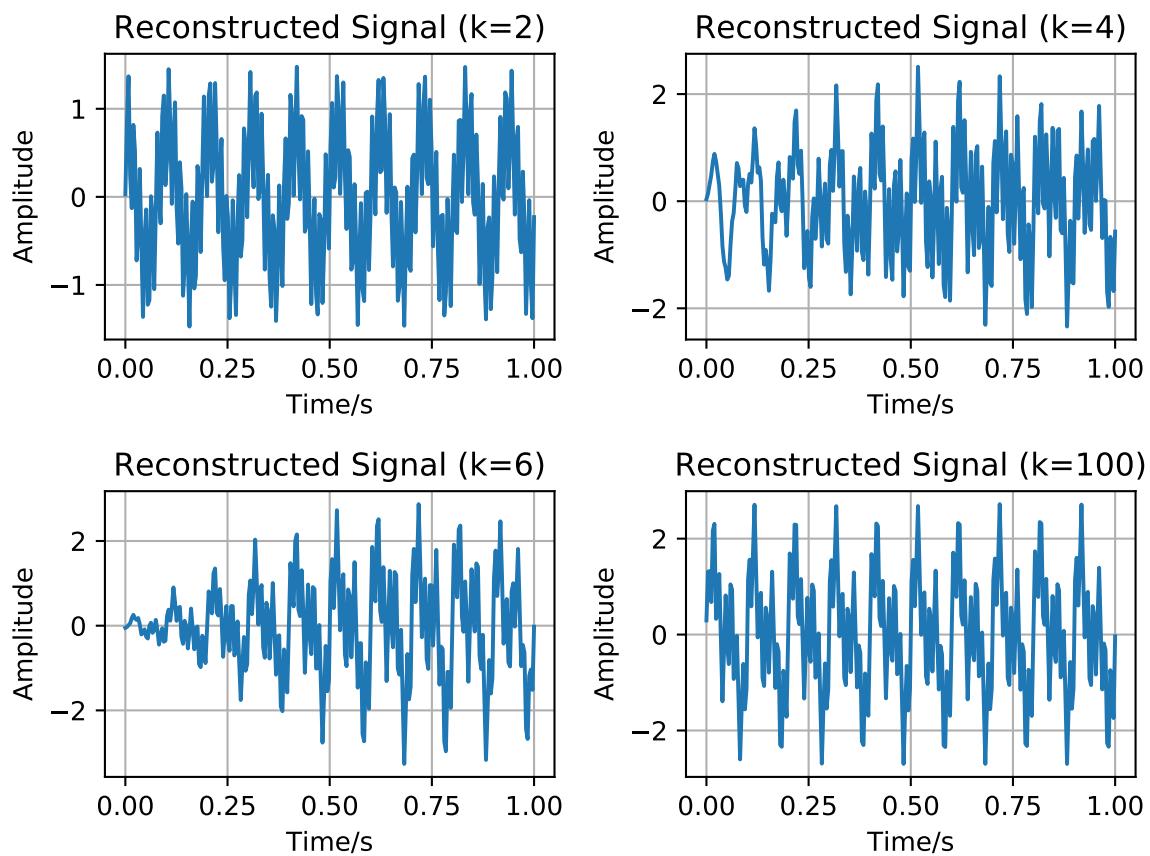


图 5.33: Recovered signal in time domain with different sparse degree k .

```
---MSE(y, y1) with k = 2: 0.9865788586325661
---MSE(y, y2) with k = 4: 0.5829159447924545
---MSE(y, y3) with k = 6: 0.3169739132218109
---MSE(y, y4) with k = 100: 0.0023321884115169826
```

求解的频域稀疏系数为

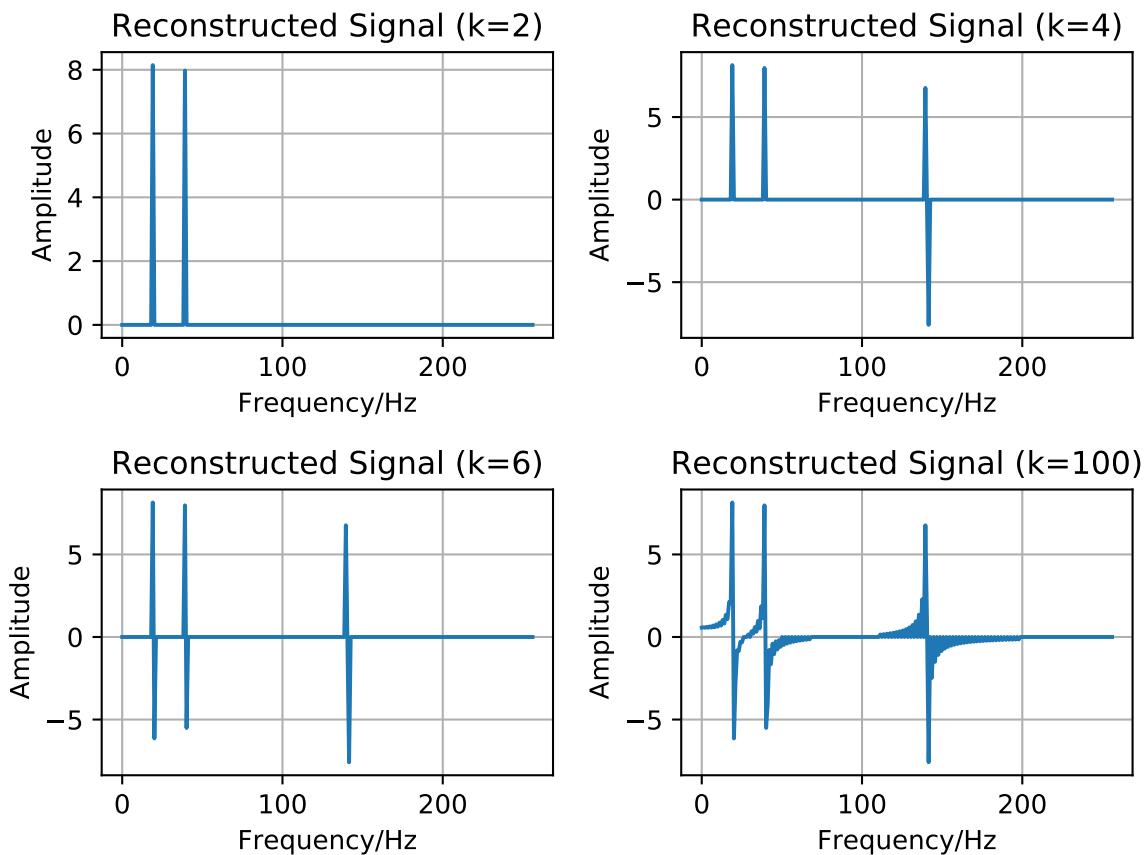


图 5.34: Recovered signal in frequency domain with different sparse degree k .

恢复的信号为

$R = 0.5$ 即 $N = M/2$ 时, 设置不同稀疏度下的信号重构均方误差结果如下:

```
---MSE(y, y1) with k = 2: 1.031848207375802
---MSE(y, y2) with k = 4: 0.6615399428007099
---MSE(y, y3) with k = 6: 0.4982313612363407
---MSE(y, y4) with k = 100: 0.32437796224425747
```

求解的频域稀疏系数为

恢复的信号为

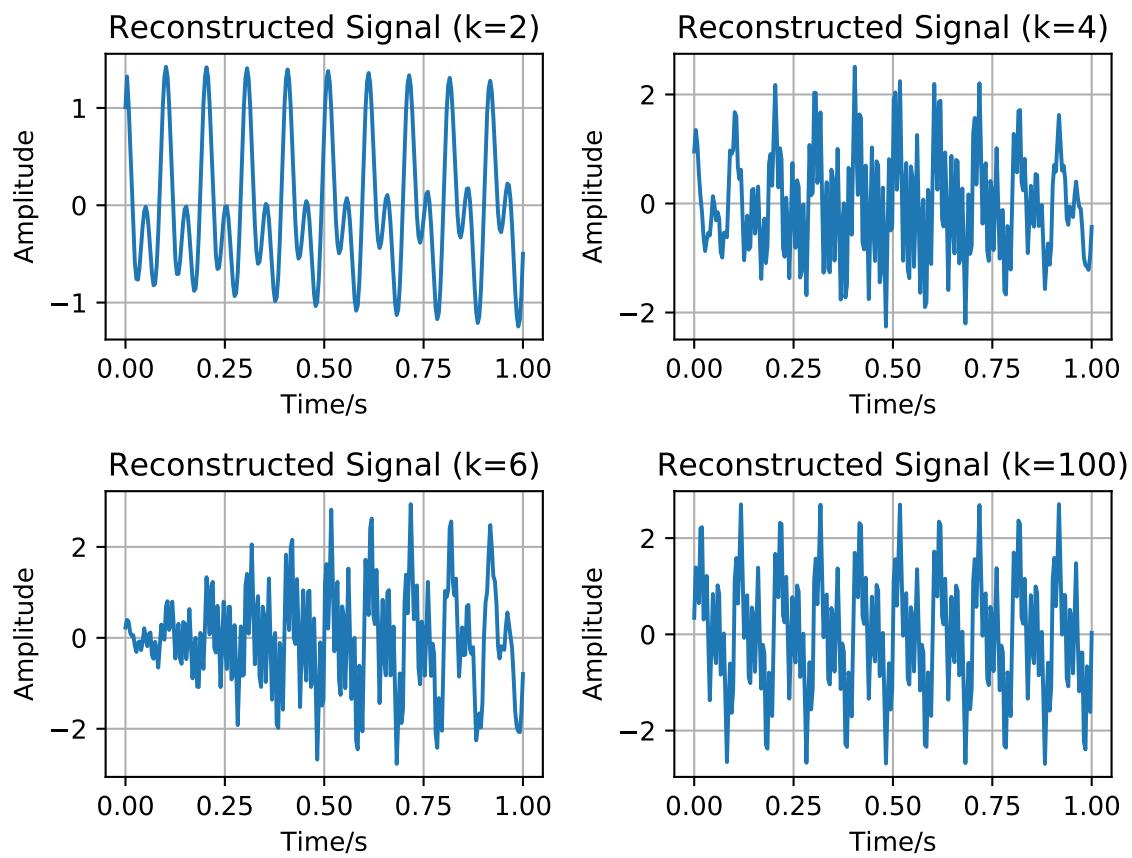


图 5.35: Recovered signal in time domain with different sparse degree k .

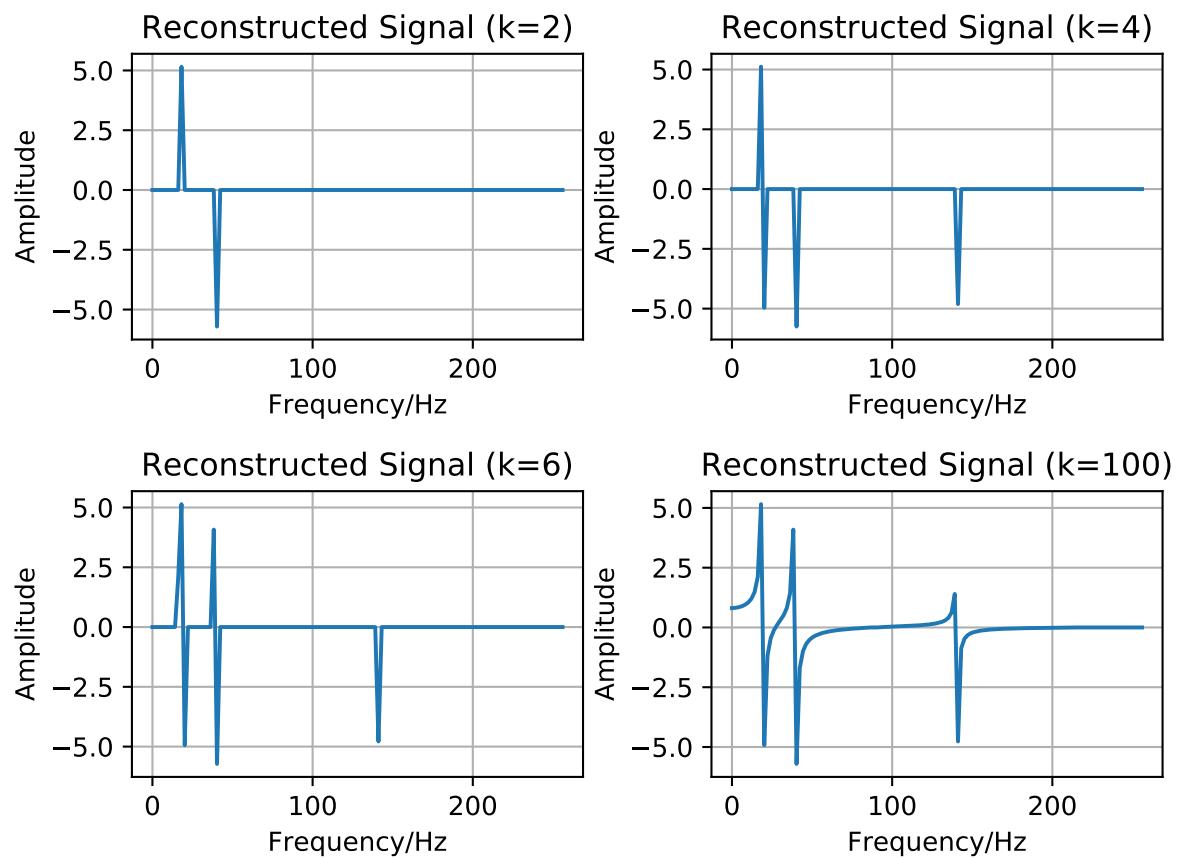


图 5.36: Recovered signal in frequency domain with different sparse degree k .

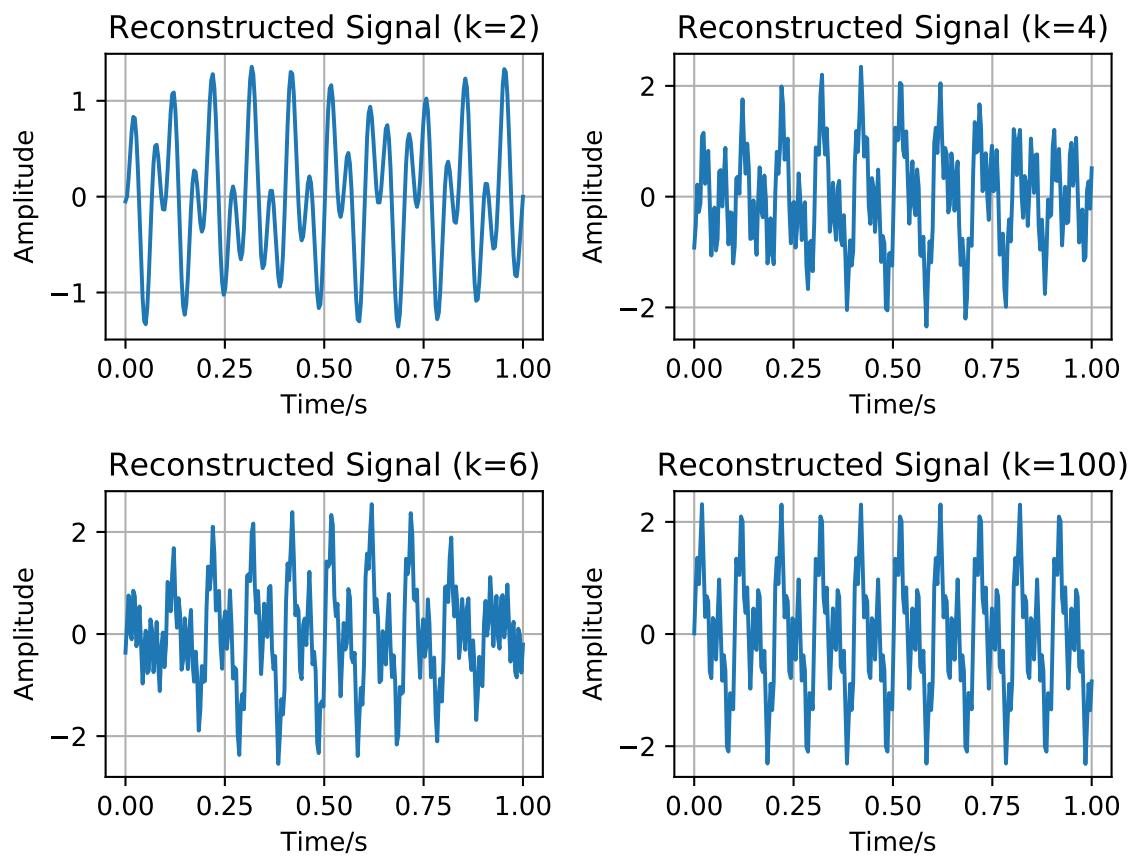


图 5.37: Recovered signal in time domain with different sparse degree k .

5.5.4.3.3.2 真实数据实验

5.5 线性压缩感知

5.5.1 简介

- 线性压缩感知 (*Linear Compressive Sensing*, LCS)
- 非线性压缩感知 (*Nonlinear Compressive Sensing*, NCS)
- 可学习压缩感知 (*Learned Compressive Sensing*, LCS)

警告: 需要注意的是, 在压缩感知中, 当信号不稀疏时, 假设信号在某一字典基 D 下可被稀疏表示, 组成新的观测矩阵 $\Phi = \mathbf{AD}$, 通过恢复稀疏系数进而恢复非稀疏信号. 这里并不要求在压缩观测时需要先进行稀疏表示 (假设某种硬件可以完成稀疏表示), 再进行压缩观测, 而是仍然以 \mathbf{A} 为观测矩阵, 只是在稀疏求解时引入稀疏表示矩阵. 如果在观测时, 先对稀疏表示, 再进行观测, 并且在恢复时使用 Φ 作为观测矩阵, 效果并不如前者好. 这些可以通过实验验证.

- [Compressive Sensing Resources](https://arquivo.pt/wayback/20160516193220/http://dsp.rice.edu/cs) (<https://arquivo.pt/wayback/20160516193220/http://dsp.rice.edu/cs>) : resources and softwares

5.5.2 线性压缩感知

压缩感知 (*Compressive Sensing*, CS) 由 [Donoho] 等人于 2004 年提出, 压缩感知线性模型, 即线性压缩感知 (*Linear Compressive Sensing*, LCS)

[压缩感知笔记](https://www.cnblogs.com/AndyJee/category/579870.html) (<https://www.cnblogs.com/AndyJee/category/579870.html>)

5.5.2.1 线性压缩感知概述

为便于描述, 将一个压缩感知系统记为 $\text{CS}(\mathfrak{G}; \mathfrak{F})$, 其中, \mathfrak{G} 为观测系统, \mathfrak{F} 为重构系统. 例如, 观测系统为 Φ, \cdot , 重构系统为 \mathbf{A}, omp .

压缩感知旨在设计一个观测系统 \mathfrak{G} 和一个恢复系统 \mathfrak{F} , 使得信号 \mathbf{x} 可以以欠奈奎斯特采样频率采样并得到较好的恢复.

其观测过程可以表示为

$$\mathbf{y} = \mathfrak{G}(\mathbf{x}) \quad (5.19)$$

恢复过程可以表示为

$$\mathbf{x} = \mathfrak{F}(\mathbf{y}). \quad (5.20)$$

对于线性压缩感知, 有 $\mathfrak{G}(\mathbf{x}) = \Phi \mathbf{x}$.

- Φ : 测量矩阵/观测矩阵
- Ψ : 稀疏基矩阵/稀疏表示字典矩阵

- $\mathbf{A} = \Phi\Psi$: 传感矩阵/感知矩阵

压缩感知理论指出, 当测量矩阵 Φ 满足有限等距性(页 251)性质时, 信号能够以较大概率被恢复.

5.5.5.2.2 一维线性压缩感知

5.5.5.2.2.1 稀疏信号的压缩采样与恢复

设有原始稀疏信号 $\mathbf{x} \in \mathbb{R}^{N \times 1}$, 测量矩阵 $\Phi \in \mathbb{R}^{M \times N}$, 则测量过程可以表示为

$$\mathbf{y} = \Phi\mathbf{x} + \mathbf{n} \quad (5.21)$$

其中, $\mathbf{y} \in \mathbb{R}^{M \times 1}$ 为测量向量, $\mathbf{n} \in \mathbb{R}^{M \times 1}$, $M < N$ 为观测系统噪声. 在已知 \mathbf{y} 和 Φ 时, 原始信号 \mathbf{x} 的恢复可以通过优化

$$\min_{\mathbf{x}} \|\mathbf{x}\|_p, \quad s.t. \quad \mathbf{y} = \Phi\mathbf{x} + \mathbf{n}, \quad (5.22)$$

其中, $|\cdot|_p$, ($0 \leq p \leq 1$) 表示 ℓ_p 范数. 式 5.22 可以通过正交匹配追踪(页 262)的方法求解. 与式 5.22 等效的优化问题为

$$\min_{\mathbf{x}} = \|\mathbf{x}\|_p + \lambda \|\mathbf{y} - \Phi\mathbf{x}\|_2, \quad (5.23)$$

其中, λ 是平衡因子, $|\cdot|_p$, ($0 \leq p \leq 1$) 表示 ℓ_p 范数.

5.5.5.2.2.2 非稀疏信号的压缩采样与恢复

设有原始非稀疏信号 \mathbf{x} , 设其可通过可逆线性变换 Ψ^{-1} 变换到稀疏信号 \mathbf{z} ($\mathbf{z} = \Psi^{-1}\mathbf{x}$), 即可在字典 Ψ 下进行稀疏表示 ($\mathbf{x} = \Psi\mathbf{z}$), 则对于非稀疏信号的压缩采样与恢复有两种方式, 下面进行介绍.

5.5.5.2.2.3 第一种方式

1. 变换: $\mathbf{z} = \Psi^{-1}\mathbf{x}$
2. 观测: $\mathbf{y} = \Phi\mathbf{z}$,
3. 求解: $\min_{\mathbf{z}} \|\mathbf{z}\|_p \quad s.t. \quad \mathbf{y} = \Phi\mathbf{z}$
4. 重构: $\mathbf{x} = \Psi\mathbf{z}$

5.5.5.2.2.4 第二种方式

1. 观测: $\mathbf{y} = \Phi\mathbf{x}$
2. 求解: $\min_{\mathbf{z}} \|\mathbf{z}\|_p \quad s.t. \quad \mathbf{y} = \Phi\Psi\mathbf{z}$
3. 重构: $\mathbf{x} = \Psi\mathbf{z}$

5.5.5.2.2.5 二维压缩感知

5.5.5.2.3 复压缩感知

对于复数域压缩感知问题可以转换成实数域压缩感知处理. 若 $\mathbf{y} \in \mathbb{C}^{M \times 1}$, $\Phi \in \mathbb{C}^{M \times N}$, $\mathbf{x} \in \mathbb{C}^{N \times 1}$, 问题可以转化为

$$\operatorname{Re}(\mathbf{y}) + j\operatorname{Im}(\mathbf{y}) = \operatorname{Re}(\Phi\mathbf{x}) + j\operatorname{Im}(\Phi\mathbf{x}) \quad (5.24)$$

写成矩阵形式为

$$\begin{bmatrix} \operatorname{Re}(\mathbf{y}) \\ \operatorname{Im}(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \operatorname{Re}(\Phi) & -\operatorname{Im}(\Phi) \\ \operatorname{Im}(\Phi) & \operatorname{Re}(\Phi) \end{bmatrix} \begin{bmatrix} \operatorname{Re}(\mathbf{x}) \\ \operatorname{Im}(\mathbf{x}) \end{bmatrix}$$

5.5.5.2.4 压缩感知与流形

流形与压缩感知中的一些重要结果有着密切的关系, 如一组满足 $\|\mathbf{x}\|_0 = k$ 的信号构成了一个 k 维黎曼流形. 又如压缩感知中的许多随机感知矩阵可看作是低维流形中的保留结构 [3][4].

参见 [5] Chapter 1

5.5.5.2.5 压缩感知分析实验

设计 Gaussian 观测矩阵 Φ , DCT 稀疏表示字典 Ψ , 记感知矩阵 $\mathbf{A} = \Phi\Psi$. 为便于描述, 记一个压缩感知系统为 $\text{CS}(\mathfrak{G}; \mathfrak{F})$, 其中, \mathfrak{G} 为观测系统, \mathfrak{F} 为重构系统. 以完成对二维图像的高度维压缩采样与恢复为例, 实验中分析: a) $\text{CS}(\Phi; \Phi)$, b) $\text{CS}(\Phi; \mathbf{A})$, c) $\text{CS}(\mathbf{A}; \Phi)$, d) $\text{CS}(\mathbf{A}; \mathbf{A})$ 四种压缩感知系统的性能.

Python 实现参见文件 [demo_csAnalysis.py](https://github.com/antsfamily/pysparse/tree/master/examples/cs/demo_csAnalysis.py) (https://github.com/antsfamily/pysparse/tree/master/examples/cs/demo_csAnalysis.py)

以 cameraman 图像为例, 可可视化该图像, 观测矩阵, DCT 字典和稀疏表示系数

压缩比设置为 4, 采用 OMP 算法重构图像信号, 稀疏度设置为 $k = 32$ 时的结果为

压缩比设置为 4, 采用 OMP 算法重构图像信号, 稀疏度设置为 $k = 64$ 时的结果为

压缩比设置为 4, 采用 OMP 算法重构图像信号, 稀疏度设置为 $k = 128$ 时的结果为

注解: 可见, 针对非稀疏信号, 正确的感知系统有两种

1. 在观测时先将信号变换为稀疏信号, 再对稀疏信号进行压缩观测, 通过测量矩阵和观测值求解稀疏信号, 再进行逆变换重构原始信号.
 2. 采用测量矩阵直接测量非稀疏信号, 在求解时, 假设信号在某一字典下可以进行稀疏表示, 通过字典和测量矩阵以及观测值求解稀疏信号, 再进行逆变换重构原始信号
-

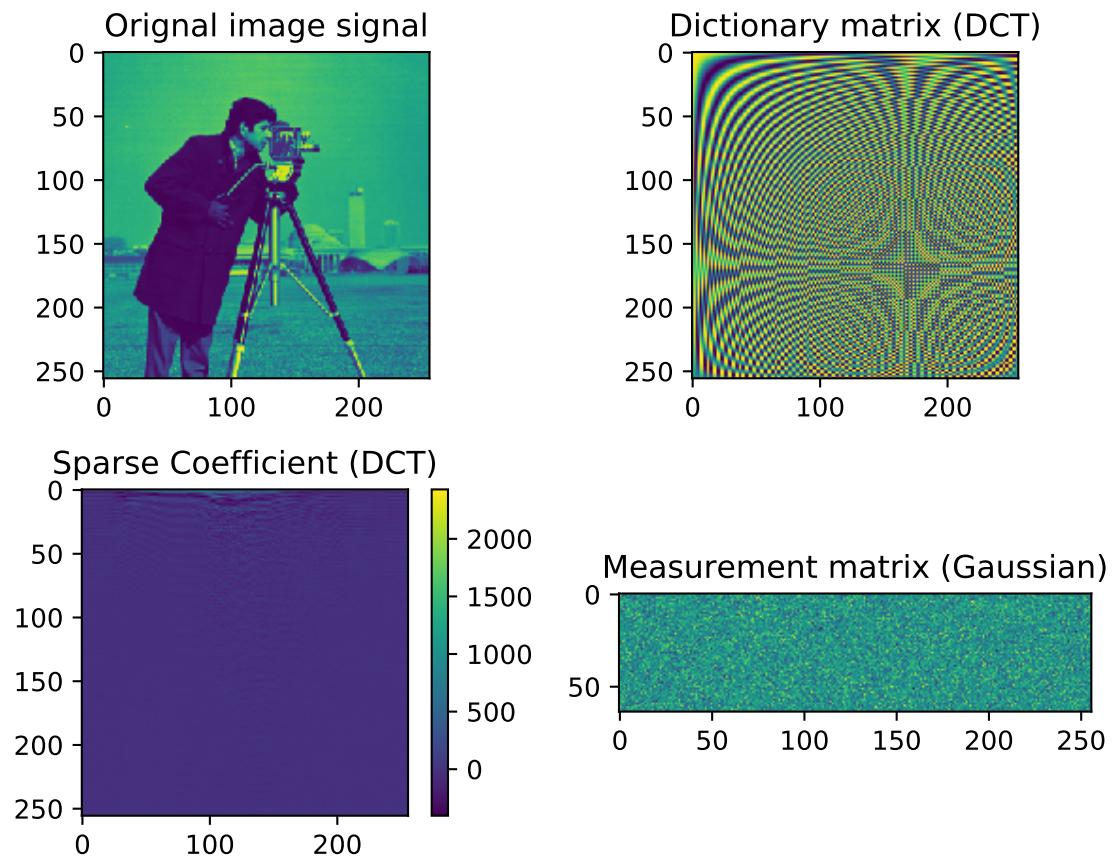
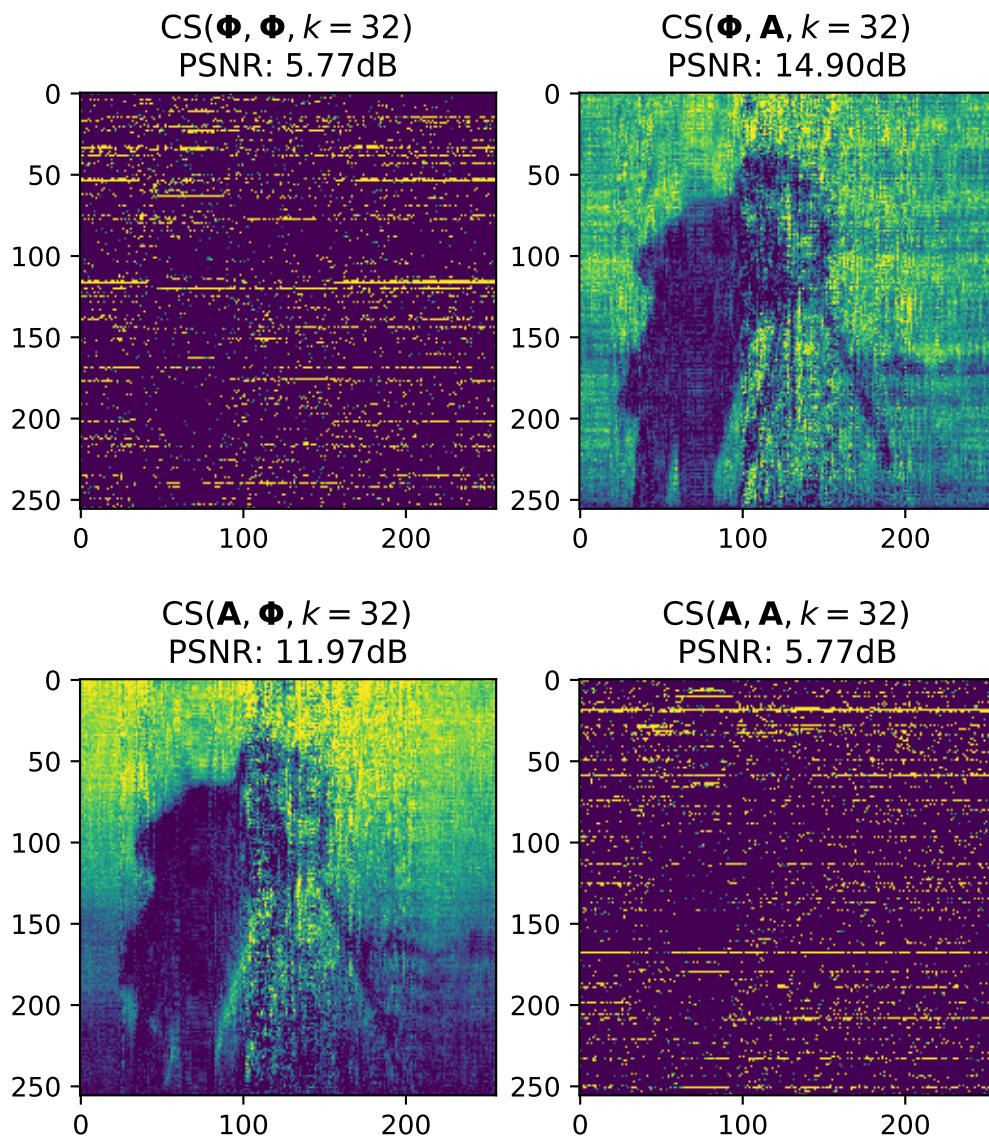
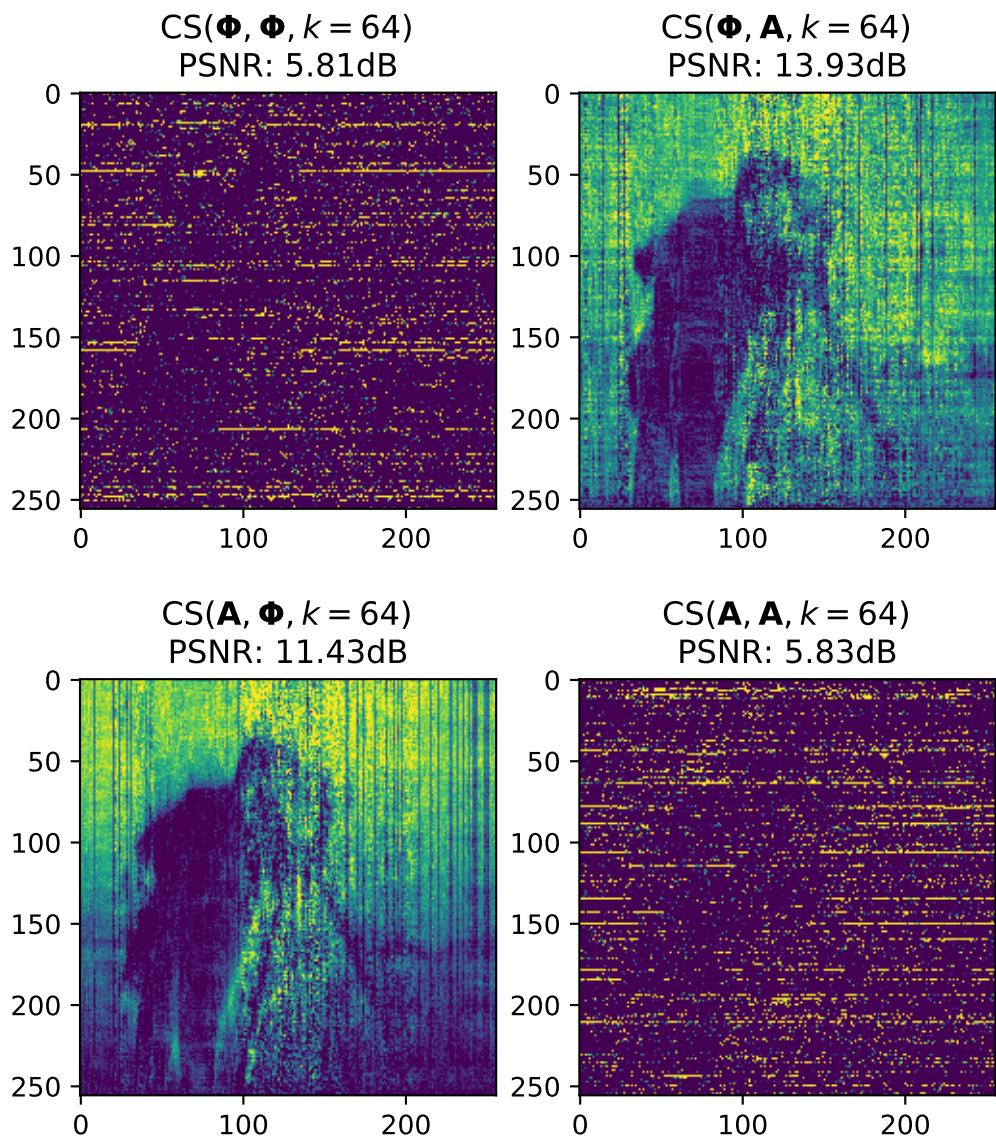
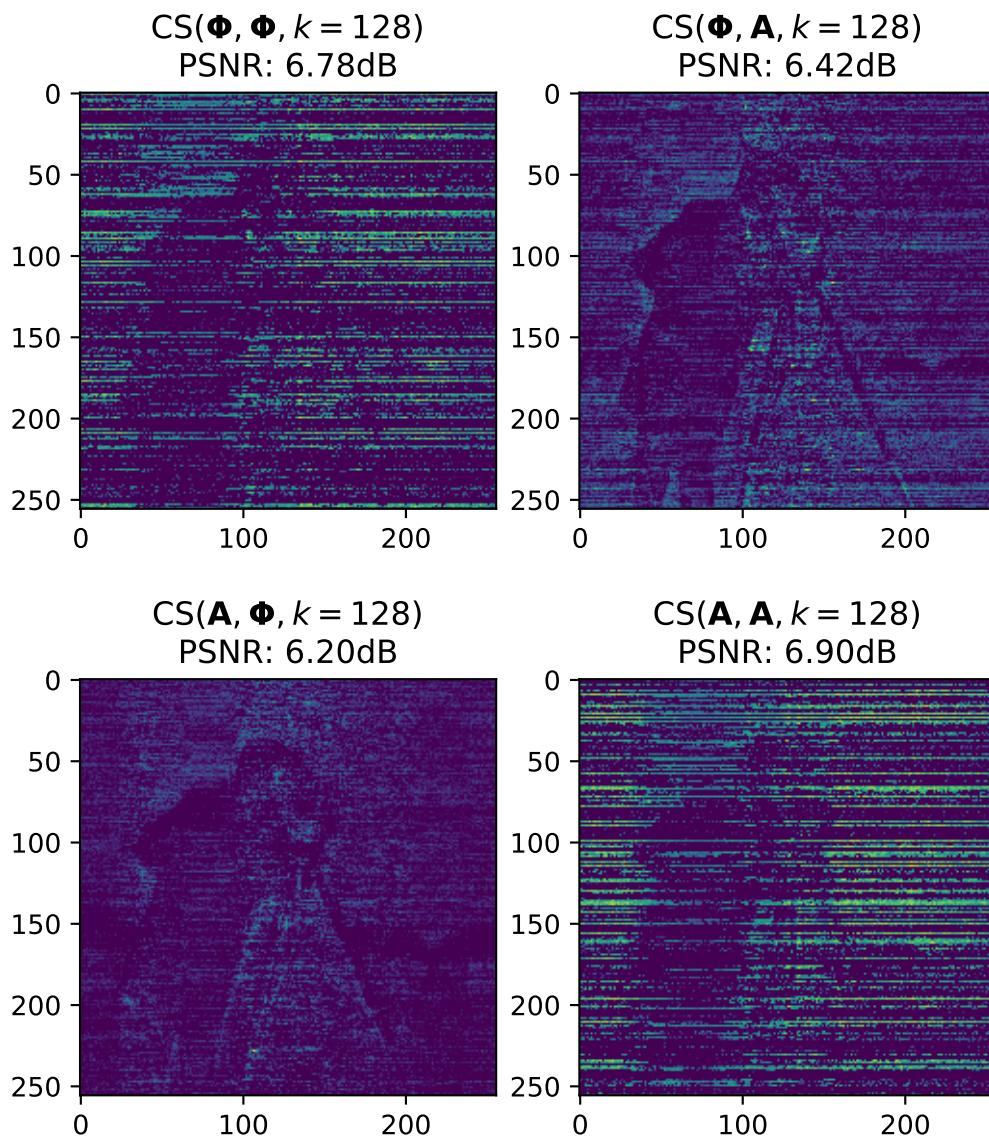


图 5.38: Original image signal, dictionary matrix, observation matrix, sparse coefficient matrix.







5.5.5.3 线性压缩感知信号模型

5.5.5.4 线性压缩感知观测

5.5.5.4.1 感知矩阵的设计

感知矩阵的设计需要满足如下条件.

5.5.5.4.1.1 零空间条件

设有信号空间 \mathbb{X}^n 中的任意不同信号 $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}^n$, 感知矩阵 $\mathbf{A} \in \mathbb{R}^{m \times n}$, 对应的观测向量分别为 $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Y}^m$. 若想从 $\mathbf{y}_1, \mathbf{y}_2$ 中恢复 $\mathbf{x}_1, \mathbf{x}_2$, 需保证不同信号在投影后依然不同 $\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1 \neq \mathbf{y}_2 = \mathbf{A}\mathbf{x}_2$, 即 $\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) \neq 0$, 亦即 $\mathbf{x}_1 - \mathbf{x}_2 \notin \mathcal{N}(\mathbf{A})$. 这一性质可以通过矩阵的 Spark, 零空间性质等表达.

Theorem 53 对于 $\forall \mathbf{y} \in \mathbb{Y}^m$, 当且仅当 $\text{Spark}(\mathbf{A}) > 2k$ 时, 最多存在一个 k 稀疏信号 $\mathbf{x} \in \mathbb{X}_k^n$, 使得 $\mathbf{y} = \mathbf{A}\mathbf{x}$ 成立.

该定理的证明参见文献 [5].

下面介绍矩阵的零空间性质 (Null Space Property, NSP) [5].

Definition 54 (零空间性质 (Null Space Property, NSP)) 设矩阵 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, $\mathbf{a}_i \in \mathbb{R}^m, i = 1, 2, \dots, n$. 若 $\forall \mathbf{v} \in \mathcal{N}(\mathbf{A})$, 满足

$$\|\mathbf{v}_{\mathbb{K}}\|_1 < \|\mathbf{v}_{\bar{\mathbb{K}}}\|_1 \quad (5.25)$$

其中, $\mathbb{K} \subset \mathbb{N}, \mathbb{N} = \{1, 2, \dots, n\}$, $\bar{\mathbb{K}}$ 为 \mathbb{K} 在 \mathbb{N} 中的补集, $\text{card}(\mathbb{K}) \leq k$. 则称矩阵 \mathbf{A} 满足 k 阶 零空间性质 (Null Space Property, NSP).

下面介绍矩阵的零空间性质 (Null Space Property, NSP) 的另一种定义 [6].

Definition 55 (零空间性质 (Null Space Property, NSP)) 设矩阵 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, $\mathbf{a}_i \in \mathbb{R}^m, i = 1, 2, \dots, n$. 若 $\forall \mathbf{v} \in \mathcal{N}(\mathbf{A}), \exists C > 0$ 满足

$$\|\mathbf{v}_{\mathbb{K}}\|_2 \leq C \frac{\|\mathbf{v}_{\bar{\mathbb{K}}}\|_1}{\sqrt{k}} \quad (5.26)$$

其中, $\mathbb{K} \subset \mathbb{N}, \mathbb{N} = \{1, 2, \dots, n\}$, $\bar{\mathbb{K}}$ 为 \mathbb{K} 在 \mathbb{N} 中的补集, $\text{card}(\mathbb{K}) \leq k$. 则称矩阵 \mathbf{A} 满足 k 阶 零空间性质 (Null Space Property, NSP).

5.5.5.4.1.2 有限等距性质

RIP 被广泛用于压缩感知 (Compressive Sensing, CS) 中 [7]. 在 CS 中, 感知矩阵 ($\mathbf{A} = \Phi\Psi$) 是否满足 RIP 条件直接决定了重构信号质量. 在 CS 中, 矩阵 \mathbf{A} 的有限等距常数定义为

Definition 56 (有限等距常数 (Restricted Isometry Constant)) 设 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, $\mathbf{a}_i \in \mathbb{R}^m, i = 1, 2, \dots, n$. 对于每个整数 $k \in [1, n]$, 定义矩阵 \mathbf{A} 的 k 阶有限等距常数为满足下式的最小的数 δ_k

$$(1 - \delta_k)\|\mathbf{x}\| \leq \|\mathbf{A}\mathbf{x}\| \leq (1 + \delta_k)\|\mathbf{x}\| \quad (5.27)$$

其中, $\mathbf{x} \in \mathbb{R}_k^n$ 为 k 稀疏信号.

更一般地, 设 $0 < \beta < \gamma < +\infty$,

$$\beta\|\mathbf{x}\| \leq \|\mathbf{Ax}\| \leq \gamma\|\mathbf{x}\| \quad (5.28)$$

其中, $\mathbf{x} \in \mathbb{R}_k^n$ 为 k 稀疏信号.

矩阵 \mathbf{A} 具有 **有限等距特性** (*Restricted Isometry Property*, RIP) [6] 是指对于足够大的 k 和充分小的 δ_k , 矩阵 \mathbf{A} 满足式.5.27. 式.5.27 表明, 矩阵 \mathbf{A} 对信号 \mathbf{x} 投影前后的长度仅有微小的改变. 即 RIP 特性保证了来自矩阵 \mathbf{A} 的投影可以保持两个信号 $\mathbf{x}_1, \mathbf{x}_2$ 间的距离, 式.5.27 可以等价地表示为

$$(1 - \delta_k)\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\| \leq (1 + \delta_k)\|\mathbf{x}_1 - \mathbf{x}_2\|. \quad (5.29)$$

一个矩阵 \mathbf{A} 以很高概率满足 RIP, 也就保证了稀疏信号 \mathbf{x} 可以被以高概率重构.

提示: 对于式.5.28 中的矩阵 \mathbf{A} , 用 $\sqrt{2}/(\beta + \gamma)$ 乘以 \mathbf{A} , 则有 $\sqrt{2}/(\beta + \gamma)\mathbf{A}$ 满足式.5.27, 其中, $\delta_k = (\beta - \gamma)/(\beta + \gamma)$. 更多内容参见**有限等距性** (页 251) 小节.

5.5.5.4.1.3 相干性/一致性条件

Definition 57 (相干性/一致性 (Coherence)) 设 $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m \times n}$, $\mathbf{a}_i \in \mathbb{R}^m$, $i = 1, 2, \dots, n$. 对于每个整数 $k \in [1, n]$, 定义矩阵 \mathbf{A} 的一致性为

$$\mu(\mathbf{A}) = \max_{1 \leq i \neq j \leq N} \left| \left\langle \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|_2}, \frac{\mathbf{a}_j}{\|\mathbf{a}_j\|_2} \right\rangle \right| = \max_{1 \leq i \neq j \leq N} \frac{|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|}{\|\mathbf{a}_i\|_2 \|\mathbf{a}_j\|_2} \quad (5.30)$$

Theorem 58 对于任意矩阵 \mathbf{A} 有

$$\text{Spark}(\mathbf{A}) \geq 1 + \frac{1}{\mu(\mathbf{A})} \quad (5.31)$$

Theorem 59 对于矩阵 \mathbf{A} , 若

$$k < \frac{1}{2} \left(1 + \frac{1}{\mu(\mathbf{A})} \right) \quad (5.32)$$

则对于 $\forall \mathbf{y} \in \mathbb{R}^m$, 最多存在一个 k 稀疏信号, 满足 $\mathbf{y} = \mathbf{Ax}$.

5.5.5.4.1.4 稳定性

Definition 60 (C 稳定系统) 设有感知系统 $\mathcal{F} = \mathbf{A} \in \mathbb{R}^{m \times n} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, 重构算法 $\mathcal{G} : \mathbb{R}^m \rightarrow \mathbb{R}^n$, $m \leq n$. 若对于任意 k 稀疏信号 $\mathbf{x} \in \mathbb{R}_k^n$ 和误差向量 $\mathbf{e} \in \mathbb{R}^m$, 有

$$\|\mathcal{G}(\mathcal{F}(\mathbf{x}) + \mathbf{e}) - \mathbf{x}\|_2 = \|\mathcal{G}(\mathbf{Ax} + \mathbf{e}) - \mathbf{x}\|_2 \leq C\|\mathbf{e}\|_2 \quad (5.33)$$

则称系统 $\{\mathcal{F} = \mathbf{A}, \mathcal{G}\}$ 是 C 稳定的.

该定义说明, 对于 C 稳定的系统, 若观测存在很小的误差或者对观测施加少量的噪声, 重建质量不会受到大的影响, RIP 保证了上述性质的成立.

5.5.5.4.1.5 讨论

- NSP 不容易证明, 且容易受噪声影响;
- RIP 比 NSP 更为严格和鲁棒¹.

5.5.5.4.2 感知矩阵的构造

5.5.5.4.3 常见感知矩阵

5.5.5 线性压缩感知重构

5.5.5.1 重构算法

5.5.5.1.1 OMP

注解: 算法 1: 正交匹配追踪算法步骤

输入: 观测矩阵 $\Phi \in \mathbb{R}^{m \times n}$, 观测向量 $\mathbf{y} \in \mathbb{R}^{m \times 1}$, 待恢复信号稀疏度 k

输出: 估计信号 $\hat{\mathbf{x}} \in \mathbb{R}^{n \times 1}$, 索引集合 $\mathbb{I}_k \subset \mathbb{U} = \{1, 2, \dots, n\}$, 观测向量近似 $\hat{\mathbf{y}}$ 和残差向量 $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$

Step1: 初始化残差 $\mathbf{r}_0 = \mathbf{y}$, 索引集合 $\mathbb{I}_0 = \emptyset$, 迭代计数器 $t = 1$.

Step2: 求解索引 λ_t 满足优化问题

$$i_t = \arg \max_{j \in \mathbb{U}} |\phi_j \mathbf{r}_{t-1}|.$$

Step3: 更新索引集 $\mathbb{I}_t = \mathbb{I}_{t-1} \cup i_t$.

Step4: 求解新的最小二乘优化问题

$$\mathbf{x}_t = \arg \max_{\mathbf{x}} \|\mathbf{y} - \Phi_{\mathbb{I}_t} \mathbf{x}\|_2^2$$

求得的解为 $\mathbf{x}_t = (\Phi_{\mathbb{I}_t}^T \Phi_{\mathbb{I}_t})^{-1} \Phi_{\mathbb{I}_t}^T \mathbf{y}$

Step5: 计算新的投影近似, 残差

$$\begin{aligned} \hat{\mathbf{y}}_t &= \Phi_{\mathbb{I}_t} \mathbf{x}_t \\ \mathbf{r}_t &= \mathbf{y} - \hat{\mathbf{y}}_t \end{aligned}$$

Step6: 更新迭代计数器 $t = t + 1$ 若 $t < k$, 重复 Step2 至 Step5, 否则停止迭代, 转 Step7.

Step7: 输出 $\hat{\mathbf{x}} = \mathbf{x}_t$, $\hat{\mathbf{y}} = \hat{\mathbf{y}}_t$, $\mathbf{r} = \mathbf{r}_t$. 其中, $\hat{\mathbf{x}}$ 在位置 \mathbb{I}_t 处非零.

¹ 参见文献 [5] p24.

警告: 待查证是否有相关文献, 没有的话, 是否可以发表实际迭代过程中, 在求解逆矩阵 $(\Phi_{\mathbb{I}_t}^T \Phi_{\mathbb{I}_t})^{-1}$ 时容易出现其奇异的情况, 可以通过求解 $(\Phi_{\mathbb{I}_t}^T \Phi_{\mathbb{I}_t} \alpha \mathbf{I})^{-1}$ 来代替, 其中 $\alpha > 0$.

提示: 记 $P_t = \Phi(\Phi_{\mathbb{I}_t}^T \Phi_{\mathbb{I}_t})^{-1} \Phi_{\mathbb{I}_t}^T$, 易知 P_t 为正交投影矩阵. 从而

$$\begin{aligned}\mathbf{x}_t &= P_t \mathbf{y} \\ \mathbf{r}_t &= \mathbf{y} - \mathbf{x}_t\end{aligned}$$

5.5.5.6 线性压缩感知实践

5.5.5.6.1 稀疏信号实验

5.5.5.6.1.1 一维仿真信号

- 信号 (时域): $\mathbf{x}_o = \sin(2\pi f_1 t) + \sin(2\pi f_2 t) + \sin(2\pi f_3 t)$
- 信号 (频域): $\mathbf{x} = \text{abs}(\text{DFT}(\mathbf{x}_o))$
- 观测矩阵: \mathbf{A} 高斯矩阵和过完备 DCT 矩阵

提示: 过完备 DCT 字典的构造参见 SubSection_ODCTDICT 小节.

设置时域信号 \mathbf{x}_o 中的三个频率分别为 $f_1 = 100Hz, f_2 = 200Hz, f_3 = 700Hz$; 设置采样率 $F_s = 2000Hz$, 采样时间 $T_s = 1s$; 则时域信号长度为 $N_s = F_s T_s = 2000$. 信号 \mathbf{x}_o 经离散 FFT 变换后, 并取模后得到频域信号 \mathbf{x} , 将该信号作为欠采样的信号进行压缩采样.

记压缩比为 CR, 频域信号长度 $N = N_s$, 则观测向量长度 $M = N/CR$. 易知频域信号 \mathbf{x} 的稀疏度为 $k = 6$ (包含负频), 实验中分别设置不同的稀疏度 $k_1 = 2, k_2 = 4, k_3 = 6, k_4 = 100$ 进行信号重构.

实现代码, 参见文件 [demo_omp_sin3.py](https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_omp_spa) (https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_omp_spa)

压缩比 $CR = 4$ 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构均方误差结果如下:

```
--MSE(x, x1) with k = 2: 2020.810604016191
--MSE(x, x2) with k = 4: 1051.920200728807
--MSE(x, x3) with k = 6: 392.8334107460182
--MSE(x, x4) with k = 100: 49.64637577207484
```

压缩比 $CR = 16$ 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构均方误差结果如下:

```
--MSE(x, x1) with k = 2: 2049.017843656981
--MSE(x, x2) with k = 4: 1091.209561325295
--MSE(x, x3) with k = 6: 411.9556346140723
--MSE(x, x4) with k = 100: 316.7049674291499
```

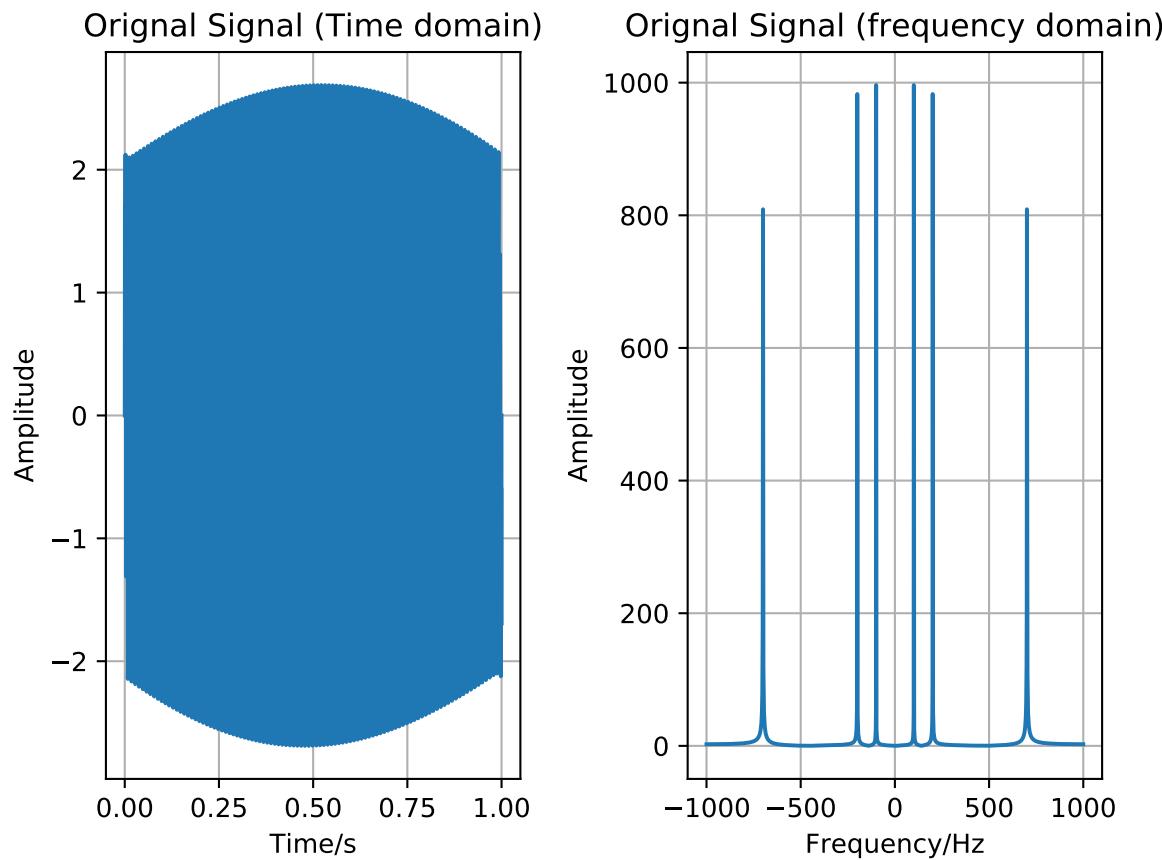


图 5.39: Signal in Time (left, x_o) and Frequency (right, x) domains.

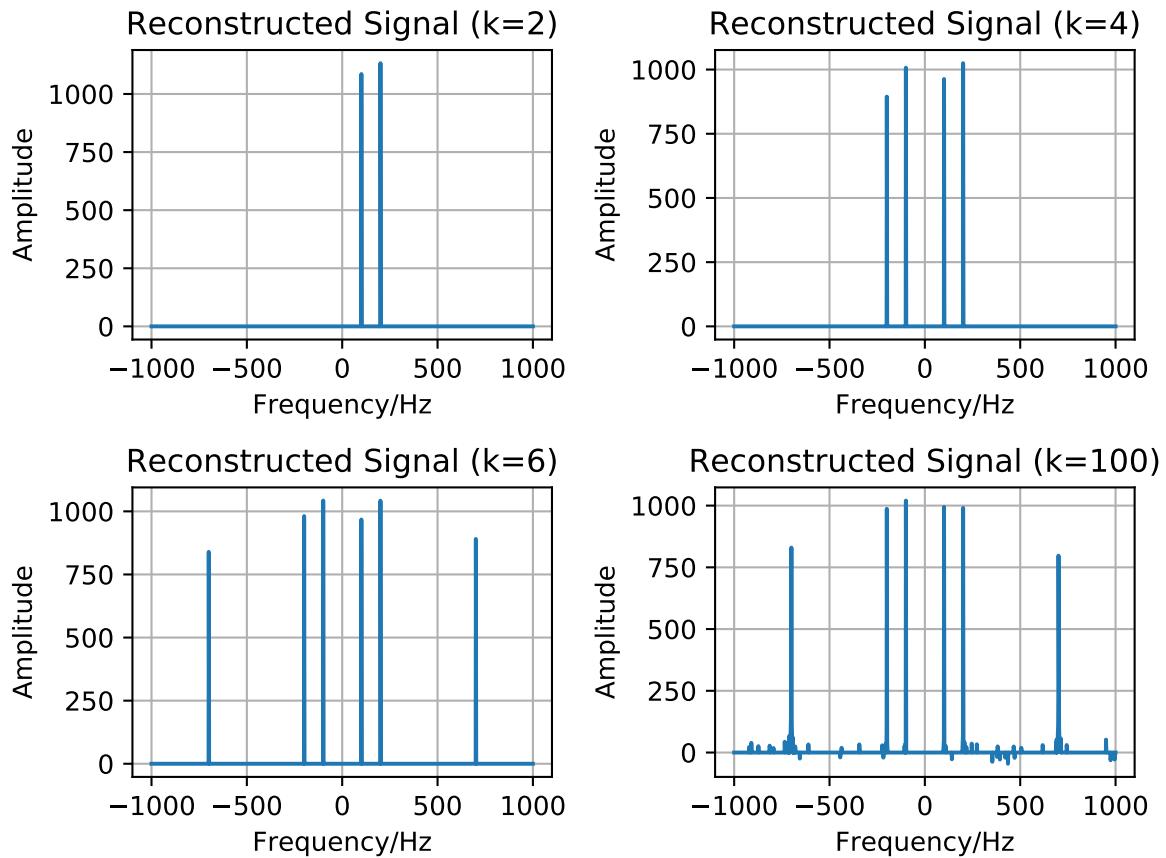


图 5.40: Recovered signal in frequency domain with different sparse degree k .

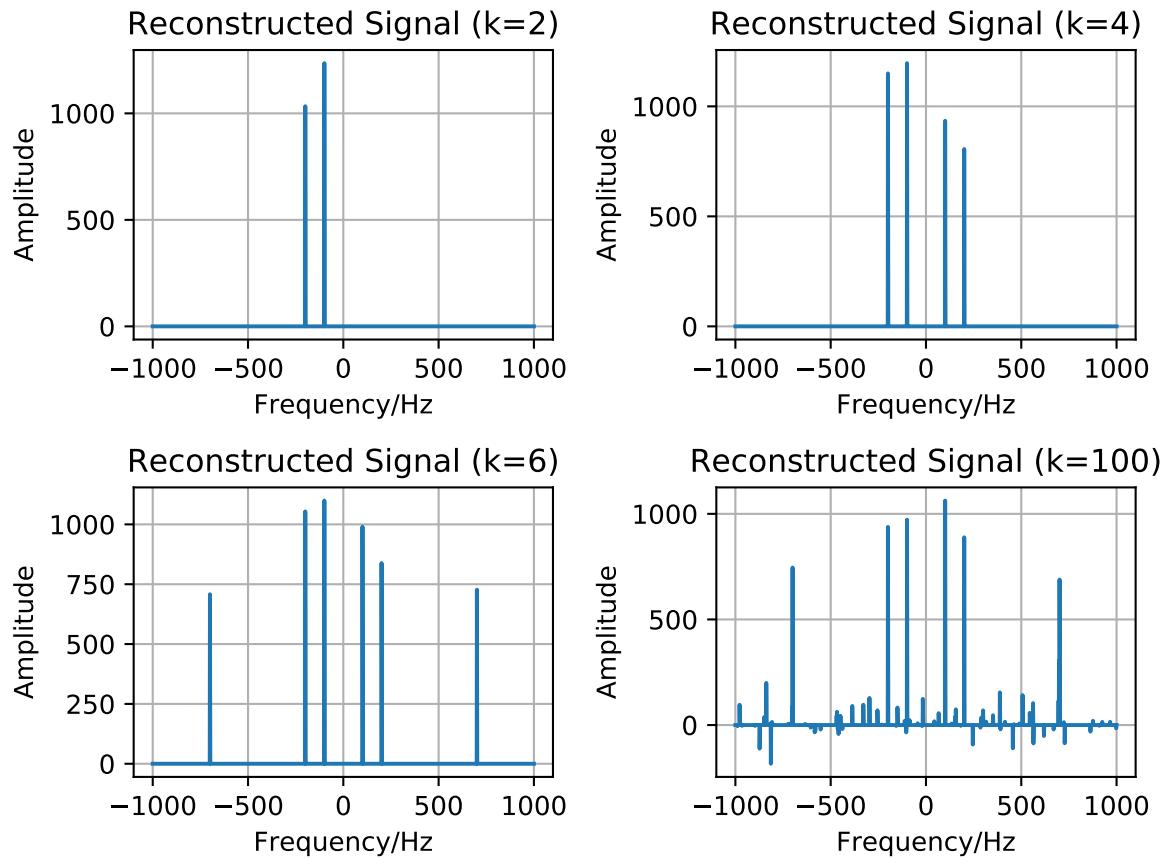


图 5.41: Recovered signal in frequency domain with different sparse degree k .

压缩比 CR = 4 时, DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构均方误差结果如下:

```
--MSE(x, x1) with k = 2: 2965.911207728454
--MSE(x, x2) with k = 4: 2055.5346968994822
--MSE(x, x3) with k = 6: 1082.970762562038
--MSE(x, x4) with k = 100: 1460.2482895078417
```

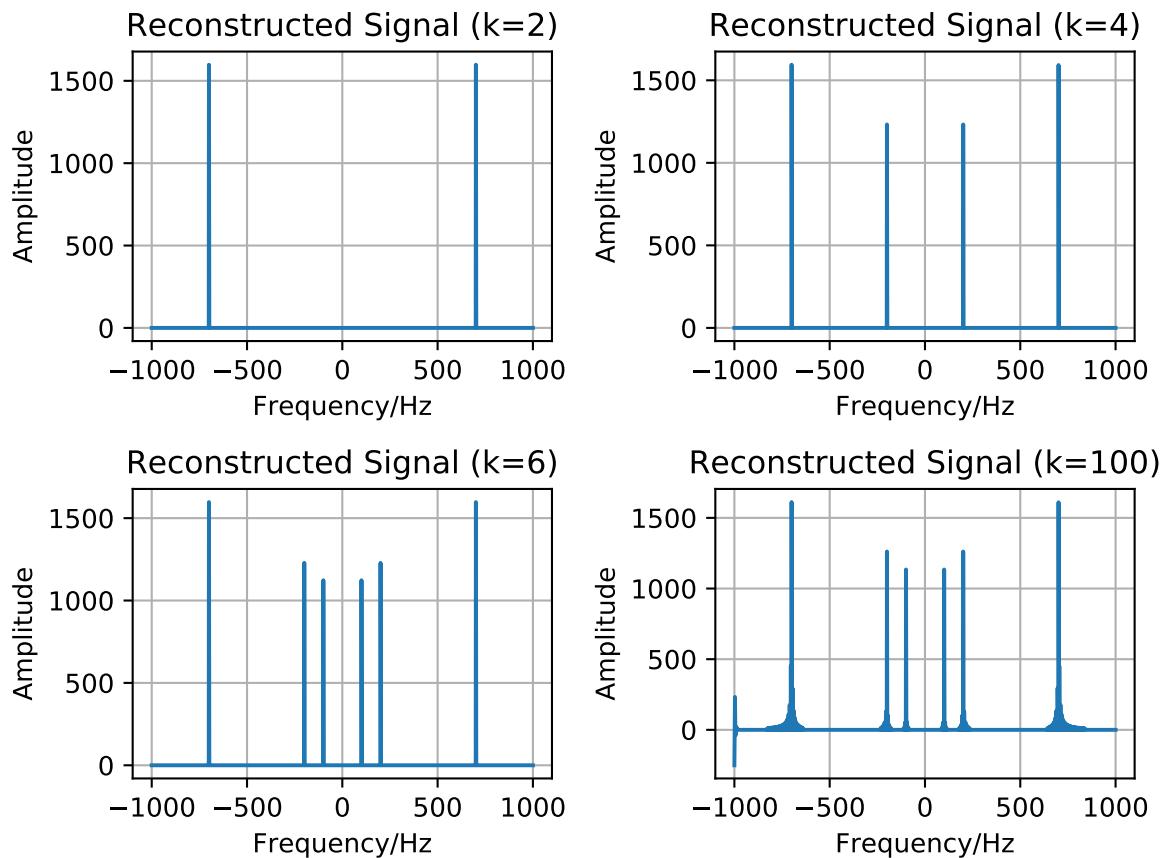


图 5.42: Recovered signal in frequency domain with different sparse degree k .

压缩比 CR = 16 时, DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构均方误差结果如下:

```
--MSE(x, x1) with k = 2: 4805.715254609675
--MSE(x, x2) with k = 4: 4182.01590082246
--MSE(x, x3) with k = 6: 3227.931982399867
--MSE(x, x4) with k = 100: 3760.2420462992363
```

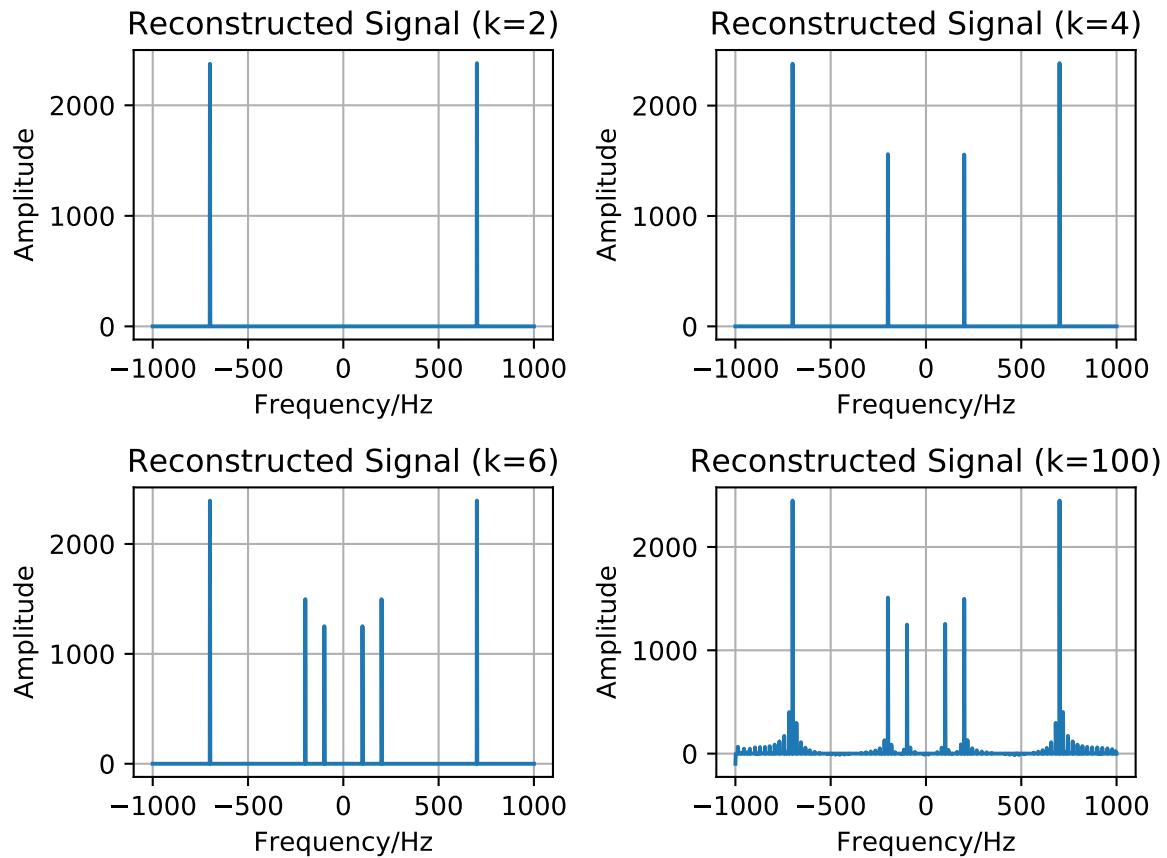


图 5.43: Recovered signal in frequency domain with different sparse degree k .

5.5.5.6.1.2 二维图像信号

5.5.5.6.1.3 实验设置

实验中使用稀疏图像信号作为实验对象，并对图像的每一列采用 OMP 进行恢复。

- 图像信号: \mathbf{X} 为稀疏图像
- 观测矩阵: \mathbf{A} 为高斯矩阵和过完备 DCT 矩阵

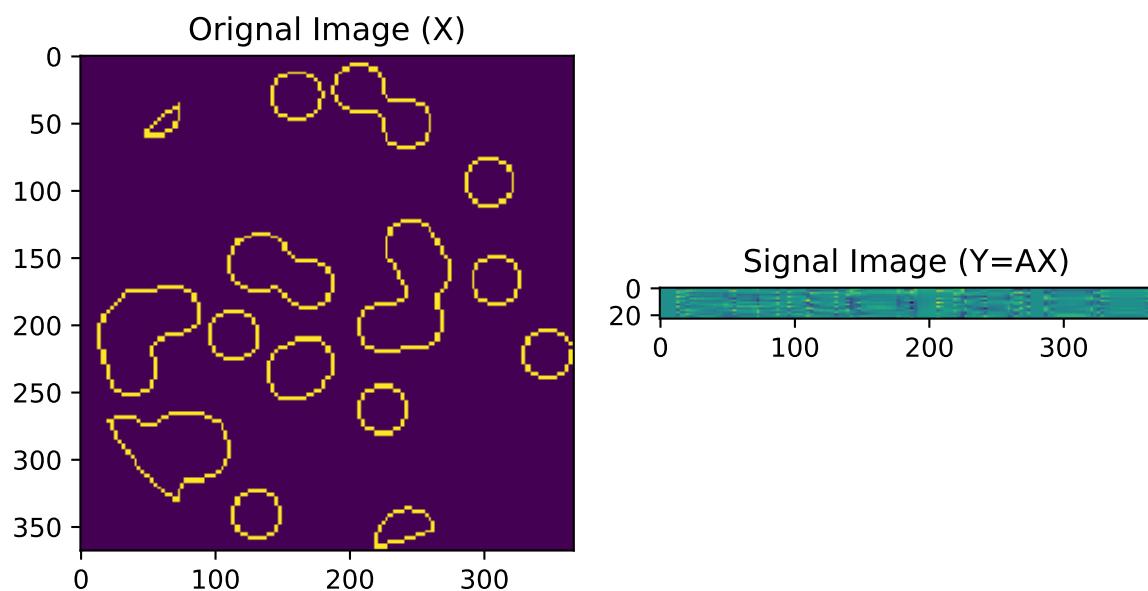


图 5.44: Tomography image

5.5.5.6.1.4 实验代码

实现代码, 参见文件 [demo_omp_sin3.py](https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_omp_img.py) (https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_omp_img.py)

5.5.5.6.1.5 实验结果

压缩比 CR = 4 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 15.580596793850813 1798.9565809738183 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 16.51005632425414 1452.36150189551 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 16.236071194072142 1546.9391881886033 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 15.509964880400881 1828.4533008529795 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 14.650356290567599 2228.6646872408483 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 13.870332301892105 2667.148094123919 255 uint8

[Finished in 1475.4s]
```

压缩比 CR = 16 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 12.971158037789456 3280.6851824428386 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 12.936695435825893 3306.8219923587253 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 14.334155129944055 2396.9824147020495 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 14.869213943506328 2119.136669233503 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 14.284916845043584 2424.312922047151 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 13.839801158638902 2685.9643555265966 255 uint8

[Finished in 1470.3s]
```

压缩比 CR = 4 时, 一维 DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 14.440057558389176 2339.239048987087 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 14.407312672319945 2356.9430754649925 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 14.392225830410263 2365.1450361331595 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 14.438509731511893 2340.0729030920506 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 15.093057465950091 2012.6794588911266 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 14.015175521723268 2579.6620142755596 255 uint8

[Finished in 1378.4s]
```

压缩比 CR = 16 时, 一维 DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 12.971158037789456 3280.6851824428386 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 12.936695435825893 3306.8219923587253 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 14.334155129944055 2396.9824147020495 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 14.869213943506328 2119.136669233503 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 14.284916845043584 2424.312922047151 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 13.839801158638902 2685.9643555265966 255 uint8
```

(下页继续)

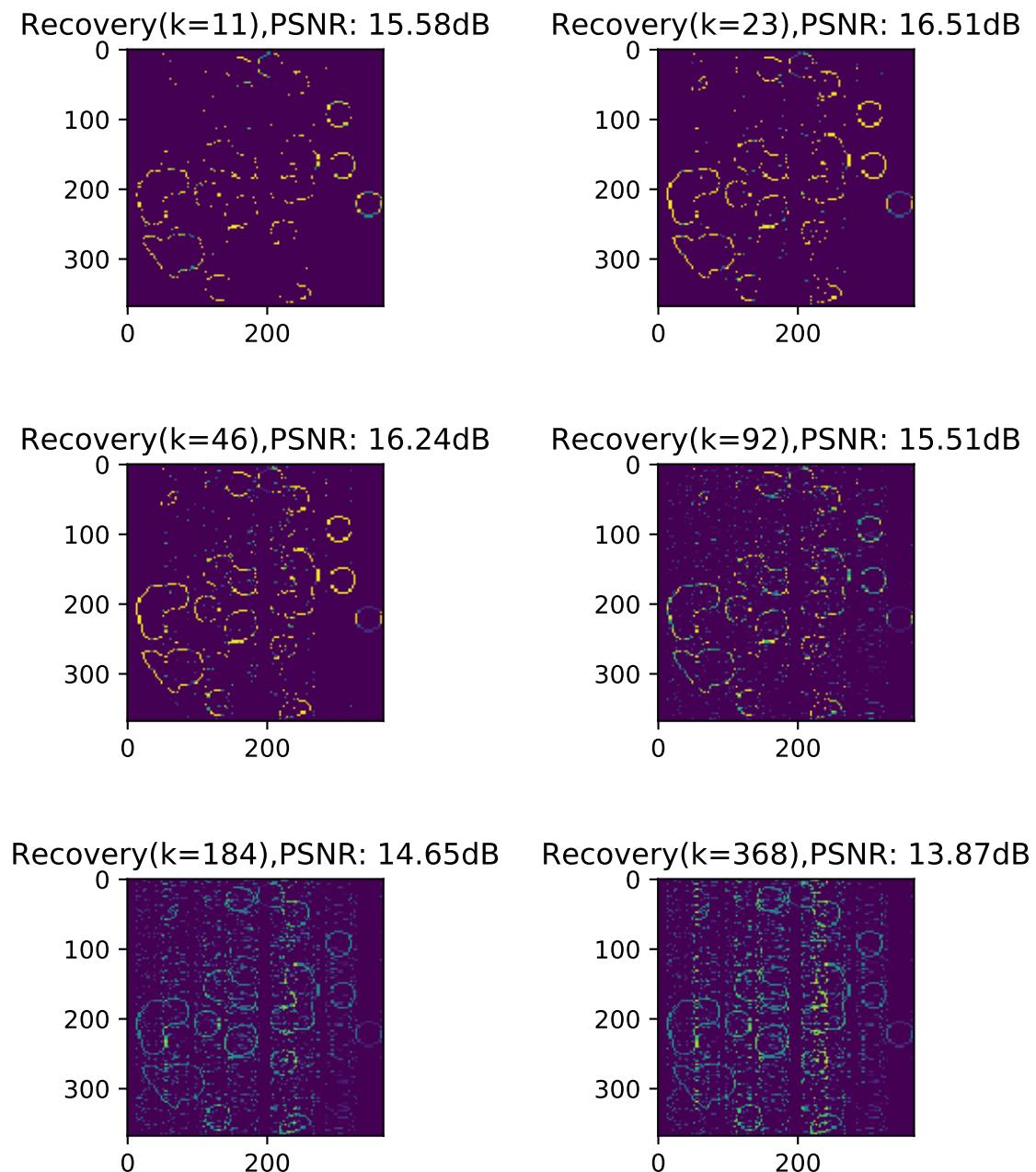


图 5.45: Recovered tomography image with different sparse degree k .

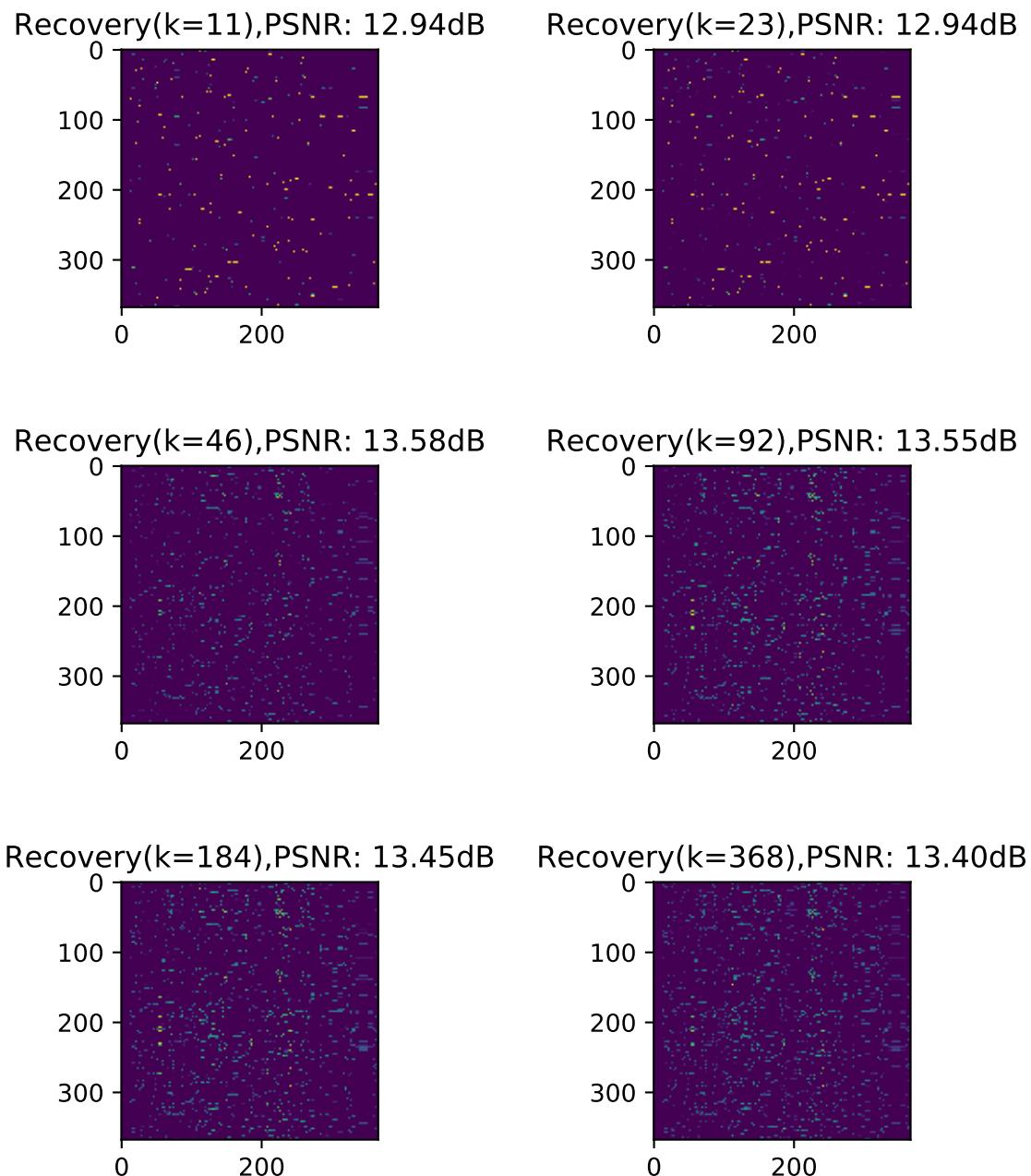


图 5.46: Recovered tomography image with different sparse degree k .

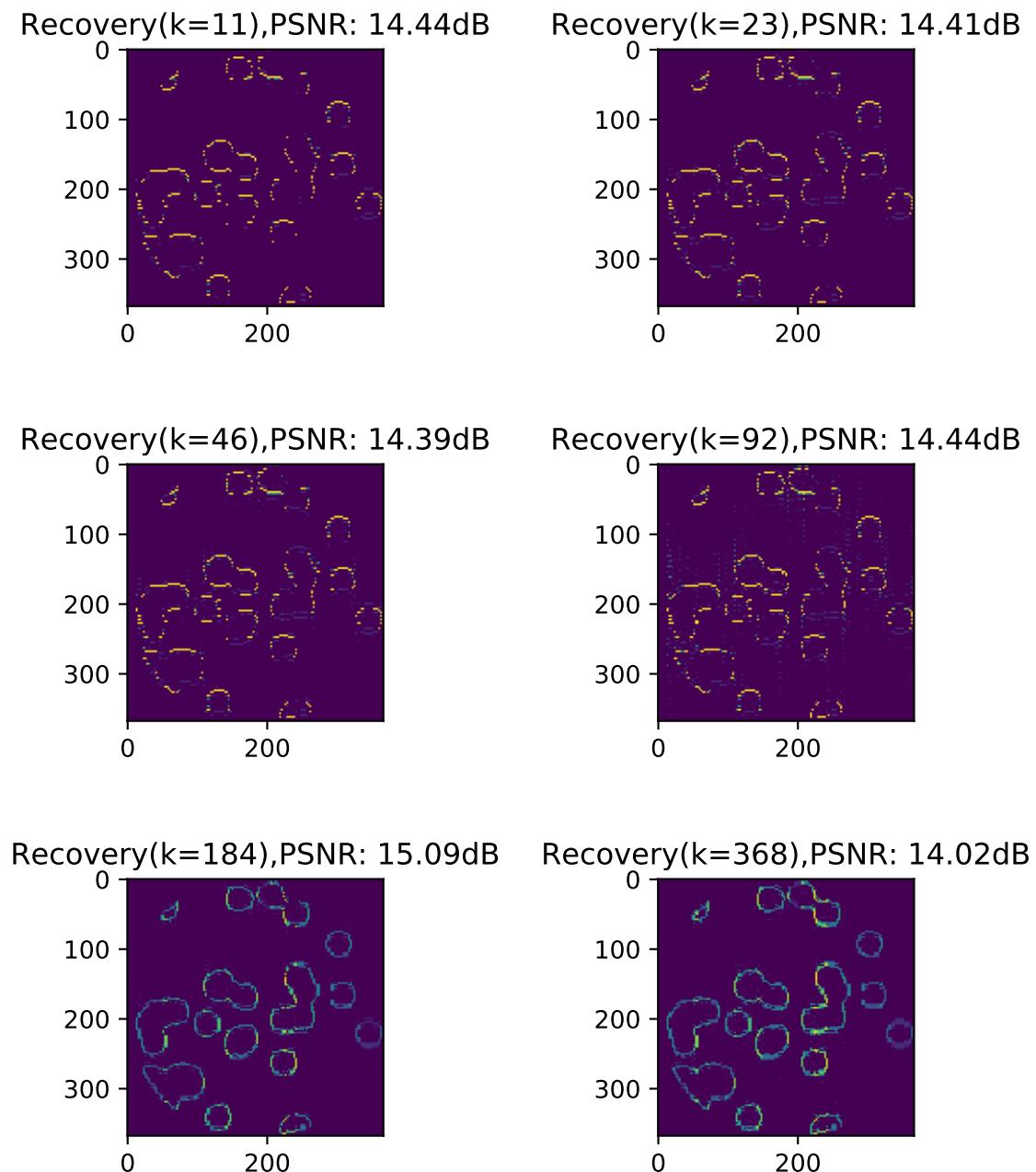


图 5.47: Recovered tomography image with different sparse degree k .

(续上页)

```
[Finished in 1380.3s]
```

5.5.5.6.2 非稀疏信号 CS

5.5.5.6.2.1 压缩感知方式实验

Python 实现参见文件 [demo_csTwoWayAnalysis.py](#) (https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_csTwoWayAnalysis.py)

以 cameraman 图像为例, 两种非稀疏信号的压缩采样与恢复结果如下

压缩比设置为 2, 采用 OMP 算法重构图像信号, 稀疏度设置为 $k = 32$ 时的结果为

压缩比设置为 4, 采用 OMP 算法重构图像信号, 稀疏度设置为 $k = 32$ 时的结果为

两种非稀疏信号的压缩采样与恢复方式表示为

方式 1

1. 变换: $\mathbf{z} = \Psi_1 \mathbf{x}$
2. 观测: $\mathbf{y} = \Phi \mathbf{z}$,
3. 求解: $\min_{\mathbf{z}} \|\mathbf{z}\|_p \text{ s.t. } \mathbf{y} = \Phi \mathbf{z}$
4. 重构: $\mathbf{x} = \Psi_2 \mathbf{z}$

方式 2

1. 观测: $\mathbf{y} = \Phi \mathbf{x}$
2. 求解: $\min_{\mathbf{z}} \|\mathbf{z}\|_p \text{ s.t. } \mathbf{y} = \Phi \Psi_1 \mathbf{z}$
3. 重构: $\mathbf{x} = \Psi_2 \mathbf{z}$

记 \mathbf{D} 为正交 DCT 变换矩阵 ($\mathbf{D}^{-1} = \mathbf{D}^T$), 使用不同的 Ψ_1, Ψ_2 组合, 衡量两种 CS 方法重构图像与原图间的均方误差, 结果为

Ψ_1	Ψ_2	方式 1	方式 2
\mathbf{D}	\mathbf{D}	3802	743
\mathbf{D}^T	\mathbf{D}	770	3816
\mathbf{D}	\mathbf{D}^T	616	3754
\mathbf{D}^T	\mathbf{D}^T	3874	589

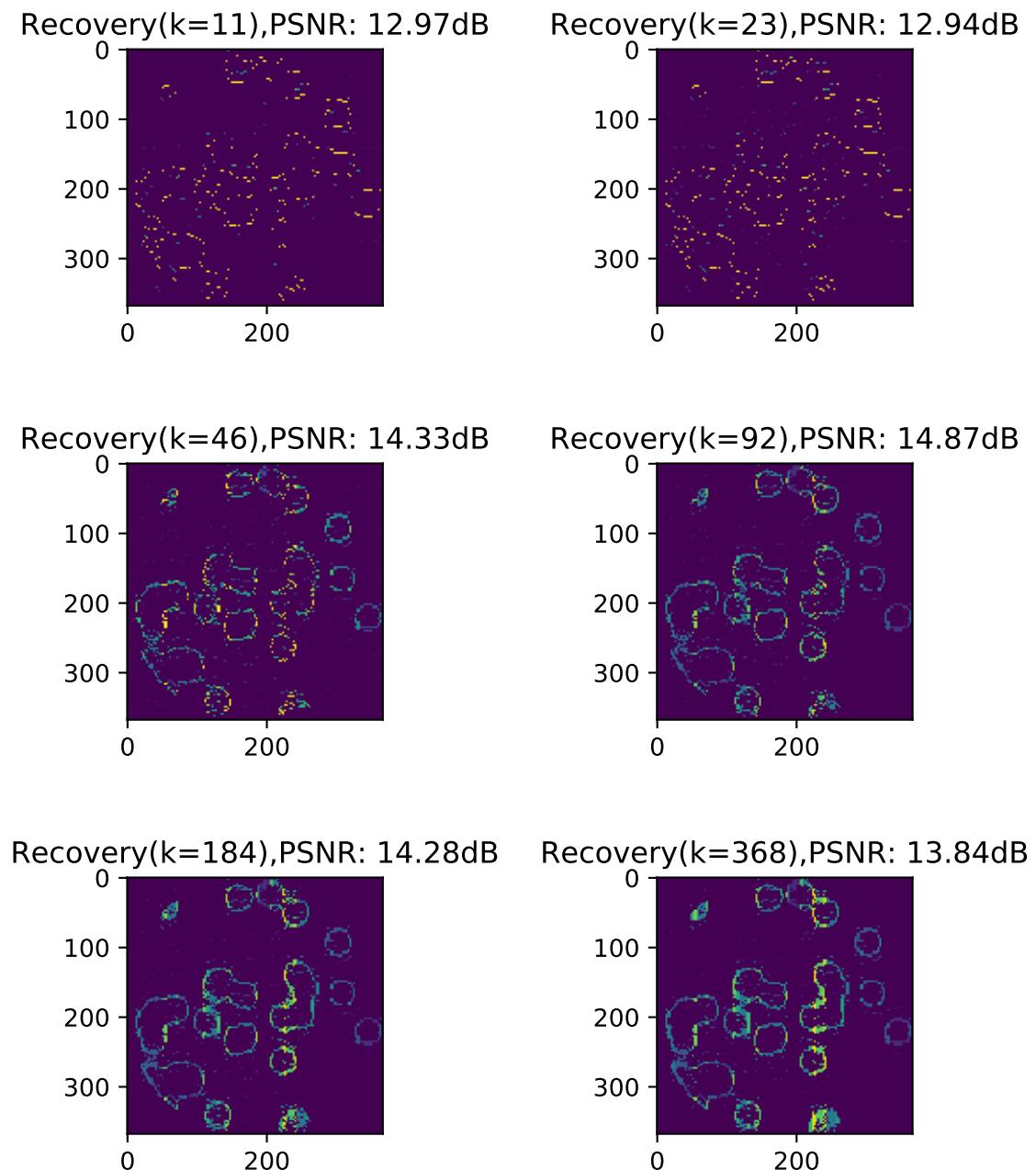
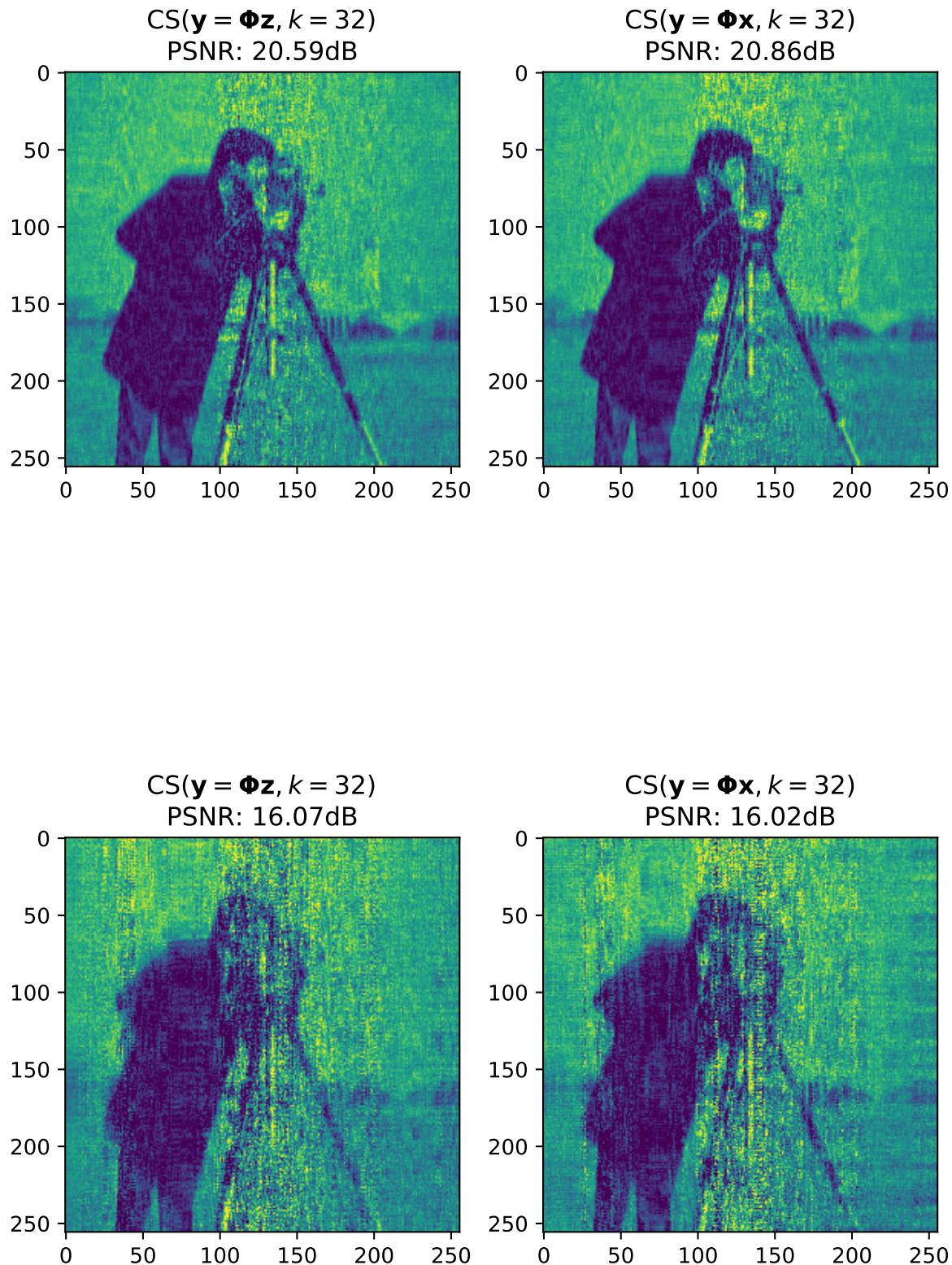


图 5.48: Recovered tomography image with different sparse degree k .



5.5.5.6.2.2 一维压缩感知实验

实验完成对二维图像的高度维压缩采样与恢复.

5.5.5.6.2.3 无稀疏表示

- 图像信号: $H \times W = 256 \times 256$
- 表示字典: 无
- 观测矩阵: Gaussian 观测矩阵, 一维 DCT 过完备观测矩阵, 尺寸 $M \times N$, $N = 256$
- 压缩比率: CR = N/M

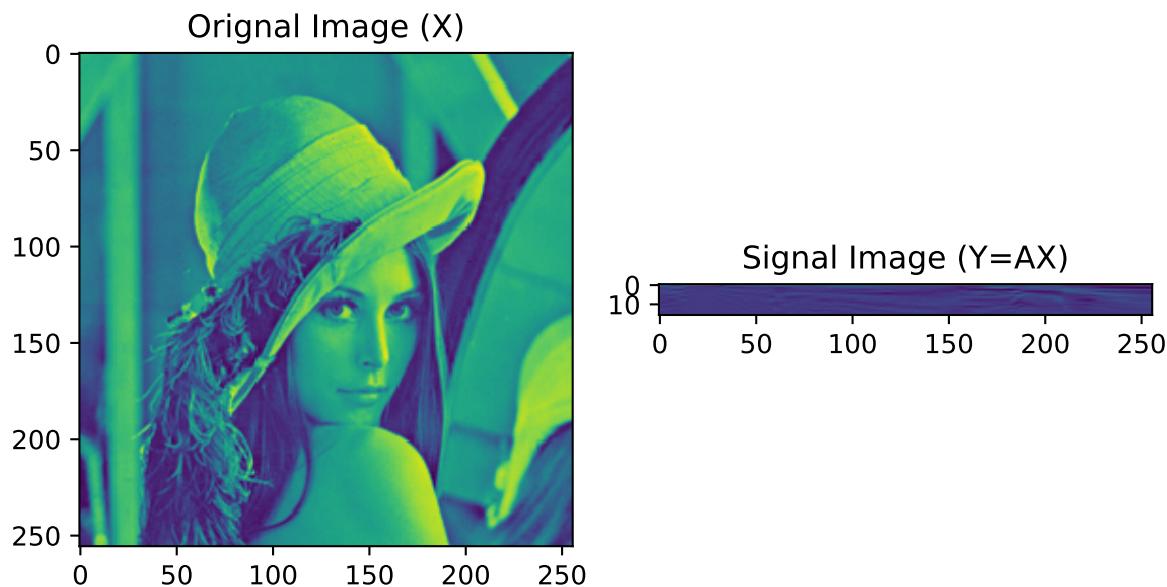


图 5.49: Lena image

压缩比 CR = 4 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 5.22232377711354 19536.554946899414 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 5.251123223918265 19407.430450439453 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 5.356613492689134 18941.702514648438 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 5.456996728640617 18508.903366088867 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 6.588493687182973 14263.660461425781 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 6.0725505137777835 16062.950073242188 255 uint8
```

(下页继续)

(续上页)

```
[Finished in 431.7s]
```

压缩比 CR = 16 时, Gaussian 观测矩阵, 设置不同稀疏度下的信号重构结果:

```
--PSNR1, MSE1, Vpeak1, dtype: 5.210776416835353 19588.569381713867 255 uint8
--PSNR2, MSE2, Vpeak2, dtype: 5.229416619643459 19504.67413330078 255 uint8
--PSNR3, MSE3, Vpeak3, dtype: 5.409661839488037 18711.740112304688 255 uint8
--PSNR4, MSE4, Vpeak4, dtype: 5.444672361916549 18561.502349853516 255 uint8
--PSNR5, MSE5, Vpeak5, dtype: 5.33433084335866 19039.137771606445 255 uint8
--PSNR6, MSE6, Vpeak6, dtype: 5.271223842201596 19317.813842773438 255 uint8
```

```
[Finished in 374.4s]
```

压缩比 CR = 4 时, 一维 DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构结果:

```
--PSNR1, MSE1, Vpeak1, dtype: 5.343276737179728 18999.960021972656 255 uint8
--PSNR2, MSE2, Vpeak2, dtype: 5.4361581642123955 18597.92724609375 255 uint8
--PSNR3, MSE3, Vpeak3, dtype: 5.456286226270971 18511.93165588379 255 uint8
--PSNR4, MSE4, Vpeak4, dtype: 5.477199979749907 18423.000457763672 255 uint8
--PSNR5, MSE5, Vpeak5, dtype: 6.330385584444022 15137.069412231445 255 uint8
--PSNR6, MSE6, Vpeak6, dtype: 10.306663047919827 6059.182815551758 255 uint8
```

```
[Finished in 426.0s]
```

压缩比 CR = 16 时, 一维 DCT 过完备观测矩阵, 设置不同稀疏度下的信号重构结果:

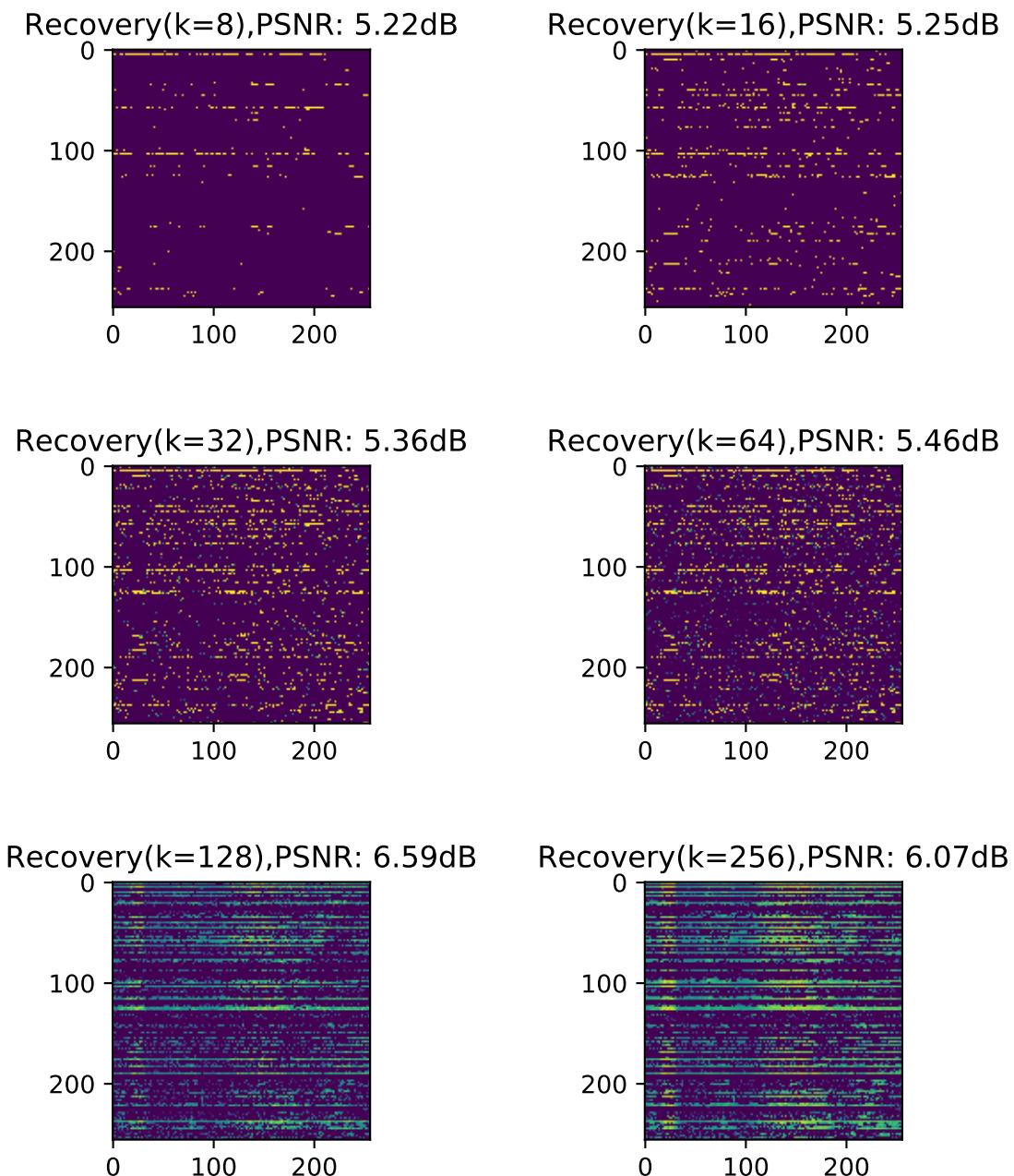
```
--PSNR1, MSE1, Vpeak1, dtype: 5.241856072986026 19448.88702392578 255 uint8
--PSNR2, MSE2, Vpeak2, dtype: 5.225959176230842 19520.208099365234 255 uint8
--PSNR3, MSE3, Vpeak3, dtype: 5.354500199748536 18950.92185974121 255 uint8
--PSNR4, MSE4, Vpeak4, dtype: 5.515306696304636 18262.056884765625 255 uint8
--PSNR5, MSE5, Vpeak5, dtype: 6.040420221353212 16182.22885131836 255 uint8
--PSNR6, MSE6, Vpeak6, dtype: 9.144875858455052 7917.585739135742 255 uint8
```

```
[Finished in 383.2s]
```

5.5.6.2.4 含稀疏表示

- 图像信号: $H \times W = 256 \times 256$
- 表示字典: DCT, 尺寸 256×256
- 观测矩阵: Gaussian, 尺寸 $M \times N$, $N = 256$
- 压缩比率: $CR = N/M$

实现代码, 参见文件 [demo_cs1h_omp_img.py](https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_cs1h_omp_img.py) (https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_cs1h_omp_img.py)

图 5.50: Recovered lena image with different sparse degree k .

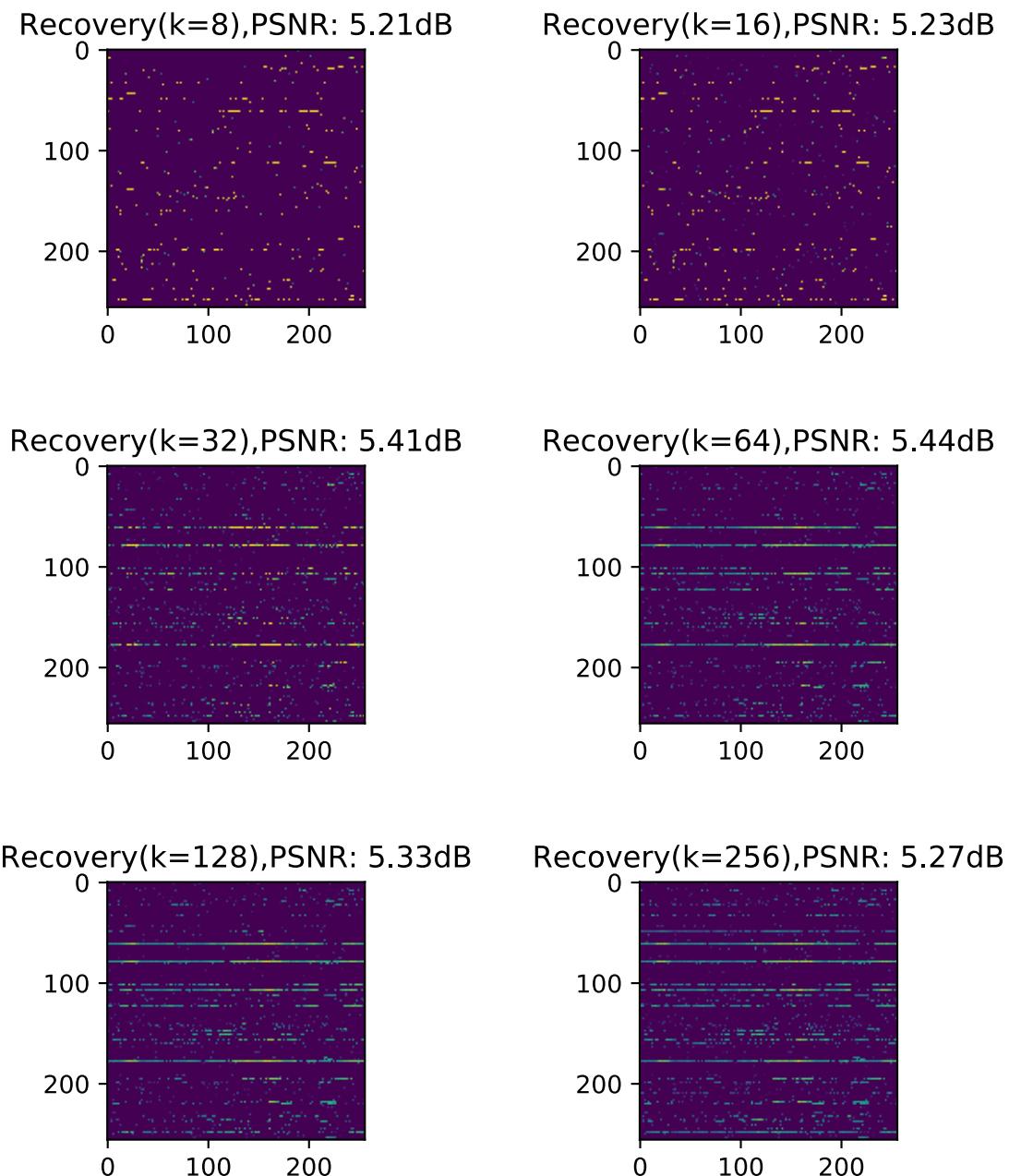
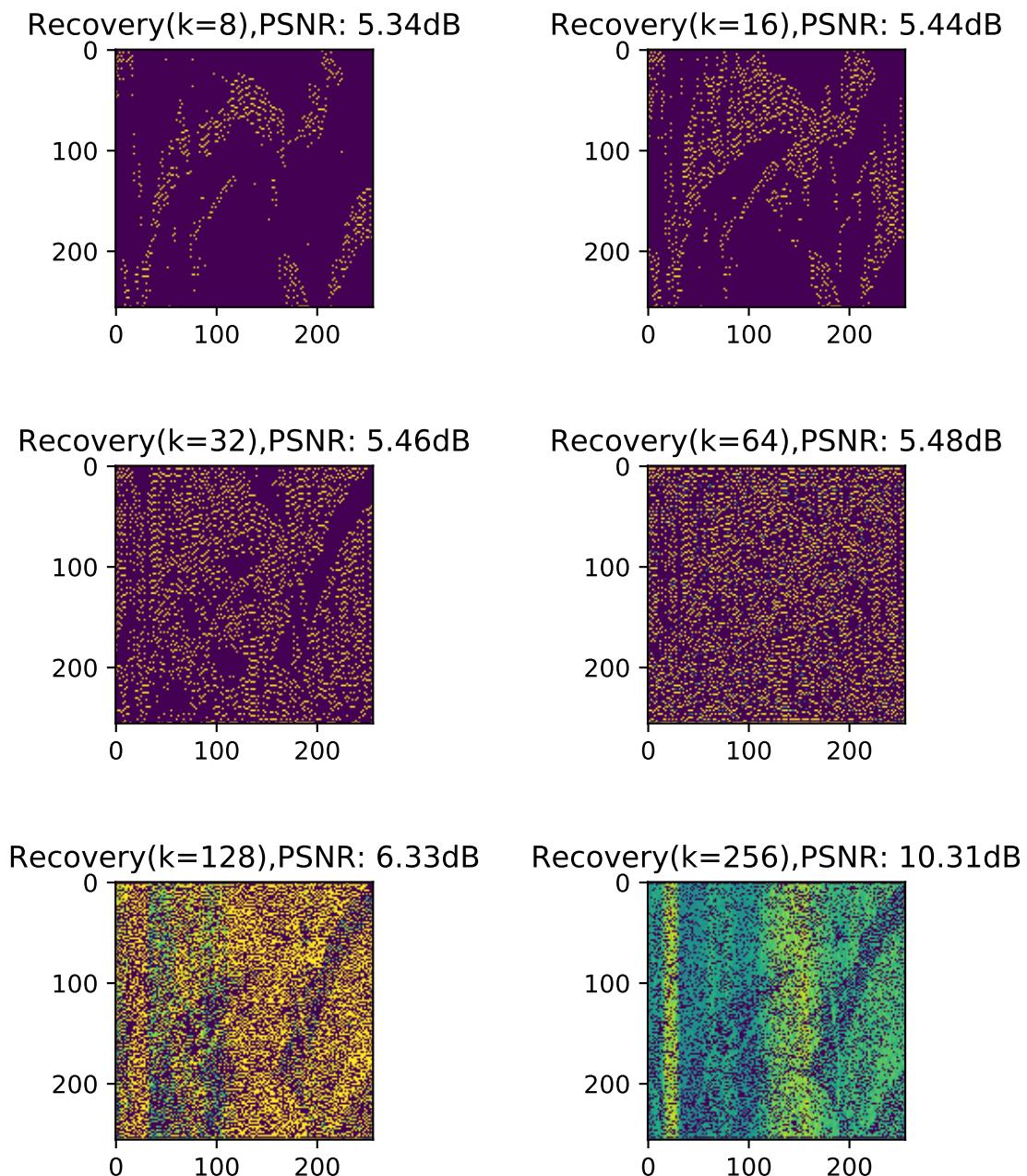


图 5.51: Recovered lena image with different sparse degree k .

图 5.52: Recovered lena image with different sparse degree k .

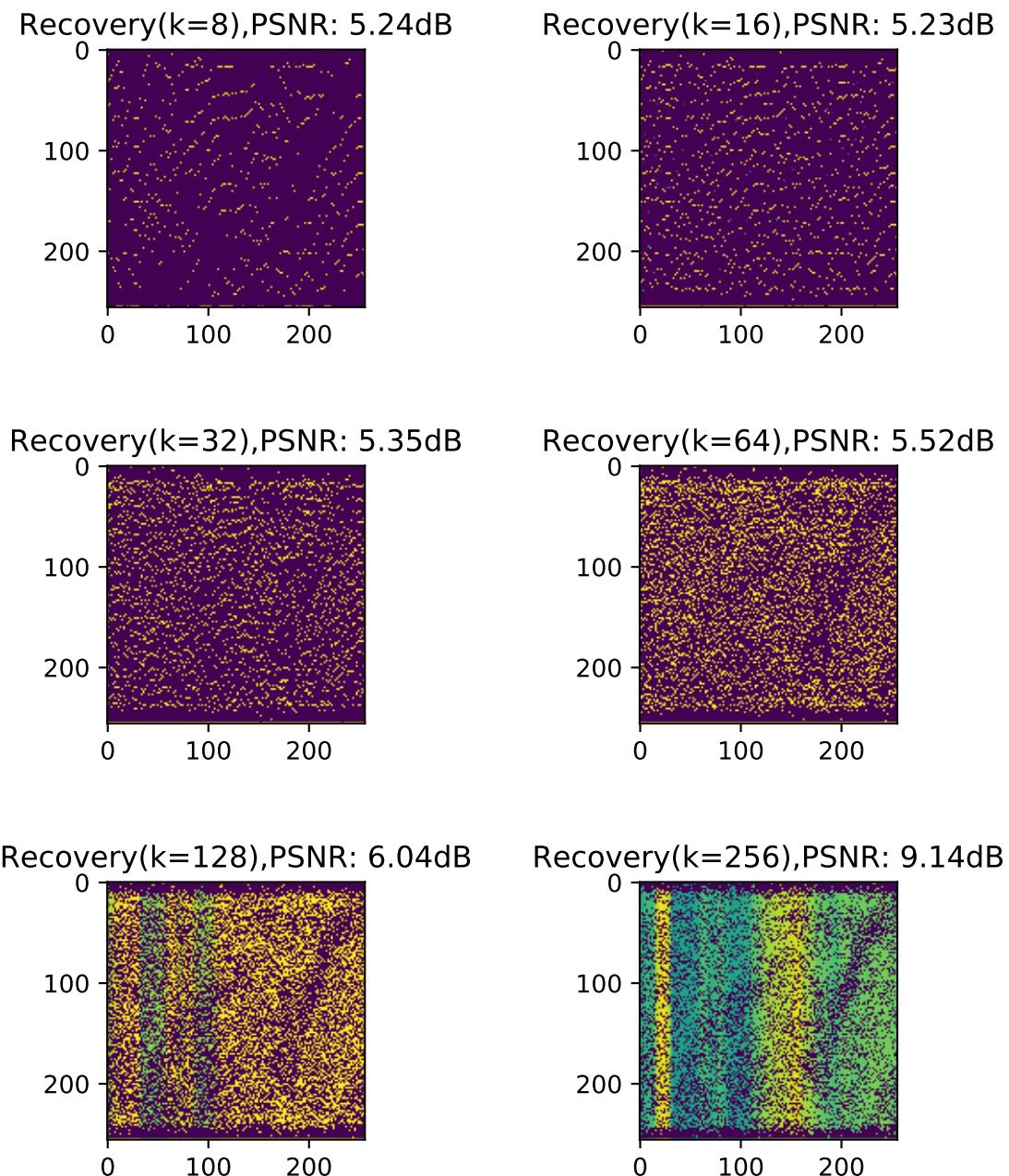


图 5.53: Recovered lena image with different sparse degree k .

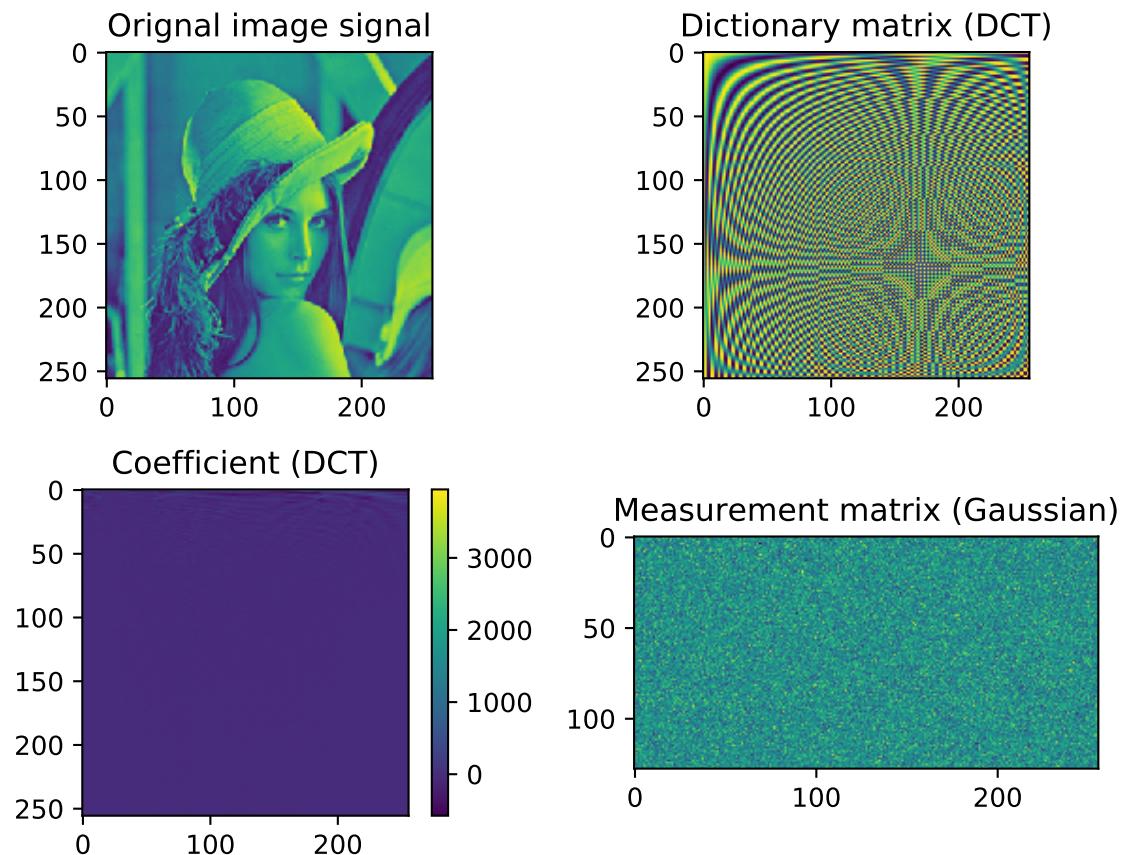


图 5.54: Original image signal, dictionary matrix, observation matrix, sparse coefficient matrix.

压缩比 CR = 2 时, Lena 图像重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 19.35324312034611 754.6681976318359 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 21.35199359181007 476.3004608154297 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 22.084221813206373 402.4001922607422 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 21.93230942365929 416.72486877441406 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 18.413869479192858 936.9003753662109 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 5.772999668544614 17209.98112487793 255 uint8
[Finished in 754.8s]
```

压缩比 CR = 4 时, Lena 图像重构结果:

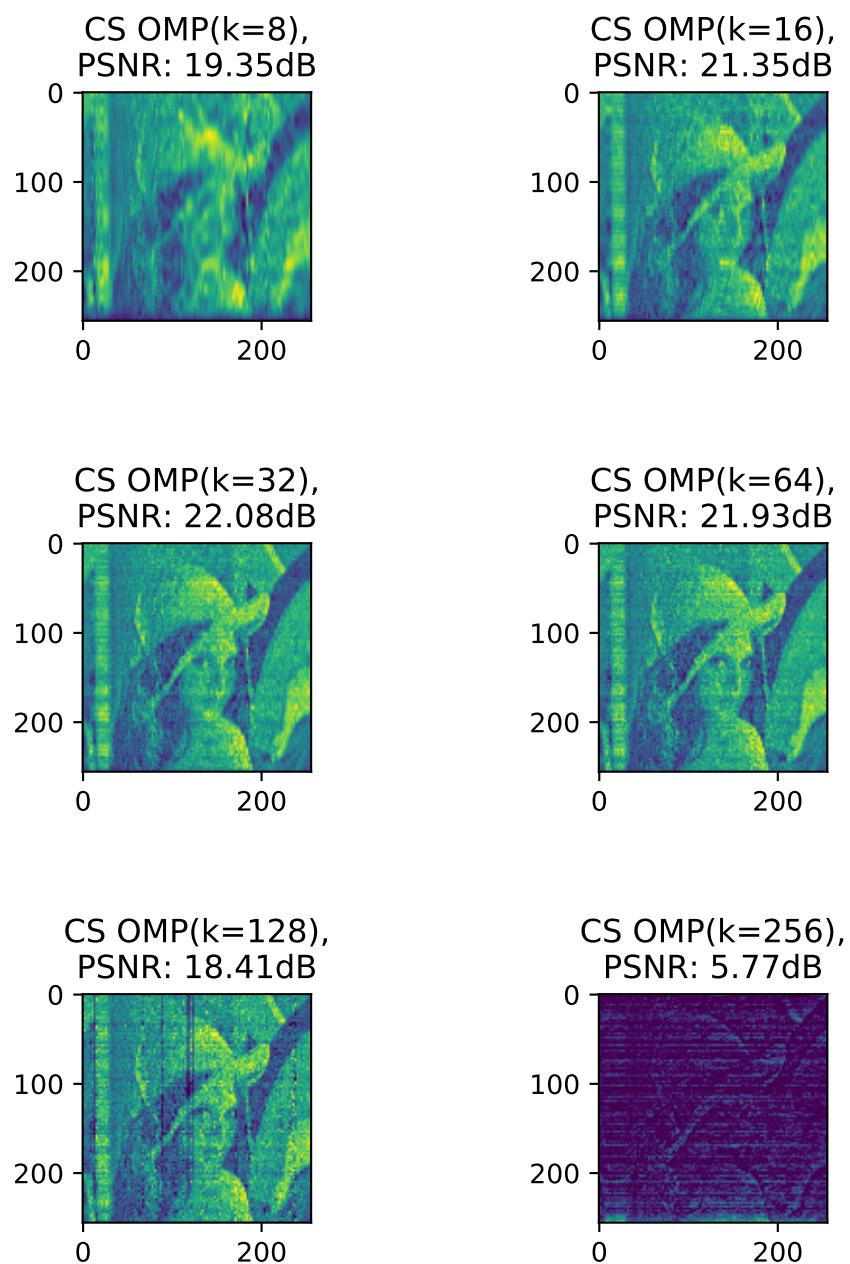
```
---PSNR1, MSE1, Vpeak1, dtype: 17.30050349526095 1210.6817932128906 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 17.13251590038045 1258.4291534423828 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 16.41341724769021 1485.0416564941406 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 15.891063092434477 1674.8428039550781 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 6.054397672821921 16130.231246948242 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 5.569662696862794 18034.914642333984 255 uint8
[Finished in 352.2s]
```

压缩比 CR = 8 时, Lena 图像重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 12.711132296542491 3483.1095275878906 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 12.24455126500644 3878.1556396484375 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 12.08576911176013 4022.5685119628906 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 6.243422364533981 15443.229461669922 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 5.658814132793683 17668.470169067383 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 5.437810904144429 18590.851013183594 255 uint8
[Finished in 380.7s]
```

压缩比 CR = 16 时, Lena 图像重构结果:

```
---PSNR1, MSE1, Vpeak1, dtype: 12.530425704027621 3631.0964965820312 255 uint8
---PSNR2, MSE2, Vpeak2, dtype: 12.46201822823682 3688.744186401367 255 uint8
---PSNR3, MSE3, Vpeak3, dtype: 6.453316341931616 14714.609008789062 255 uint8
---PSNR4, MSE4, Vpeak4, dtype: 5.758194148603205 17268.751739501953 255 uint8
---PSNR5, MSE5, Vpeak5, dtype: 5.461627943974914 18489.17642211914 255 uint8
---PSNR6, MSE6, Vpeak6, dtype: 5.355063389623071 18948.464477539062 255 uint8
[Finished in 396.7s]
```

图 5.55: Recovery results ($\text{CR} = 2$) with different sparse degree k .

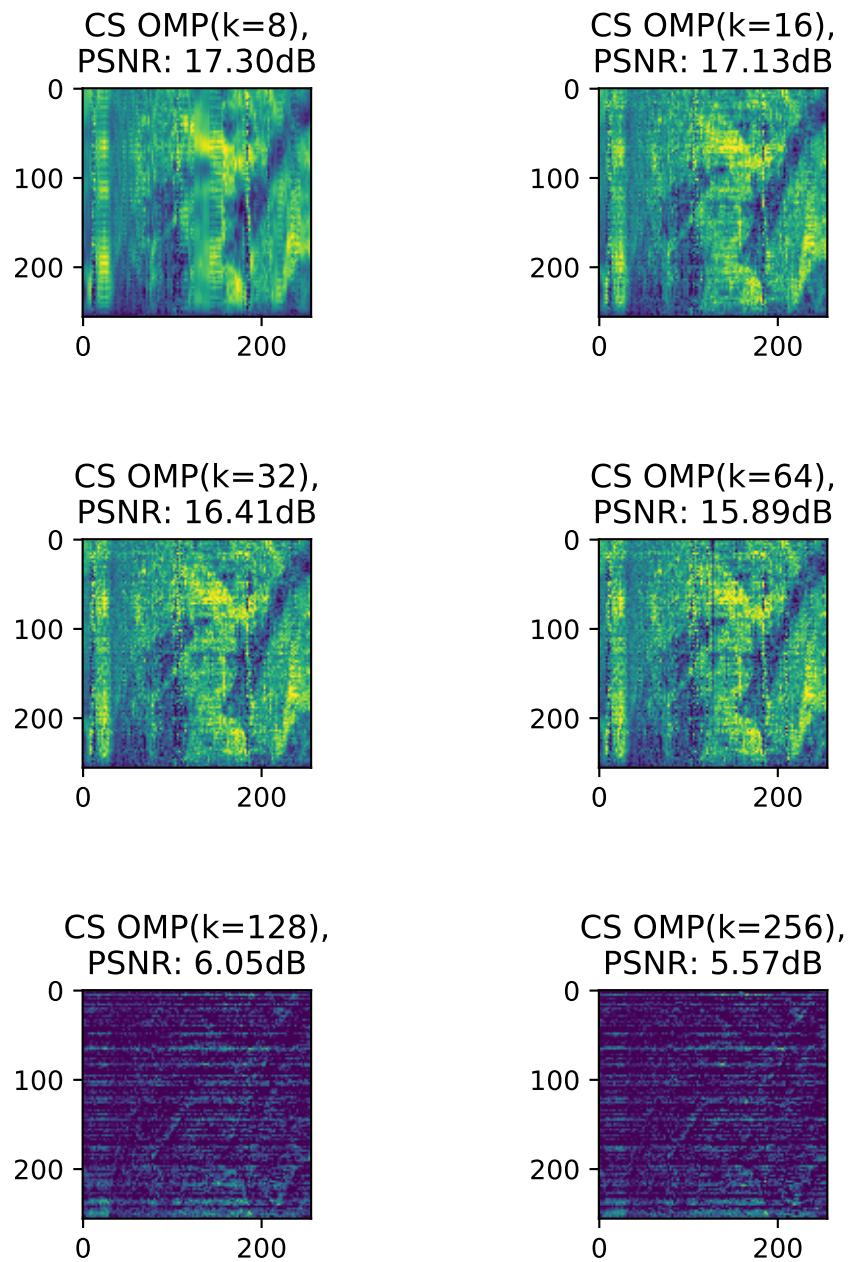


图 5.56: Recovery results ($\text{CR} = 4$) with different sparse degree k .

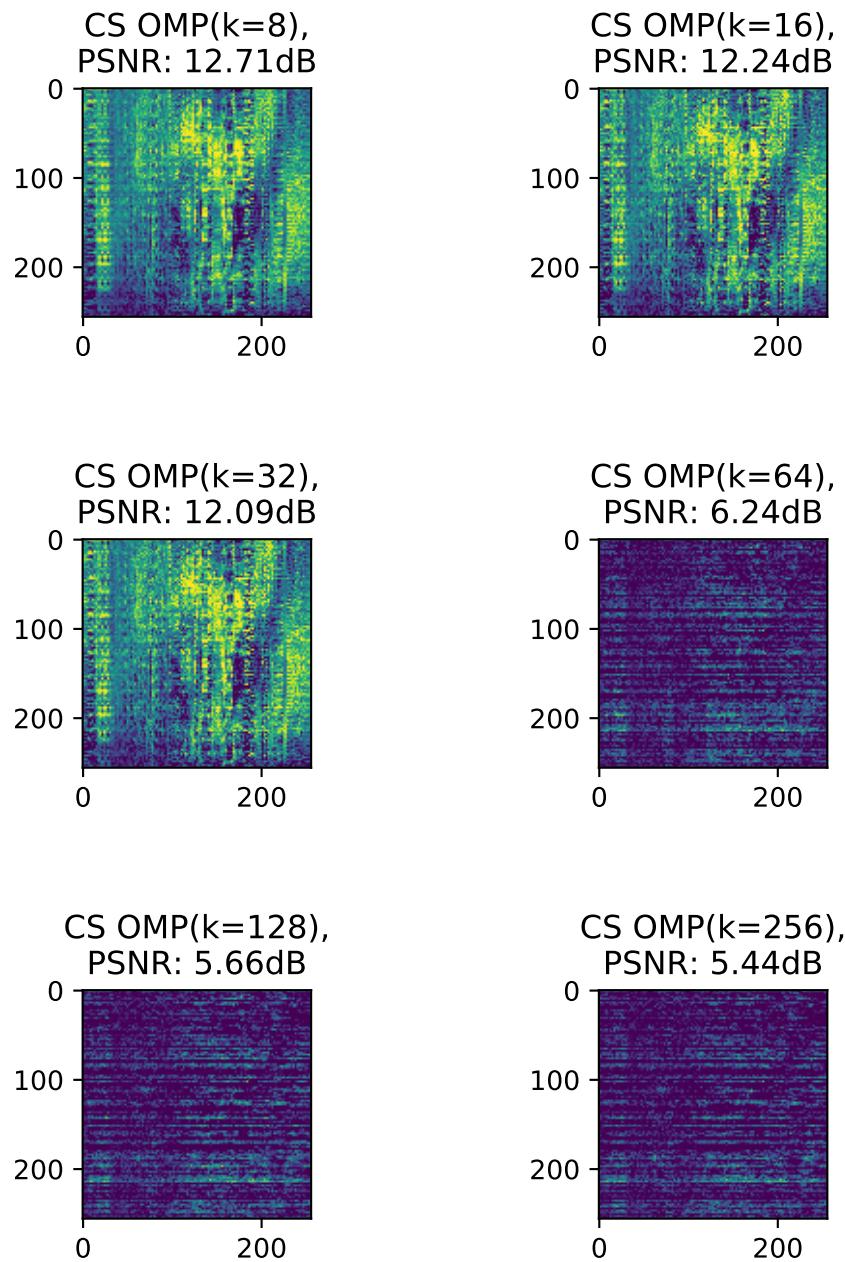


图 5.57: Recovery results ($\text{CR} = 8$) with different sparse degree k .

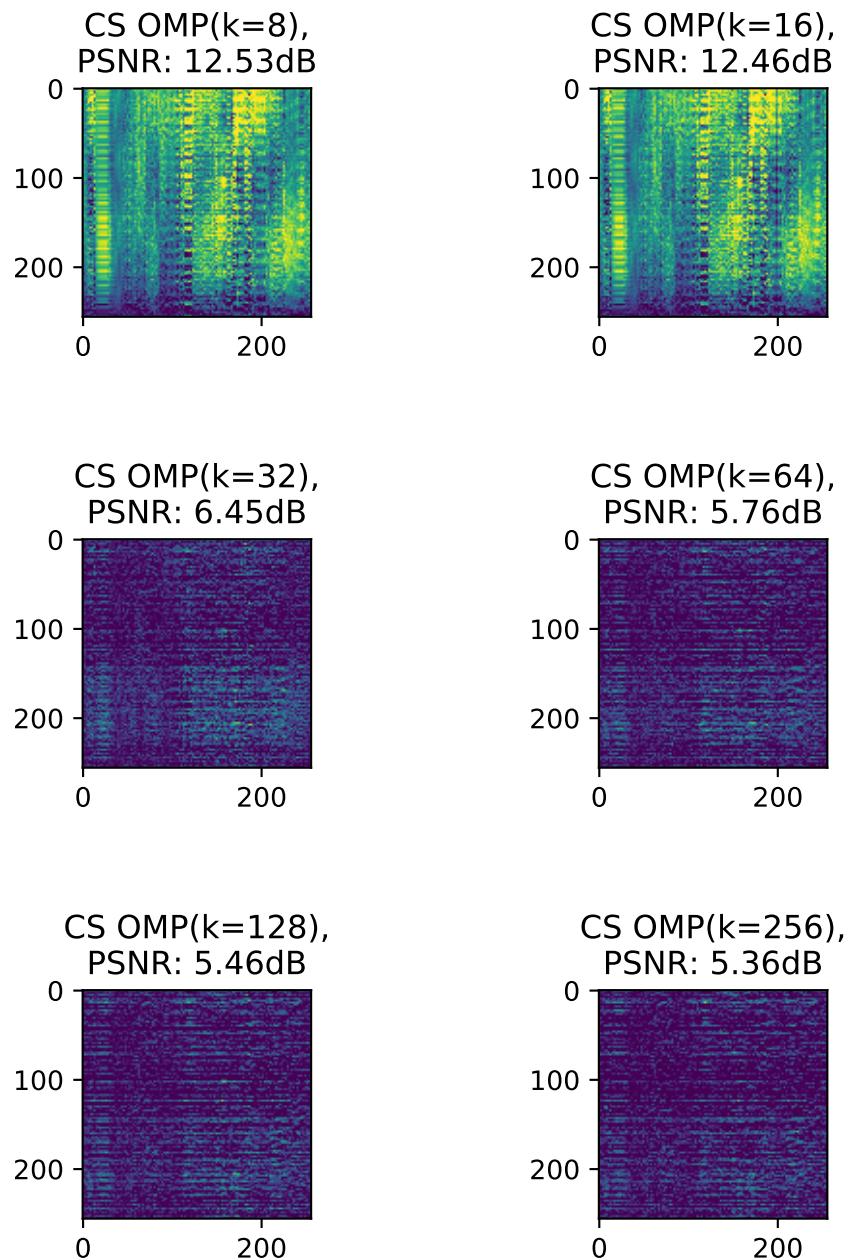


图 5.58: Recovery results ($\text{CR} = 16$) with different sparse degree k .

5.5.5.6.2.5 二维压缩感知实验 1

实验完成对二维图像的高度维和宽度维压缩采样与恢复, 实验中将原始图像拉成列向量, 应用压缩感知理论进行采样模拟与恢复.

5.5.5.6.2.6 二维压缩感知实验 2

实验完成对二维图像的高度维和宽度维压缩采样与恢复, 实验中, 应用压缩感知理论, 对图像信号分别进行高度维和宽度维的两次压缩采样与恢复.

5.5.5.6.3 复压缩感知实验

5.5.5.6.3.1 核磁共振成像

对核磁共振图像 (MRI) 原始 k-space 数据进行行方向上的压缩采样与恢复, 采用 OMP 算法恢复信号.

5.5.5.6.3.2 实验代码

实现代码, 参见文件 [demo_cs1h_01.py](https://github.com/antsfamily/pysparse/tree/master/examples/LinearCS/demo_cs1h_01.py)

5.5.5.6.3.3 实验结果

压缩比 CR = 4, 无稀疏表示字典时, MRI 图像重构结果:

压缩比 CR = 4, DCT 稀疏表示字典时, MRI 图像重构结果:

5.5.6 学习式压缩感知

5.5.6.1 引言

贝叶斯压缩感知 (Bayesian Compressive Sensing, BCS)

5.5.6.1.1 Resources

- **Compressive Sensing Resources** [Compressive Sensing Resources](http://dsp.rice.edu/cs/) (<http://dsp.rice.edu/cs/>) : research publications software courses
- **Bayesian Compressive Sensing** [Derin Babacan's homepage](http://www.dbabacan.info/software.html) (<http://www.dbabacan.info/software.html>) : papers and codes, :cite:S.D.Babacan.2010

*

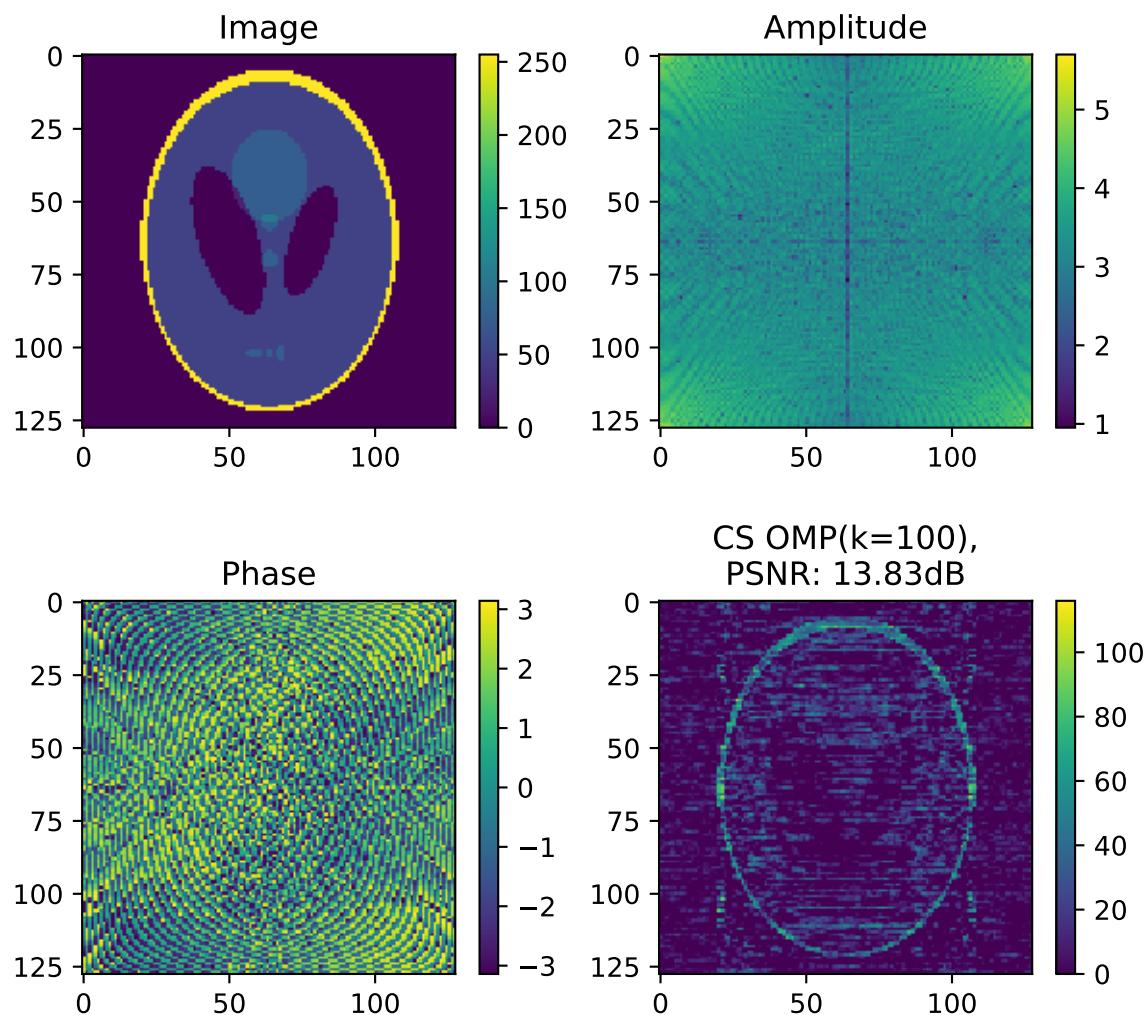


图 5.59: Recovery results ($\text{CR} = 4$) with sparse degree $k = 100$.

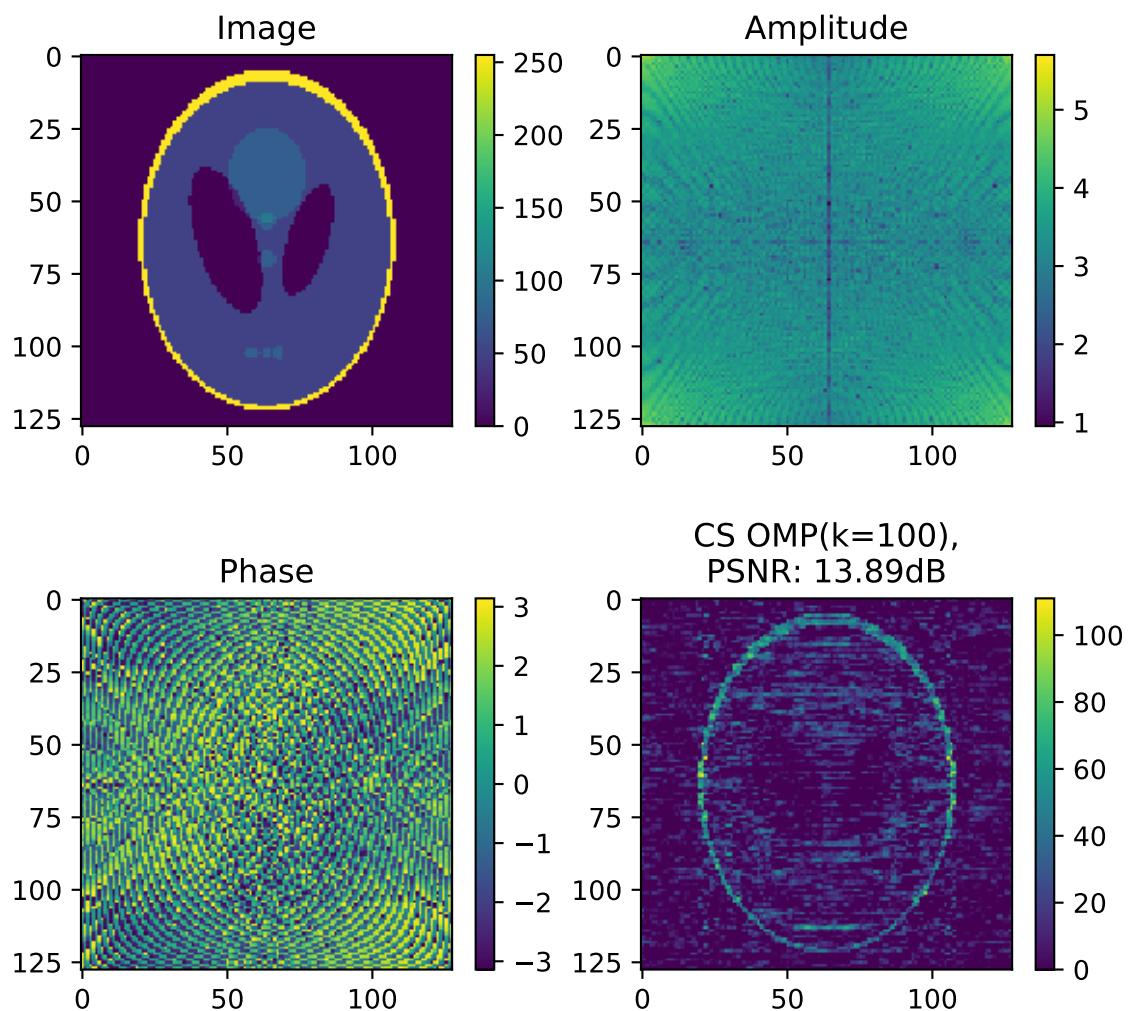


图 5.60: Recovery results ($\text{CR} = 4$) with sparse degree $k = 100$.

5.5.7 学习式压缩感知

5.5.7.1 学习式压缩感知

学习式压缩感知 (*Learned Compressive Sensing*, LCS)

5.5.8 深度学习与压缩感知

5.5.8.1 基于 GAN 的压缩感知

深度压缩感知 (Deep Compressive Sensing, DCS) [2]

5.5.8.2 基于 ISTA 的压缩感知网络

深度压缩感知 (Deep Compressive Sensing, DCS) [2]

5.5.8.3 基于 ADMM 的压缩感知网络

深度压缩感知 (Deep Compressive Sensing, DCS) [2]

5.5.8.4 基于 SALSA 的压缩感知网络

深度压缩感知 (Deep Compressive Sensing, DCS) [2]

5.5.8.5 深度压缩感知

深度压缩感知 (Deep Compressive Sensing, DCS) [2]

5.5.9 参考文献

5.5.10 名词术语

Compressive Sensing 压缩感知 ([Compressive Sensing](http://en.volupedia.org/wiki/Compressed_sensing) (http://en.volupedia.org/wiki/Compressed_sensing) , CS)

Discrete Cosine Transform 离散余弦变换 ([Discrete Cosine Transform](http://en.volupedia.org/wiki/Discrete_cosine_transform) (http://en.volupedia.org/wiki/Discrete_cosine_transform))

Learned Compressive Sensing 可学习压缩感知 (Learned Compressive Sensing, LCS)

Linear Compressive Sensing 线性压缩感知 (Learned Compressive Sensing, LCS)

Nonlinear Compressive Sensing 可学习压缩感知 (Learned Compressive Sensing, NCS)

Null Space Property 零空间特性 ([Null Space Property](http://en.volupedia.org/wiki/Null_Space_property) (http://en.volupedia.org/wiki/Null_Space_property))

Restricted Isometry Property 有 限 等 距 性 (Restricted Isometry Property (http://en.volupedia.org/wiki/Restricted_isometry_property))

Sparse Signal Processing 稀疏信号处理 (Sparse Signal Processing, SSP)

nal Decomposition 信号分解 (Signal Decomposition)

5.6 数字图像处理

5.6.1 图像压缩

5.6.2 图像增强

5.6.2.1 图像模糊

5.6.2.1.1 模糊算子

5.6.2.2 直方图均衡化



图 5.61: 直方图均衡化, 原图, OpenCV 彩色 RGB 均衡化, skimage 彩色 RGB 均衡化, 每个通道分别均衡化.

CHAPTER 6

第六卷人工智能

6.1 简介

该部分为进阶教程，通过该部分教程学习，你学会从代码注释 API 生成函数手册！

待完成 ······ 敬请期待 ······

[Outline of Artificial Intelligence](http://en.volupedia.org/wiki/Outline_of_artificial_intelligence) (http://en.volupedia.org/wiki/Outline_of_artificial_intelligence)

6.2 优化方法

6.2.1 简介

6.2.2 名词术语

Automatic Differentiation 自动微分 ([Automatic Differentiation](http://en.volupedia.org/wiki/Automatic_differentiation) (http://en.volupedia.org/wiki/Automatic_differentiation))

Numerical Differentiation 数值微分 ([Numerical Differentiation](http://en.volupedia.org/wiki/Numerical_differentiation) (http://en.volupedia.org/wiki/Numerical_differentiation))

Symbolic Differentiation 符号微分 ([Symbolic Differentiation](http://en.volupedia.org/wiki/Symbolic_differentiation) (http://en.volupedia.org/wiki/Symbolic_differentiation))

6.3 机器学习

6.3.1 简介

6.3.1.1 What is ?

6.3.1.1.1 dddddddd

6.3.1.1.1.1 ssssssssss

6.3.1.2 Why ?

6.3.1.2.1 dddddddd

6.3.1.2.1.1 ssssssssss

6.3.1.3 How?

6.3.1.3.1 materials

Outline Of Machine Learning (http://en.volupedia.org/wiki/Outline_of_machine_learning)

6.3.1.3.1.1 books

6.3.2 监督学习

6.3.2.1 监督学习实践 1 - 回归与优化

源自 [UFLDL Tutorial](http://ufldl.stanford.edu/tutorial/) (<http://ufldl.stanford.edu/tutorial/>) , 原始代码可以从这里 ([GitHub repository](https://github.com/amaas/stanford_dl_ex) (https://github.com/amaas/stanford_dl_ex)) 一次性下载。需要注意的是有些数据需要自己去下载, 比如, 在做 PCA 的练习时, 需要下载 MNIST 数据集, 可以到 [THE MNIST DATABASE](http://yann.lecun.com/exdb/mnist/) (<http://yann.lecun.com/exdb/mnist/>) 下载.

6.3.2.1.1 线性回归

6.3.2.1.1.1 线性回归预测房价

Exercise 1A : 线性回归预测房价, 只需补充目标函数及其梯度, 计算公式见原网页:

补充代码

`linear_regression.m`

```
%% YOUR CODE HERE %%
theta = theta';
%Compute the linear regression objective
for j = 1:m
    f = f + (theta*X(:,j) - y(j))^2;
end
f = f/2;

%Compute the gradient of the objective
for j = 1:m
    g = g + X(:,j)*(theta*X(:,j) - y(j));
end
```

实验结果

如原文所述, 训练和测试误差一般在 4.5 和 5 之间, 本人实验结果如 图 6.1 所示:

```
Optimization took 1.780584 seconds.
RMS training error: 4.731236
RMS testing error: 4.584099
```

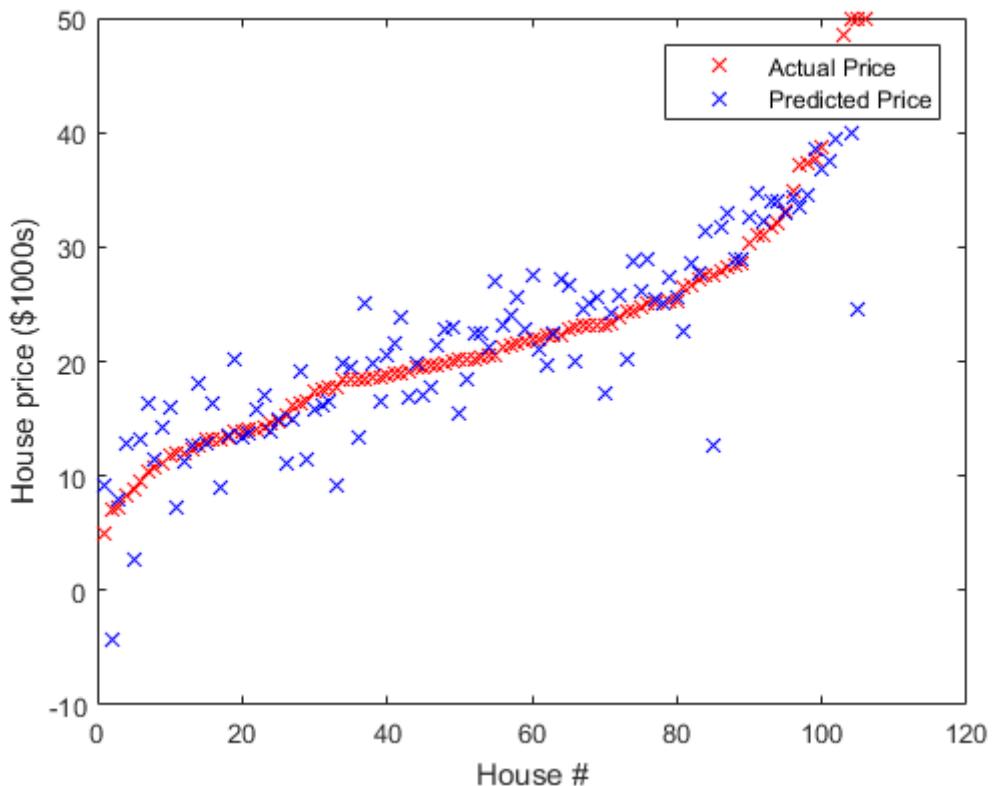


图 6.1: 线性回归——房价
线性回归——房价

6.3.2.1.2 Logistic Regression

[Logistic Regression](http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/) (<http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/>)

上面的线性回归有两个特点：

- 预测连续值(房价);
- 输出是输入的线性函数($y = h_{\theta}(x) = \theta^T x$);
- 代价函数为均方误差函数.

Logistic Regression:

- 预测离散值, 通常用于分类;
- 输出是输入的非线性函数(sigmoid 或 Logistic): $y = h_{\theta}(x) = \sigma(\theta^T x)$, $\sigma(z) = \frac{1}{1+exp(-z)}$;
- 代价函数取交叉熵(概率模型推), 最大似然, 见[CS229 Notes](http://cs229.stanford.edu/notes/cs229-notes1.pdf) (<http://cs229.stanford.edu/notes/cs229-notes1.pdf>) .

6.3.2.1.2.1 Logistic Regression 手写体分类

Logistic 分类, 用于手写体. 只需补充目标函数及其梯度, 计算公式见原网页, 推导见 [CS229 Notes](http://cs229.stanford.edu/notes/cs229-notes1.pdf) (<http://cs229.stanford.edu/notes/cs229-notes1.pdf>) :

补充代码与线性回归基本相同, 只是假设 $y = h_{\theta}(x) = \sigma(\theta^T x)$, $\sigma(z) = \frac{1}{1+exp(-z)}$ 为 sigmoid 函数, 不再是线性函数.

```
%% YOUR CODE HERE %%  
  
%Compute the linear regression objective and it's gradient  
for j = 1:m  
    coItem = sigmoid(theta'*X(:,j));  
    f = f - y(j)*log(coItem) - (1-y(j))*log(1-coItem);  
    g = g + X(:,j)*(coItem-y(j));  
end
```

实验结果

如原网页所述, 最终训练和测试精度都为 100%, 本人实验结果:

```
Optimization took 15.115248 seconds.  
Training accuracy: 100.0%  
Test accuracy: 100.0%
```

6.3.2.1.3 Vectorization

[Vectorization](http://ufldl.stanford.edu/tutorial/supervised/Vectorization/) (<http://ufldl.stanford.edu/tutorial/supervised/Vectorization/>)

补充代码

需要取消 `ex1a_linreg.m` 和 `ex1b_logreg.m` 文件中下面的注释：

`ex1a_linreg.m`

```
% theta = rand(n, 1);
% tic;
% theta = minFunc(@linear_regression_vec, theta, options, train.X, train.y);
% fprintf('Optimization took %f seconds.\n', toc);
```

`ex1b_logreg.m`

```
% theta = rand(n, 1)*0.001;
% tic;
% theta=minFunc(@logistic_regression_vec, theta, options, train.X, train.y);
% fprintf('Optimization took %f seconds.\n', toc);
```

`linear_regression_vec.m`

```
%%% YOUR CODE HERE %%%
f = (norm(theta'*X - y)) ^ 2 / 2;
g = X * (theta'*X - y)';
```

`logistic_regression_vec.m`

```
%%% YOUR CODE HERE %%%
coItem = sigmoid(theta'*X);
f = -log(coItem)*y' -log(1-coItem)*(1-y)';
g = X*(coItem-y)';
```

实验结果

速度快了好些，如下：

线性回归：

```
Optimization took 0.032485 seconds.(矢量化前约0.3s)
RMS training error: 4.023758
RMS testing error: 6.783703
```

Logistic 分类：

```
Optimization took 3.419164 seconds.(矢量化前约12s)
Training accuracy: 100.0%
Test accuracy: 100.0%
```

6.3.2.1.4 Debugging: Gradient Checking

[Debugging: Gradient Checking](http://ufldl.stanford.edu/tutorial/supervised/DebuggingGradientChecking/) (<http://ufldl.stanford.edu/tutorial/supervised/DebuggingGradientChecking/>)

补充代码下面是一次进行上述线性回归 Logistic 分类练习的梯度检验代码 grad_check_demo.m

```
%% for linear regression

% Load housing data from file.
data = load('housing.data');
data=data'; % put examples in columns

% Include a row of 1s as an additional intercept feature.
data = [ ones(1,size(data,2)); data ];

% Shuffle examples.
data = data(:, randperm(size(data,2)));

% Split into train and test sets
% The last row of 'data' is the median home price.
train.X = data(1:end-1,1:400);
train.y = data(end,1:400);

test.X = data(1:end-1,401:end);
test.y = data(end,401:end);

m=size(train.X,2);
n=size(train.X,1);

% Initialize the coefficient vector theta to random values.
theta0 = rand(n,1);

num_checks = 20;
% without vectorize
average_error = grad_check(@linear_regression, theta0, num_checks, train.X, train.y)
% vectorize
average_error = grad_check(@linear_regression_vec, theta0, num_checks, train.X, train.y)

%% for Logistic Classification
binary_digits = true;
[train,test] = ex1_load_mnist(binary_digits);

% Add row of 1s to the dataset to act as an intercept term.
train.X = [ones(1,size(train.X,2)); train.X];
test.X = [ones(1,size(test.X,2)); test.X];
```

(下页继续)

(续上页)

```
% Training set dimensions
m=size(train.X, 2);
n=size(train.X, 1);

% Train logistic regression classifier using minFunc
options = struct('MaxIter', 100);

% First, we initialize theta to some small random values.
theta0 = rand(n, 1)*0.001;

num_checks = 20;
% without vectorize
average_error = grad_check(@logistic_regression, theta0, num_checks, train.X, %
                           -train.y)
% vectorize
average_error = grad_check(@logistic_regression_vec, theta0, num_checks, train.X, %
                           -train.y)
```

实验结果

验证 20 次的平均误差分别为：

```
1.7030e-05(linear)
1.2627e-05(linear_vec)
6.0687e-06(Logistic)
8.1527e-06(Logistic_vec)
```

6.3.2.1.5 Softmax Regression

Softmax Regression (<http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/>)

多分类, Logistic 回归的推广.

6.3.3 无监督学习

6.3.3.1 无监督学习实践 1 - PCA 与白化

Preprocessing: PCA and Whitening

6.3.3.1.1 主成分分析 (PCA)

[PCA](http://deeplearning.stanford.edu/wiki/index.php/PCA) (<http://deeplearning.stanford.edu/wiki/index.php/PCA>)

6.3.3.1.2 白化 (Whitening)

[Whitening](http://deeplearning.stanford.edu/wiki/index.php/Whitening) (<http://deeplearning.stanford.edu/wiki/index.php/Whitening>)

白化的目的就是降低输入的冗余性，即想通过白化过程使得输入：

1. 特征间的相关性较低; (PCA)
2. 所有特征具有相同的方差. ($\frac{1}{\sqrt{\lambda_i + \epsilon}}$ 缩放)

PCA 已经使得数据间的相关性降低，为使每个输入特征具有单位方差，用 $\frac{1}{\sqrt{\lambda_i + \epsilon}}$ 作为缩放因子（ ϵ 用于正则化，防止分母为零，导致数据上溢到无穷，同时具有平滑即低通滤波作用），缩放每个特征 $x_{rot,i}$ ，即 PCA 白化后的数据 $x_{PCAwhite}$ 满足：

$$x_{PCAwhite,i} = \frac{x_{rot,i}}{\sqrt{\lambda_i + \epsilon}}, i \in \{1, 2, 3, \dots, n\}$$

PCA 白化后的数据已经具有单位协方差，ZCA 白化是在 PCA 白化的基础上，左乘一个任意正交矩阵 R ($RR^T = R^T R = I$, 也可以是旋转或反射矩阵)，那么处理后的数据仍然具有单位协方差，在 ZCA 白化中，取 PCA 投影矩阵 $R = U$ ，则：

$$x_{ZCAwhite} = U x_{PCAwhite}$$

这种旋转使得 $x_{ZCAwhite}$ 尽可能的接近原始数据 x .

6.3.3.1.3 PCA/Whitening 计算步骤

[Implementing PCA/Whitening](http://deeplearning.stanford.edu/wiki/index.php/Implementing_PCA/Whitening) (http://deeplearning.stanford.edu/wiki/index.php/Implementing_PCA/Whitening)

1. 使数据均值为零;
2. 计算协方差矩阵;
3. 计算协方差矩阵的特征值和特征向量;
4. 使用特征向量组成的 **PCA 变换矩阵** U (旋转矩阵，按特征值大小降序且按列排) 旋转数据，得到 **PCA 变换**后的数据 x_{rot} ，取前 k 个特征向量组成旋转矩阵旋转数据，得到 **PCA 降维**后的数据 (假设输入 $x \in R^n, k < n$) ;
5. 使用 $\frac{1}{\sqrt{\lambda_i + \epsilon}}$ 对 PCA 旋转后的数据缩放，使其具有相同单位协方差，得到 **PCA 白化**后的数据 $x_{PCAwhite}$;
6. 对 PCA 白化后的数据，左乘 PCA 变换矩阵得到 **ZCA 白化**后的数据 $x_{ZCAwhite}$.

6.3.3.1.4 二维数据 PCA 实验

Exercise:PCA_in_2D (http://deeplearning.stanford.edu/wiki/index.php/Exercise:PCA_in_2D)

MATLAB 实现:

- 使数据均值为零: (提供的二维数据以具有相同均值 0, 此步省去)

```
avg = mean(x, 1);
x = x - repmat(x, 1);
```

- 计算协方差矩阵:

```
sigma = cov(x', 1); % MATLAB function, '1' normalize with n
sigma = x*x'/size(x, 2); % or you can use this code
```

原始数据协方差:

```
sigma =
0.0883    0.0733
0.0733    0.0890
```

- 计算协方差矩阵的特征值和特征向量:

```
[U, S, ~] = svd(sigma);
```

```
U =
-0.7055   -0.7087
-0.7087   0.7055
S =
0.1620      0
0      0.0154
```

- PCA 变换旋转数据 x 得到 x_{rot} :

```
xRot = U'*x;
```

- PCA 降维数据:

```
xRot = U(:,1:k) * x; % projecting to 1 dimension
xHat = U(:,1:k) * xRot; % projecting the xRot back
```

- PCA 白化:

```
xPCAWhite = diag(1./sqrt(diag(S)+epsilon)) * U'*x; % xPCAWhite_i = xRot_i /
sqrt(lambda_i)
sigmaPCAWhite = cov(xPCAWhite', 1); % computes the covariance of PCAWhite data
```

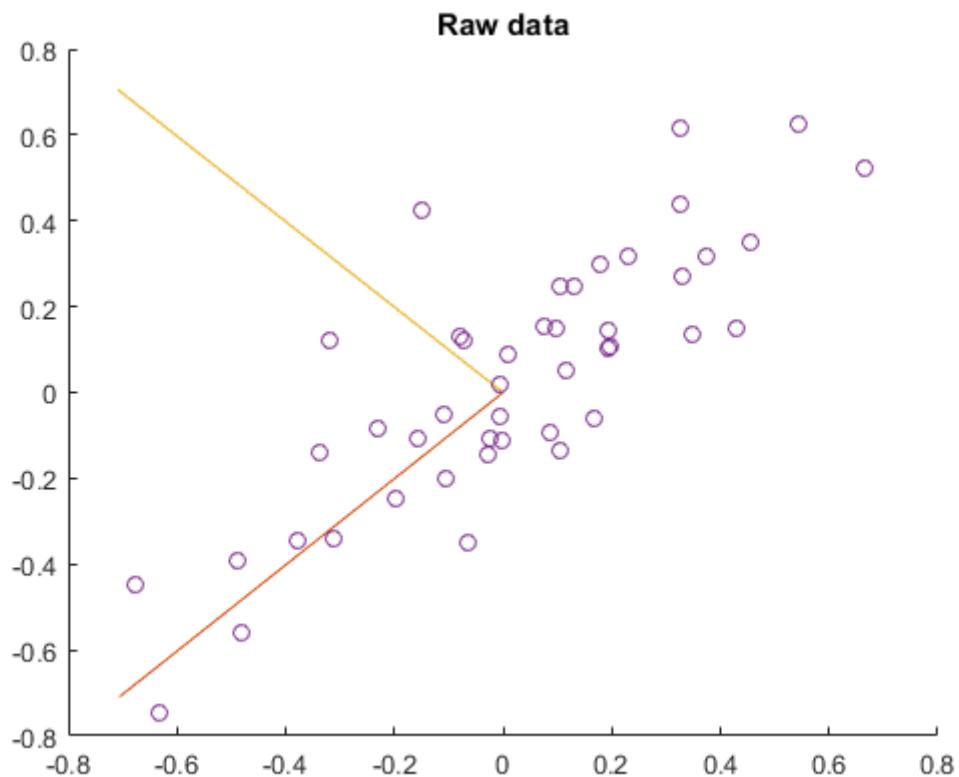


图 6.2: 原始二维数据及其协方差矩阵的特征向量
原始二维数据及其协方差矩阵的特征向量

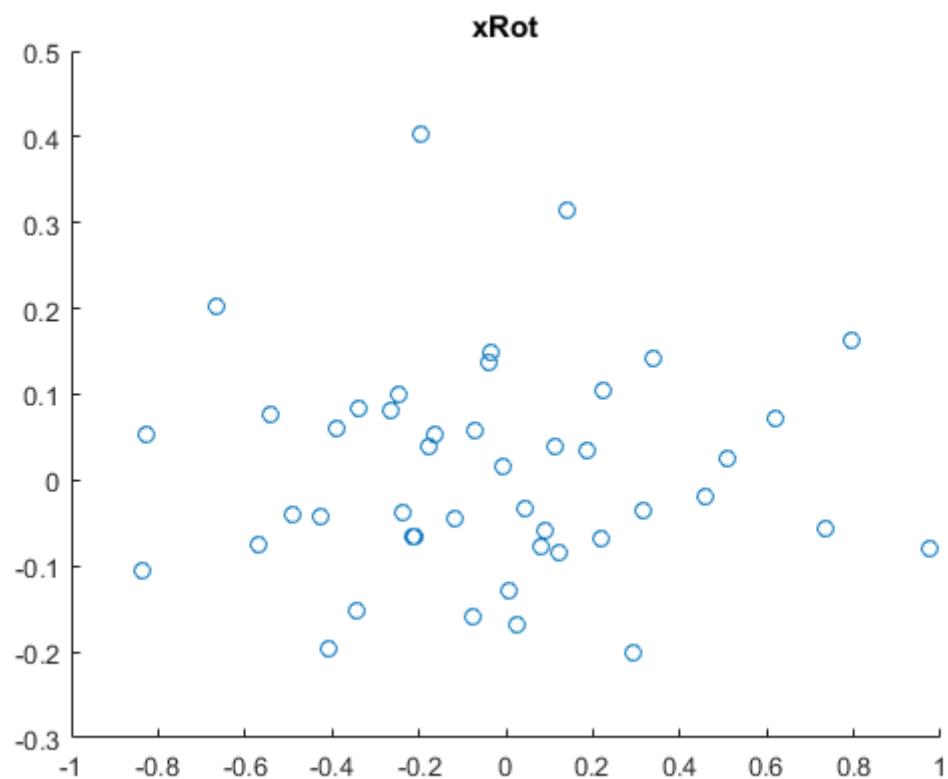


图 6.3: PCA 旋转后的数据
PCA 旋转后的数据

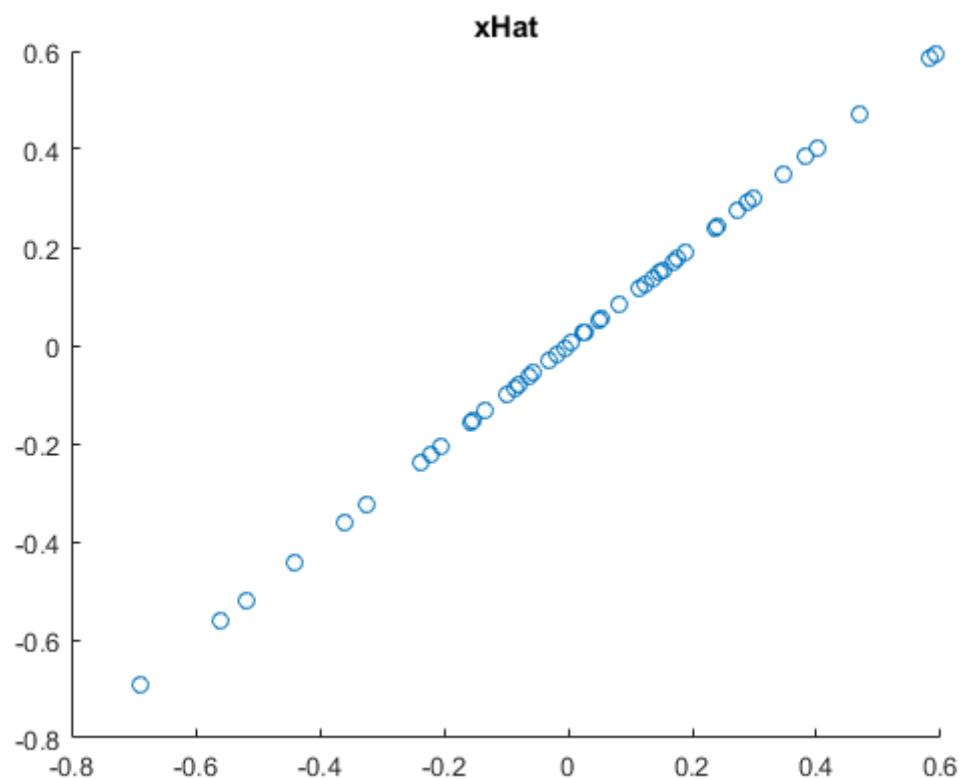


图 6.4: PCA 降维 - 旋转后映射回的数据
PCA 降维 - 旋转后映射回的数据

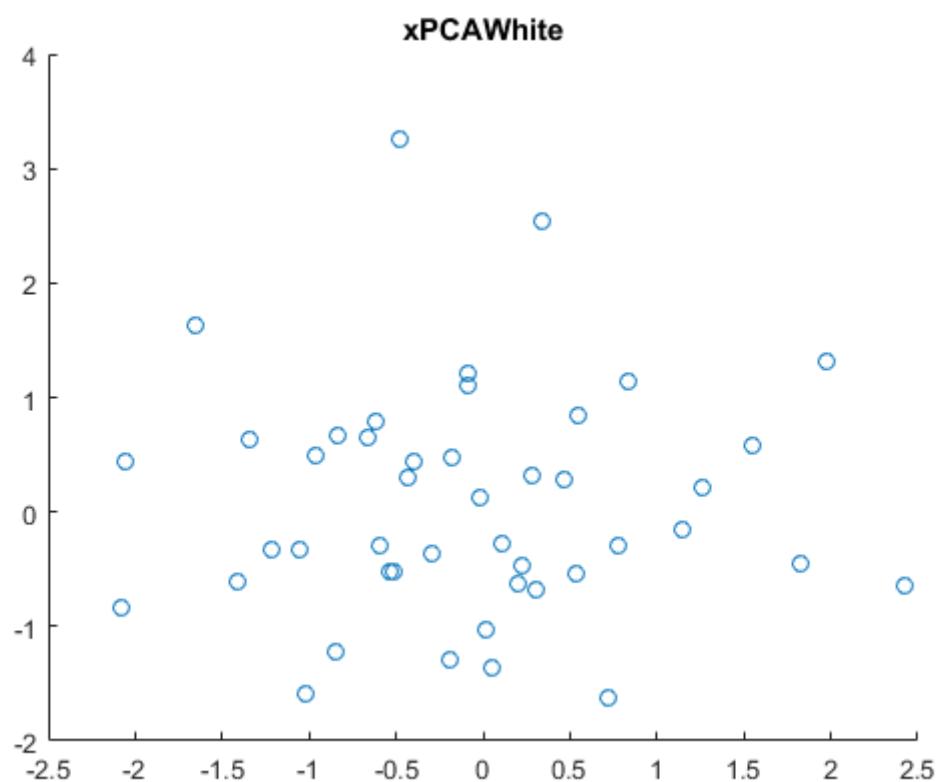


图 6.5: PCA 降维 - PCA 白化后的数据
PCA 降维 - PCA 白化后的数据

```
sigmaPCAWhite =
0.9921    0.0066
0.0066    0.9937
```

- ZCA 白化:

```
xZCAWhite = U*diag(1./sqrt(diag(S)+epsilon))*U'*x; % xZCAWhite_i = U*xPCAWhite

sigmaZCAWhite = cov(xZCAWhite', 1); % computes the covariance of ZCAWhite data
```

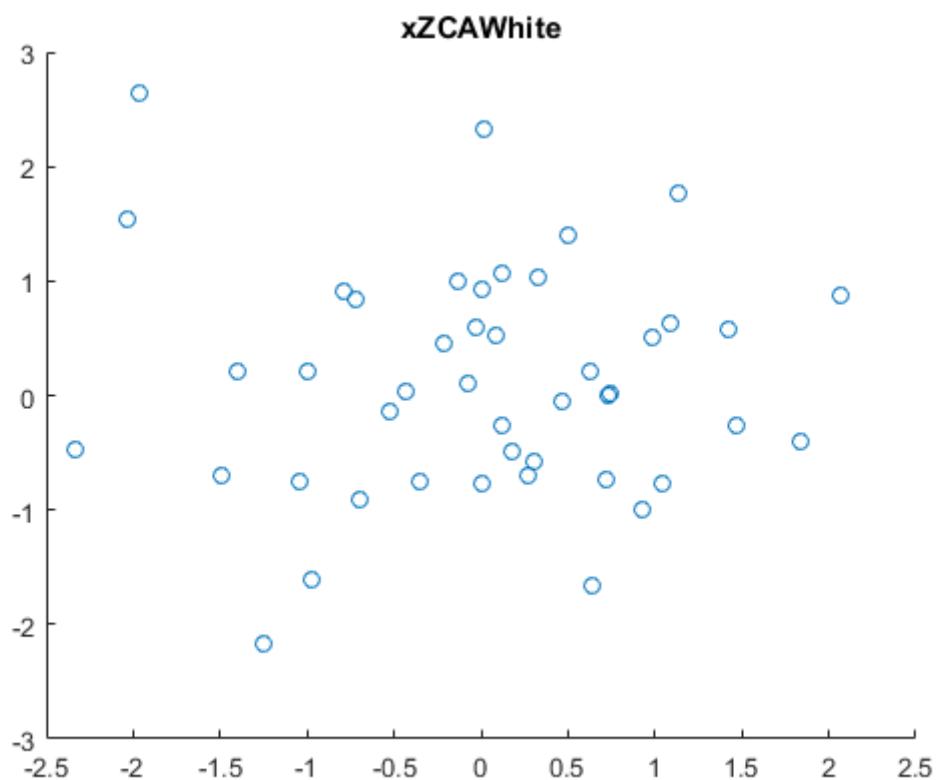


图 6.6: ZCA 降维 - ZCA 白化后的数据
ZCA 降维 - ZCA 白化后的数据

```
sigmaZCAWhite =
0.9996   -0.0008
-0.0008   0.9863
```

6.3.3.1.5 PCA 与白化图像数据实验

Exercise:PCA and Whitening (http://deeplearning.stanford.edu/wiki/index.php/Exercise:PCA_and_Whitening)

6.3.3.1.5.1 PCA 实验

当把 PCA 应用于图像时，有点疑惑的是，按照协方差的公式：

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])],$$

减去的均值应该是所有样本的均值。这样，对于图像块矩阵 $x_{n \times m}$ ，其中， n 为一个样本的特征维度即一个图像块拉成列向量的维度， m 为图像块的数目，求其特征维上的协方差时，均值应该这样求： $\bar{x} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$ ，而教程中是这样求的： $\bar{x}^{(i)} = \frac{1}{n} \sum_{j=1}^n x_j^{(i)}$ ，即每个块求均值。文中说我们对图像块的平均亮度值不感兴趣，所以可以减去这个值进行均值规整化。

So, we won't use variance normalization. The only normalization we need to perform then is mean normalization, to ensure that the features have a mean around zero. Depending on the application, very often we are not interested in how bright the overall input image is. For example, in object recognition tasks, the overall brightness of the image doesn't affect what objects there are in the image. More formally, we are not interested in the mean intensity value of an image patch; thus, we can subtract out this value, as a form of mean normalization.

6.3.3.1.5.2 PCA 白化实验

Exercise: PCA Whitening (<http://ufldl.stanford.edu/tutorial/unsupervised/ExercisePCAWhitening>)

一点说明：教程中叙述的与展示的实验结果图不一致：文中叙述的是 144×10000 的矩阵（与 Wiki 上的教程一致），实验数据集却是手写体 784×60000 ，这里也用手写体数据，可以到 [THE MNIST DATABASE](http://yann.lecun.com/exdb/mnist/) (<http://yann.lecun.com/exdb/mnist/>) 下载。

- 零均值化数据

这里处理的是手写体图像，所以求得是每个图像块（patch）的均值。

```
% avg = mean(x, 1);
% x = x - repmat(avg, size(x, 1), 1);
x = bsxfun(@minus, x, mean(x, 1)); % or this code
```

- 旋转数据：

```
sigma = x*x'/m; % n-by-n
[u, s, ~] = svd(sigma);
xRot = u'*x; % n-by-m
```

- 验证 PCA 实现的正确性，即旋转数据的协方差仅对角元素非零：

```
covar = xRot*xRot'/m; % the mean of xRot is 0
```

- 寻找保留主成分数目：

Raw images	Raw images(zero-mean)
6 4 2 6 2 3 2 7 8 3 0 2 1 9 5 1 0	6 4 2 6 2 3 2 7 8 3 0 2 1 9 5 1 0
9 4 4 8 7 9 8 1 2 4 6 1 6 0 1 7 0	9 4 4 8 7 9 8 1 2 4 6 1 6 0 1 7 0
8 8 3 8 5 0 3 8 1 2 9 4 6 8 1 2 9	8 8 3 8 5 0 3 8 1 2 9 4 6 8 1 2 9
2 7 2 2 1 3 3 1 9 1 7 7 1 0 6 4 4	2 7 2 2 1 3 3 1 9 1 7 7 1 0 6 4 4
3 8 7 7 6 8 5 4 2 0 4 6 3 7 4 9 1	3 8 7 7 6 8 5 4 2 0 4 6 3 7 4 9 1
6 5 8 7 7 7 1 8 6 2 3 2 6 2 7 3 9	6 5 8 7 7 7 1 8 6 2 3 2 6 2 7 3 9
6 4 9 5 5 7 0 3 3 8 0 0 0 9 8 1 9	6 4 9 5 5 7 0 3 3 8 0 0 0 9 8 1 9
1 5 4 5 0 5 5 5 3 9 5 4 7 6 0 8 3	1 5 4 5 0 5 5 5 3 9 5 4 7 6 0 8 3
8 1 2 6 7 0 4 2 8 5 3 4 3 4 1 2 8	8 1 2 6 7 0 4 2 8 5 3 4 3 4 1 2 8
2 1 2 1 0 3 4 3 5 4 8 3 8 8 4 1 6	2 1 2 1 0 3 4 3 5 4 8 3 8 8 4 1 6
4 8 4 5 0 1 0 9 5 4 7 0 3 1 4 3 7	4 8 4 5 0 1 0 9 5 4 7 0 3 1 4 3 7
7 5 6 9 1 4 7 8 3 8 9 3 8	7 5 6 9 1 4 7 8 3 8 9 3 8

图 6.7: 手写体数据及零均值化数据示意
手写体数据及零均值化数据示意

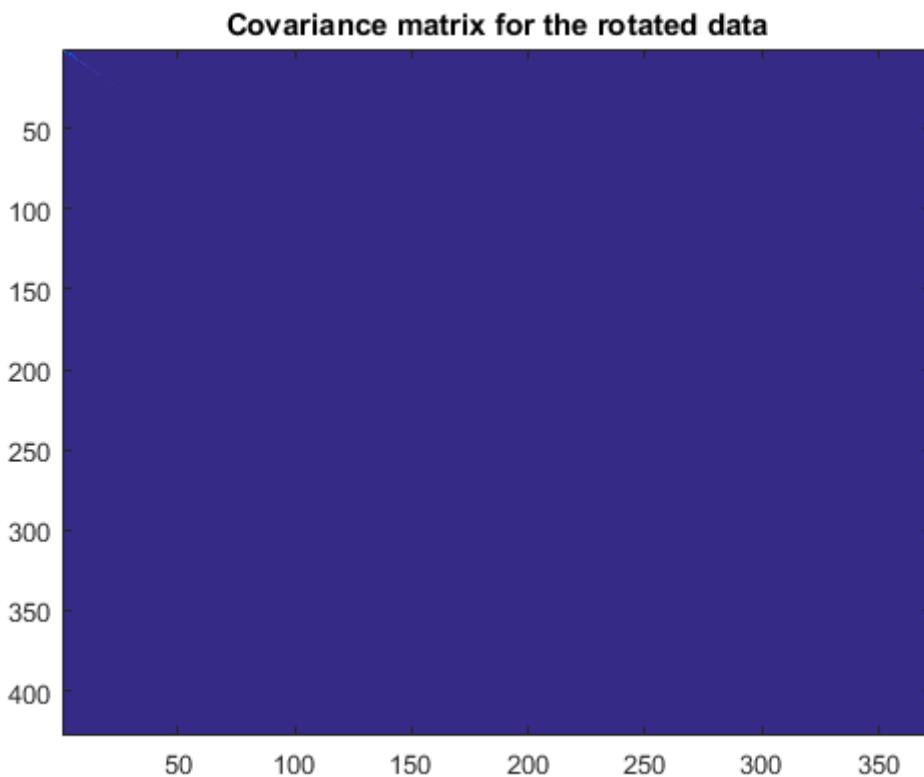


图 6.8: 手写体数据旋转后数据的协方差
手写体数据旋转后数据的协方差

保留方差百分比: 99%:

```
lambda = diag(s);
lambda_sum = sum(lambda);
lambda_sumk = 0;
perc = 0;
k = 0;
while perc < 0.99
    k = k + 1;
    lambda_sumk = lambda_sumk + lambda(k);
    perc = lambda_sumk / lambda_sum;
end
```

- 维数约简与重构:

```
xDimk = u(:, 1:k)' * x;
xHat = u(:, 1:k) * xDimk;
```

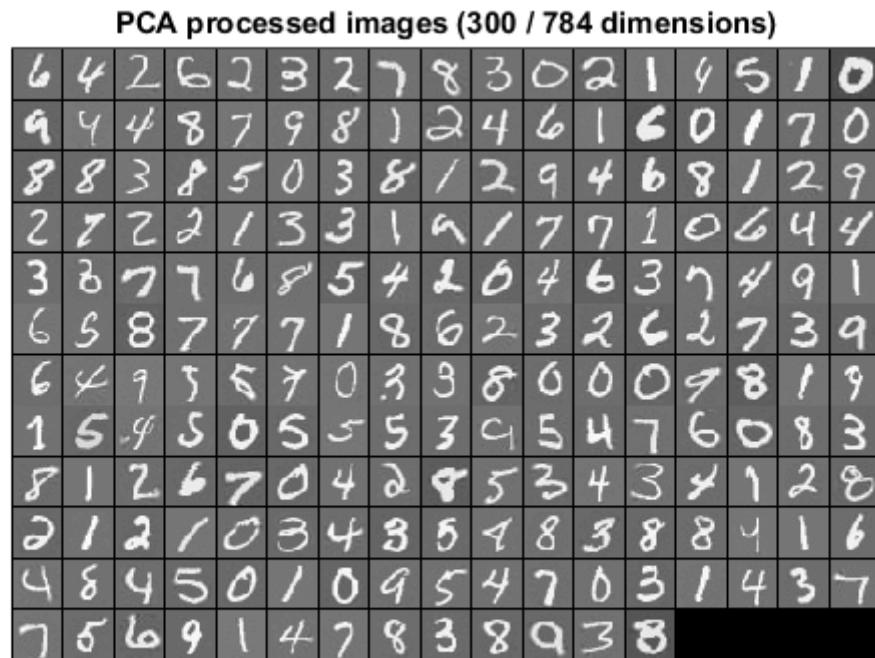


图 6.9: PCA 维数约简后重构的图像

PCA 维数约简后重构的图像 (采用每个图像块的均值零均值化图像, 方差保留百分比: 99.01%, 784 -> 300)

如果零均值化过程, 采用所有图像块的均值, 则零均值化后的图像和 PCA 重构后的图像如下图所示:

- PCA 白化及检验:

通过观察白化后数据的协方差矩阵, 分别使用不同的值正则化和不使用正则化 (epsilon 近似为 0):

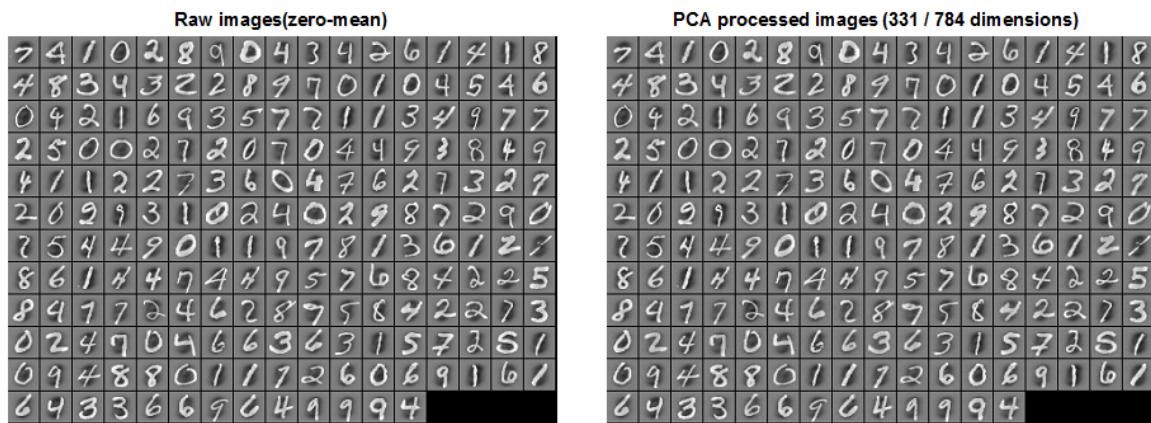


图 6.10: PCA 维数约简后重构的图像

PCA 维数约简后重构的图像 (采用所有图像块的均值零均值化图像, 方差保留百分比: 99.01%, 784 -> 300)

```
epsilon = 1e-1;
%%%% YOUR CODE HERE %%%
xPCAWHite = diag(1./sqrt(diag(s)+epsilon))*u'*x;
```

- ZCA 白化:

```
xZCAWhite = u*diag(1./sqrt(diag(s)+epsilon))*u'*x;
```

6.3.4 名词术语

Machine Learning 机器学习神经网络神经网络神经网络

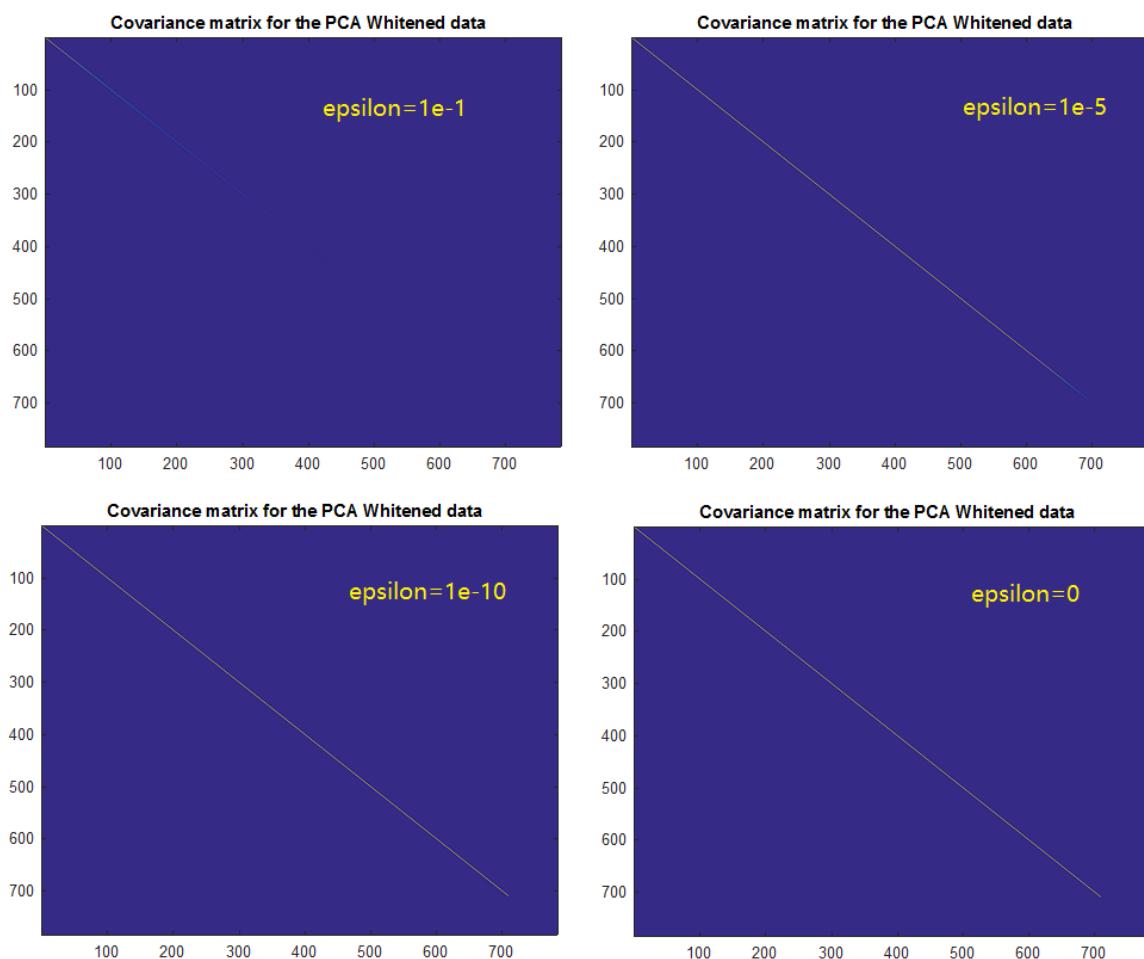


图 6.11: 白化后数据的协方差矩阵
白化后数据的协方差矩阵

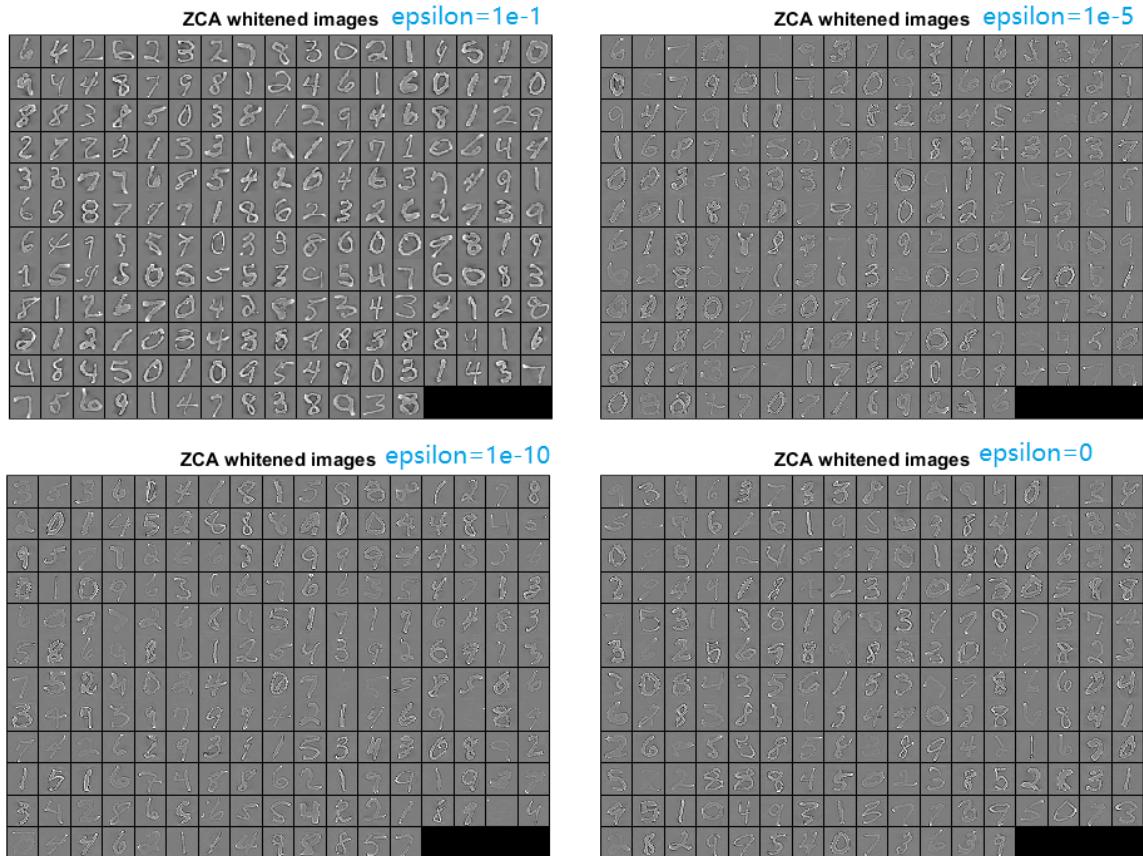


图 6.12: ZCA 维数约简后重构的图像

ZCA 维数约简后重构的图像

6.4 神经网络

6.4.1 简介

6.4.1.1 What is ?

6.4.1.1.1 dddddddd

6.4.1.1.1.1 ssssssssss

6.4.1.2 Why ?

6.4.1.2.1 dddddddd

6.4.1.2.1.1 ssssssssss

6.4.1.3 How?

6.4.1.3.1 materials

6.4.1.3.1.1 books

6.4.2 神经网络组件

6.4.2.1 激活函数单元

6.4.2.1.1 什么是激活函数

6.4.2.1.2 为什么要有激活函数

6.4.2.1.3 经典激活函数分类

- 参考文档

- [CS231N: Commonly used activation functions](http://cs231n.github.io/neural-networks-1/) (<http://cs231n.github.io/neural-networks-1/>)
- [激活函数 \(ReLU, Swish, Maxout\)](https://www.cnblogs.com/makefile/p/activation-function.html) (<https://www.cnblogs.com/makefile/p/activation-function.html>)

思维导图补充

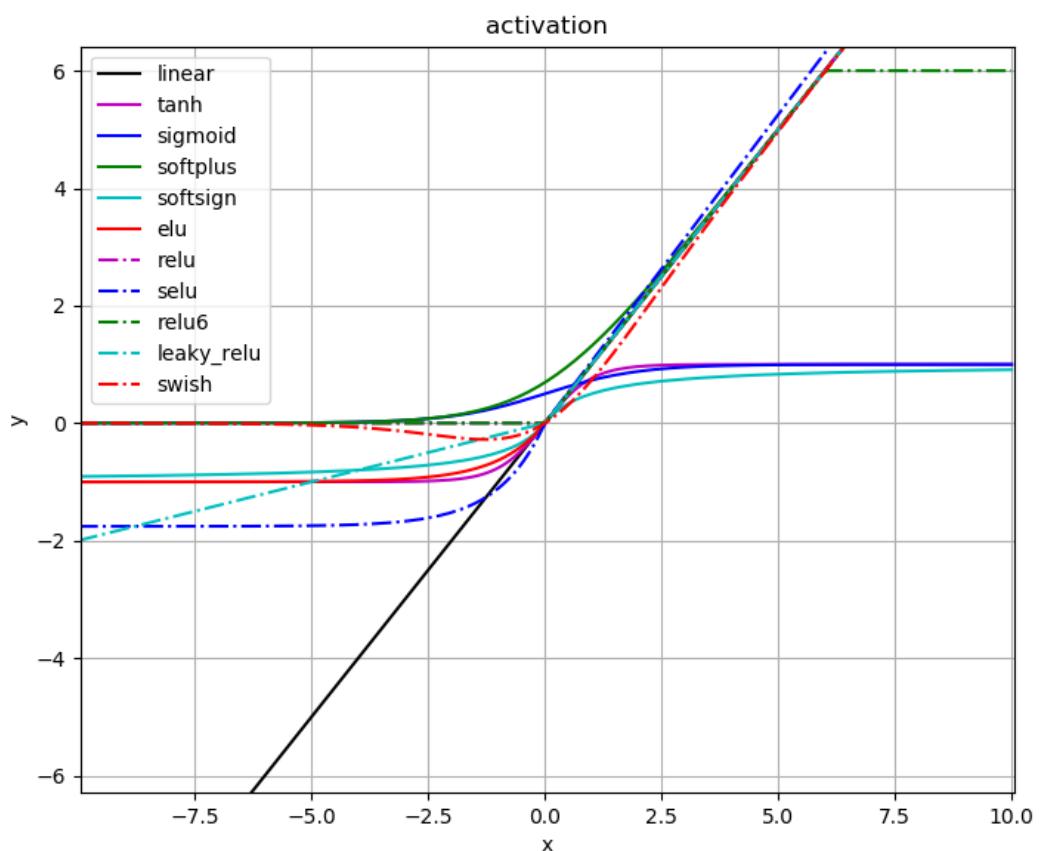


图 6.13: Activations 函数图像

Activations 函数图像

6.4.2.1.3.1 恒等函数

- 函数表达式: $y = x$
- 函数特性: 线性

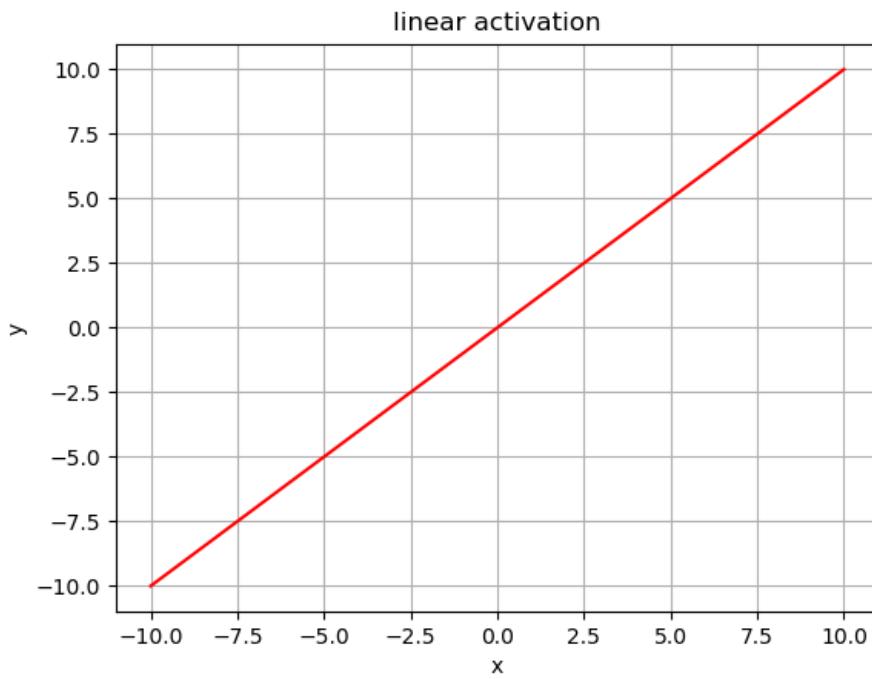


图 6.14: 恒等函数图像
恒等函数图像, 线性函数.

6.4.2.1.3.2 tanh

- [tanh, sinh, cosh](http://en.volupedia.org/wiki/Hyperbolic_function#Tanh) (http://en.volupedia.org/wiki/Hyperbolic_function#Tanh)
- 函数表达式: $y = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- 函数特性: 非线性, 存在梯度弥散

6.4.2.1.3.3 Sigmoid

- [sigmoid](http://en.volupedia.org/wiki/Sigmoid_function) (http://en.volupedia.org/wiki/Sigmoid_function)
- 函数表达式: $y = \frac{e^x}{e^x+1}$
- 函数特性: 非线性, 存在梯度弥散

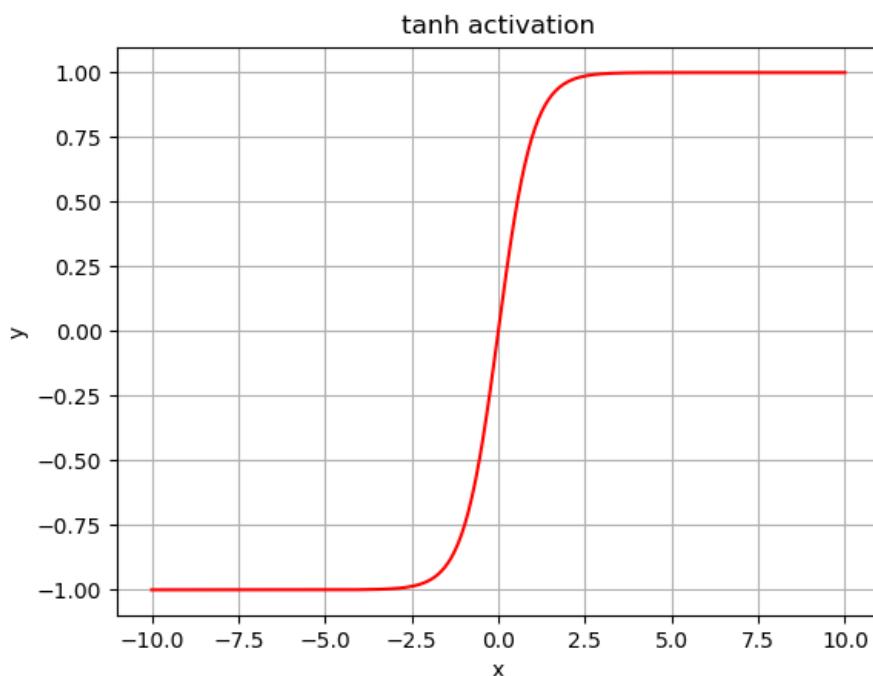


图 6.15: tanh 函数图像
tanh 函数图像, 非线性函数.

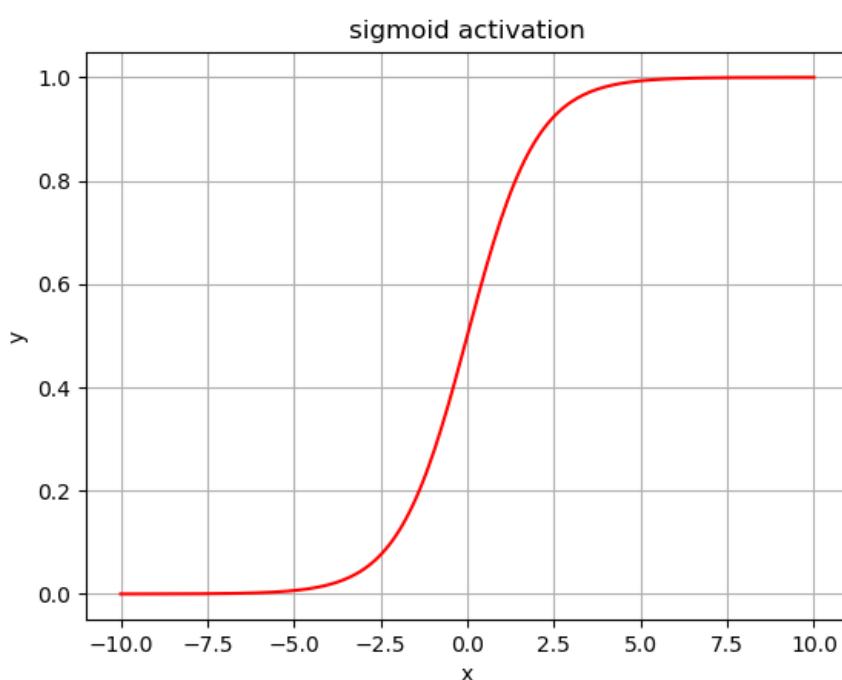


图 6.16: Sigmoid 函数图像
Sigmoid 函数图像, 非线性函数.

6.4.2.1.3.4 softplus

- 函数表达式: $\log(e^x + 1)$
- 函数特性: 非线性

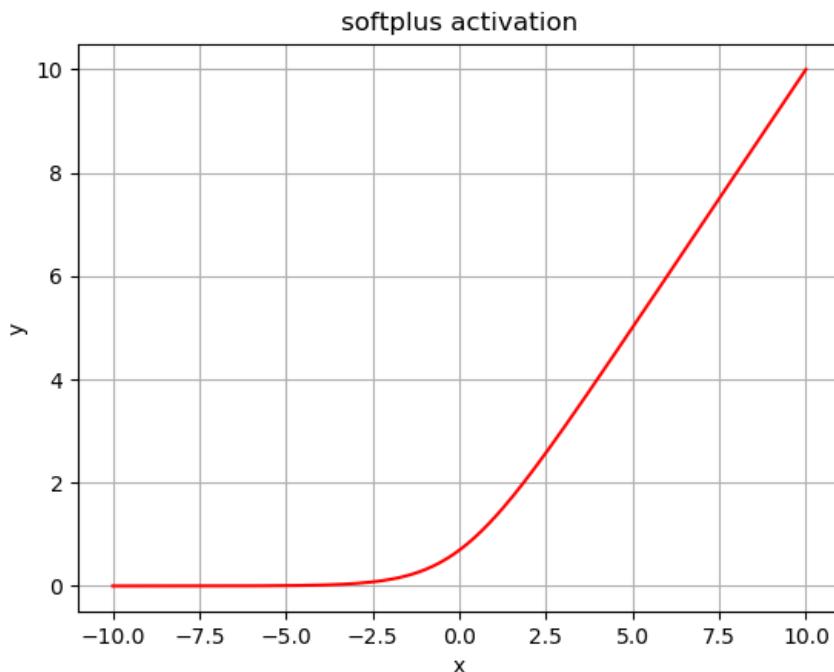


图 6.17: softplus 函数图像
softplus 函数图像, 非线性函数.

6.4.2.1.3.5 softsign

- 函数表达式: $\frac{x}{(\text{abs}(x)+1)}$
- 函数特性: 非线性

6.4.2.1.3.6 elu

- 函数表达式: $y = \begin{cases} x, & x \geq 0 \\ e^x - 1, & x < 0 \end{cases}$
- 函数特性: 非线性

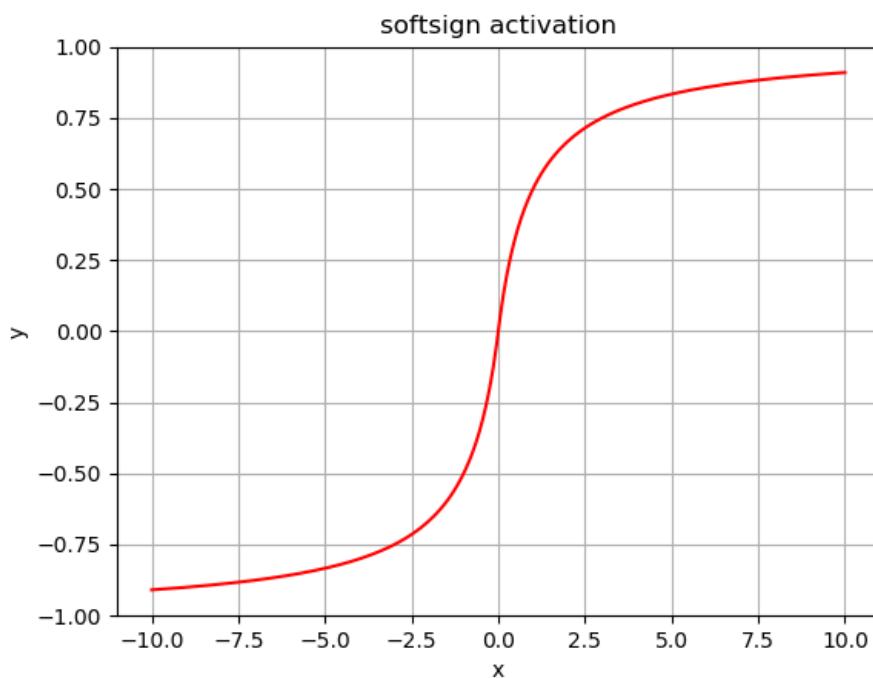


图 6.18: softsign 函数图像
softsign 函数图像, 非线性函数.

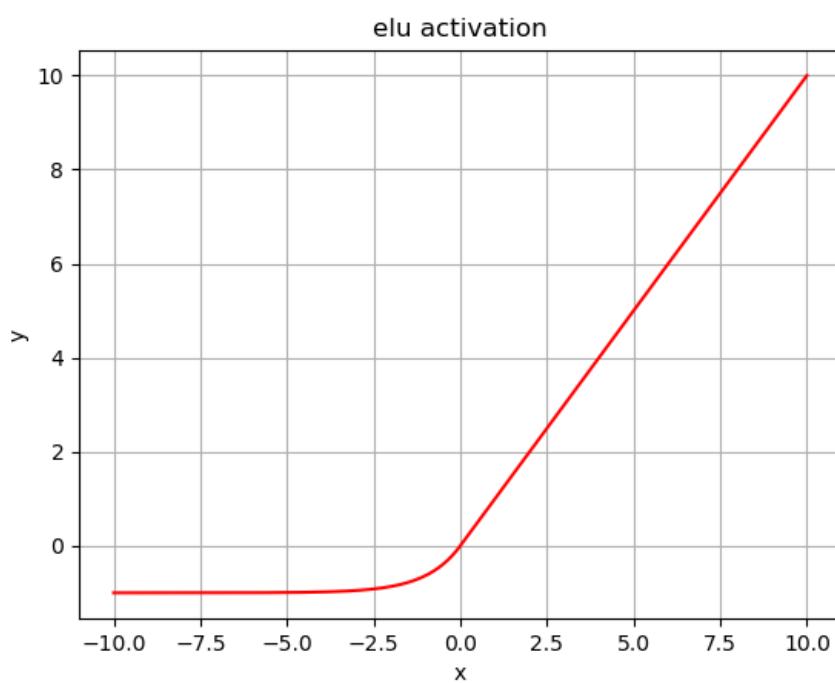


图 6.19: elu 函数图像
elu 函数图像, 非线性函数.

6.4.2.1.3.7 relu

- 函数表达式: $\max(x, 0)$
- 函数特性: 非线性 + 线性

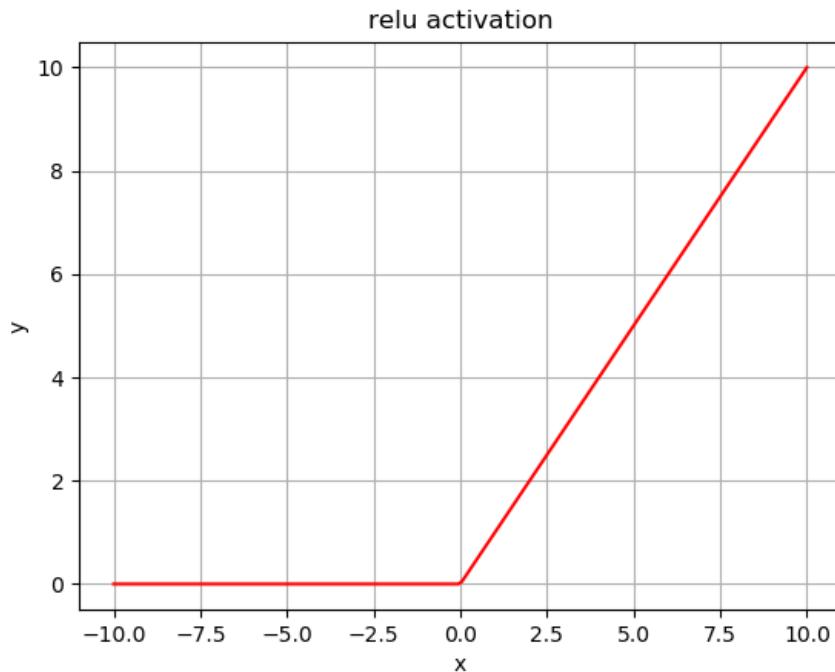


图 6.20: relu 函数图像
relu 函数图像, 非线性函数.

6.4.2.1.3.8 relu6

- Convolutional Deep Belief Networks on CIFAR-10. A. Krizhevsky (<http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf>)
- 函数表达式: $\min(\max(x, 0), 6)$
- 函数特性: 非线性 + 线性

6.4.2.1.3.9 leaky relu

- “Rectifier Nonlinearities Improve Neural Network Acoustic Models” AL Maas, AY Hannun, AY Ng - Proc. ICML, 2013 (http://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf)
- 函数表达式: $y = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$
- 函数特性: 非线性 + 线性

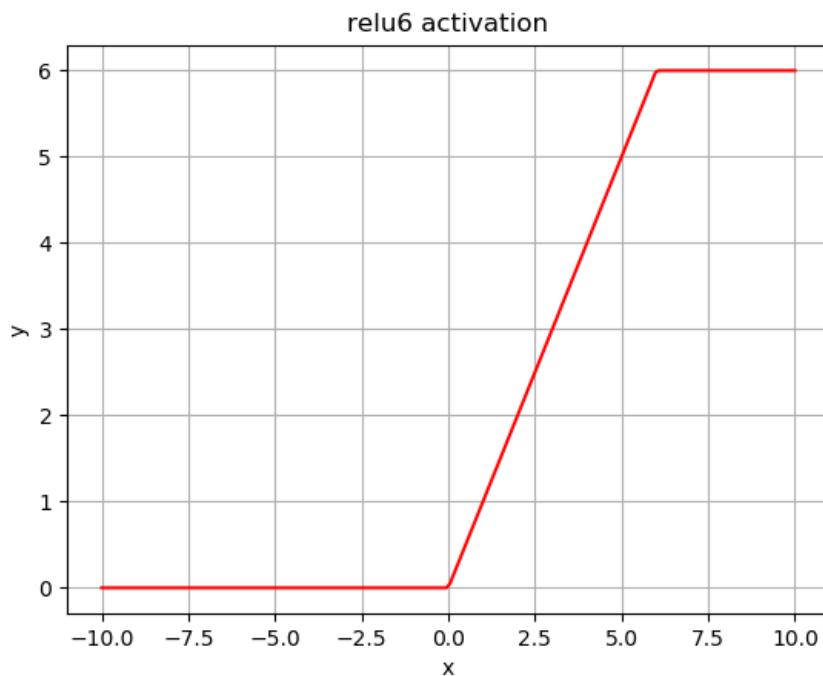


图 6.21: relu6 函数图像
relu6 函数图像, 非线性函数.

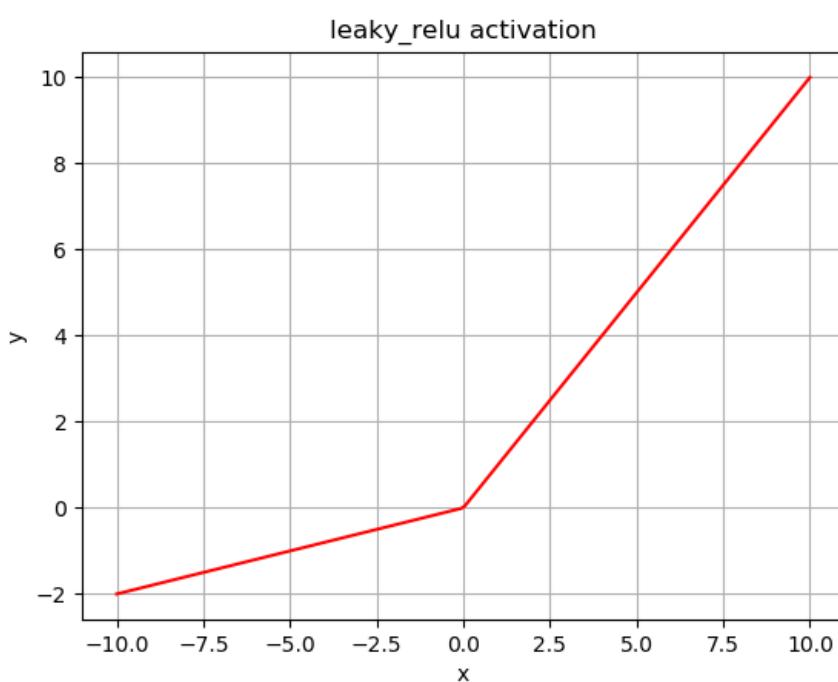


图 6.22: leaky relu 函数图像
leaky relu 函数图像, 非线性函数.

6.4.2.1.3.10 selu

- Self-Normalizing Neural Networks (<https://arxiv.org/abs/1706.02515>)
- 函数表达式: $y = \lambda \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$
- 函数特性: 非线性, 自归一化

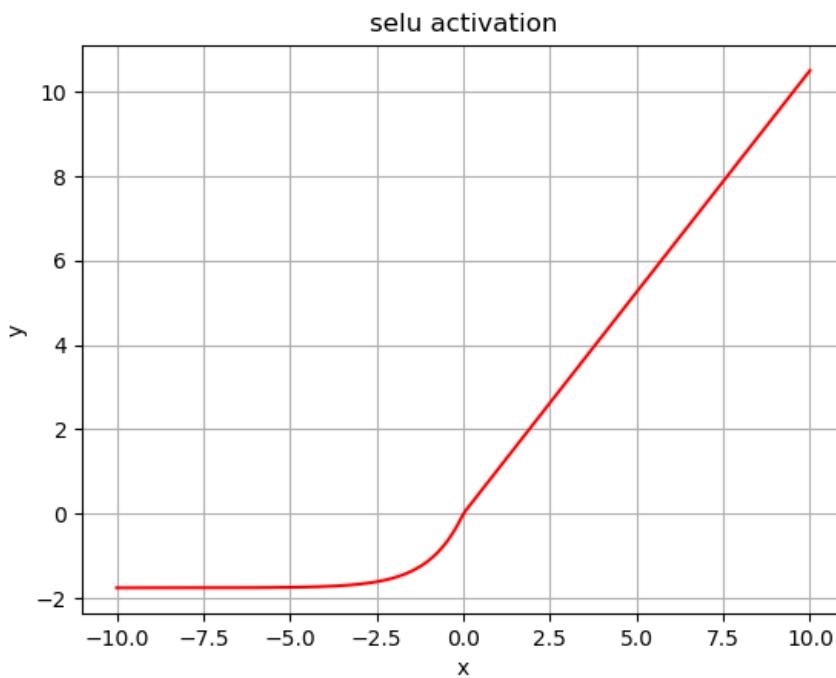


图 6.23: selu 函数图像
selu 函数图像, 非线性函数.

6.4.2.1.3.11 crelu

- Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. W. Shang, et al. (<https://arxiv.org/abs/1603.05201>)
- 函数表达式: $\frac{y=e^x}{(e^x+1)}$
- 函数特性: 非线性, 存在梯度弥散

图 6.24: crelu 函数图像
crelu 函数图像, 非线性函数.

6.4.2.1.3.12 Swish

- “Searching for Activation Functions” (Ramachandran et al. 2017) (<https://arxiv.org/abs/1710.05941>)
- 函数表达式: $y = x \cdot \text{sigmoid}(\beta x) = \frac{e^{(\beta x)}}{e^{(\beta x)} + 1} \cdot x$
- 函数特性: 非线性, 存在梯度弥散

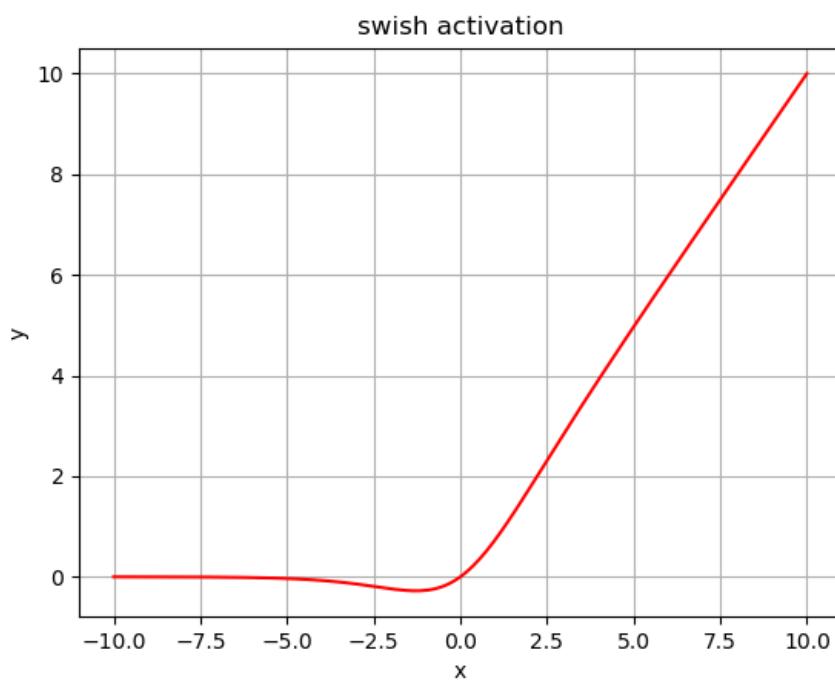


图 6.25: Swish 函数图像
Swish 函数图像, 非线性函数.

6.4.2.1.4 新颖激活函数

6.4.2.2 卷积单元

6.4.2.2.1 经典卷积运算

设有 N_i 个二维卷积输入 $\mathbf{I} \in \mathbb{R}^{N_i \times C_i \times H_i \times W_i}$, $C_k \times C_i$ 个二维卷积核 $\mathbf{K} \in \mathbb{R}^{C_k \times C_i \times H_k \times W_k}$, N_o 个卷积输出记为 $\mathbf{O} \in \mathbb{R}^{N_o \times C_o \times H_o \times W_o}$, 在经典卷积神经网络中, 有 $C_k = C_o$, $N_o = N_i$, \mathbf{K} 与 \mathbf{I} 间的二维卷积运算可以表示为

$$\begin{aligned}\mathbf{O}_{n_o, c_o, :, :} &= \sum_{c_i=0}^{C_i-1} \mathbf{I}_{n_o, c_i, :, :} * \mathbf{K}_{c_o, c_i, :, :} \\ &= \sum_{c_i=0}^{C_i-1} \mathbf{Z}_{n_o, c_o, c_i, :, :}\end{aligned}\quad (6.1)$$

其中, $*$ 表示经典二维卷积运算, 卷积核 $\mathbf{K}_{c_o, c_i, :, :}$ 与输入 $\mathbf{I}_{n_o, c_i, :, :}$ 的卷积结果记为 $\mathbf{Z}_{n_o, c_o, c_i, :, :} \in \mathbb{R}^{H_o \times W_o}$, 则

$$\mathbf{Z}_{n_o, c_o, c_i, h_o, w_o} = \sum_{h=0}^{H_k-1} \sum_{w=0}^{W_k-1} \mathbf{I}_{n_o, c_i, h_o+h-1, w_o+w-1} \cdot \mathbf{K}_{c_o, c_i, h, w}. \quad (6.2)$$

记卷积过程中, 高度与宽度维上填补 (padding) 大小为 $H_p \times W_p$, 卷积步长为 $H_s \times W_s$, 则卷积输出大小满足

$$\begin{aligned}H_o &= \left\lfloor \frac{\frac{H_i+2 \times H_p-H_k}{H_s}+1}{\frac{W_i+2 \times W_p-W_k}{W_s}+1} \right\rfloor \\ W_o &= \left\lfloor \frac{W_i+2 \times W_p-W_k}{W_s}+1 \right\rfloor\end{aligned}\quad (6.3)$$

警告: 卷积神经网络中的卷积操作, 实际上是相关操作, 因为在运算过程中, 未对卷积核进行翻转操作, 卷积与相关的关系参见 [卷积与相关 \(页 121\)](#) 小节.

如图 6.26 所示为二维卷积操作示意图.

[卷积与转置卷积](https://blog.csdn.net/LoseInVain/article/details/81098502) (<https://blog.csdn.net/LoseInVain/article/details/81098502>)

[Deconvolution and Checkerboard Artifacts](https://distill.pub/2016/deconv-checkerboard/) (<https://distill.pub/2016/deconv-checkerboard/>)

6.4.2.2.1.2 经典膨胀二维卷积运算

设有 N_i 个二维卷积输入 $\mathbf{I} \in \mathbb{R}^{N_i \times C_i \times H_i \times W_i}$, $C_k \times C_i$ 个二维卷积核 $\mathbf{K} \in \mathbb{R}^{C_k \times C_i \times H_k \times W_k}$, 高度与宽度维上填补 (padding) 大小为 $H_p \times W_p$, 膨胀 (dilation) 大小为 $H_d \times W_d$, 卷积步长为 $H_s \times W_s$, 在经典膨胀二维卷积神经网络中, 有 $C_k = C_o$, $N_o = N_i$, 则卷积后的输出为 $\mathbf{O} \in \mathbb{R}^{N_o \times C_o \times H_o \times W_o}$, 其中

$$\begin{aligned}H_o &= \left\lfloor \frac{\frac{H_i+2 \times H_p-H_d \times (H_k-1)-1}{H_s}+1}{\frac{W_i+2 \times W_p-W_d \times (W_k-1)-1}{W_s}+1} \right\rfloor \\ W_o &= \left\lfloor \frac{W_i+2 \times W_p-W_d \times (W_k-1)-1}{W_s}+1 \right\rfloor\end{aligned}\quad (6.4)$$

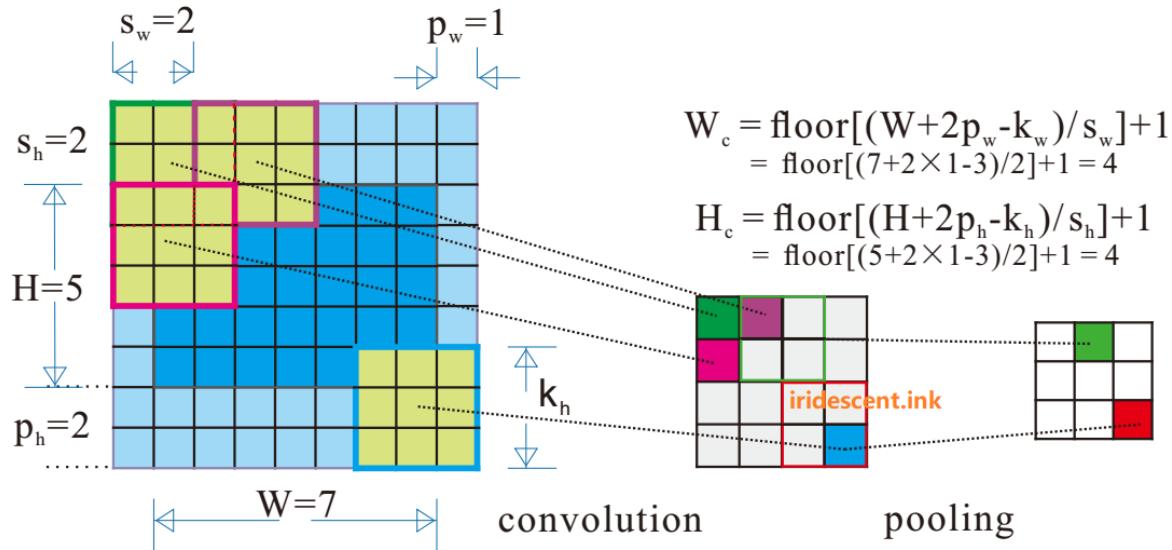


图 6.26: 卷积示意图
卷积示意图

对比式.6.3 和式.6.4, 可以发现当膨胀大小为 $H_d \times W_d = 1 \times 1$ 时, 膨胀卷积退化为经典卷积.

更多二维卷积示意图参见 [A technical report on convolution arithmetic in the context of deep learning](https://github.com/vdumoulin/conv_arithmetic) (https://github.com/vdumoulin/conv_arithmetic).

6.4.2.2.2 经典二维转置卷积运算

二维转置卷积是一种解卷积方法,

设有二维卷积核 $\mathbf{K} \in \mathbb{R}^{C_o \times H_k \times W_k}$, 二维卷积输入 $\mathbf{I} \in \mathbb{R}^{N_i \times C_i \times H_i \times W_i}$, 高度与宽度维上填补(padding)大小为 $H_p \times W_p$, 膨胀(dilation)大小为 $H_d \times W_d$, 卷积步长为 $H_s \times W_s$, 则卷积后填补(output-padding)大小为 $H_{op} \times W_{op}$, 则卷积后的输出为 $\mathbf{Y} \in \mathbb{R}^{N \times C_o \times H_o \times W_o}$, 其中

$$\begin{aligned} H_o &= (H_i - 1) \times H_s - 2 \times H_p + H_d \times (H_k - 1) + H_{op} + 1 \\ W_o &= (W_i - 1) \times W_s - 2 \times W_p + W_d \times (W_k - 1) + W_{op} + 1 \end{aligned} \quad (6.5)$$

6.4.2.2.3 新型卷积

6.4.2.2.3.1 平衡卷积

$$Z_{n_o, c_i, h_o, w_o} = \sum_{h=0}^{H_k-1} \sum_{w=0}^{W_k-1} [\mathbf{I}_{n_o, c_i, h_o+h-1, w_o+w-1} + \mathbf{K}_{c_o, c_i, h, w} - \mathbf{I}_{n_o, c_i, h_o+h-1, w_o+w-1} \cdot \mathbf{K}_{c_o, c_i, h, w}]. \quad (6.6)$$

6.4.2.2.4 实验分析

6.4.2.2.4.1 卷积与相关

6.4.2.2.4.2 实验说明

以二维卷积为例, 设有矩阵 a, b ,

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

则有卷积 $a * b$

$$a * b = \begin{bmatrix} 1 & 4 & 7 & 6 \\ 7 & 23 & 33 & 24 \\ 19 & 53 & 63 & 42 \\ 21 & 52 & 59 & 36 \end{bmatrix}$$

互相关 $a * b$

$$a * b = \begin{bmatrix} 4 & 11 & 18 & 9 \\ 18 & 37 & 47 & 21 \\ 36 & 67 & 77 & 33 \\ 14 & 23 & 26 & 9 \end{bmatrix}$$

6.4.2.2.4.3 实验结果

在 Matlab 环境中, 输入如下代码, 求解卷积 $a * b$ 与相关 $a * b$

代码 6.1: 2D convolution and cross-correlation Matlab

```

1 a = [1 2 3;4 5 6;7 8 9];
2 b = [1 2;3 4];
3
4 disp(a)
5 disp(b)
6
7 % convolution
8 disp(conv2(a, b))
9
10 % cross-correlation
11 disp(xcorr2(a, b))

```

MATLAB 中的 2D 卷积和相关结果为

```

1      2      3
4      5      6
7      8      9

1      2
3      4

1      4      7      6
7      23     33     24
19     53     63     42
21     52     59     36

4      11     18     9
18     37     47     21
36     67     77     33
14     23     26     9

```

在 Python 环境中, 输入如下代码, 求解卷积 $a * b$

代码 6.2: 2D convolution in PyTorch

```

1 import torch as th
2
3 a = th.tensor([[1., 2., 3], [4., 5., 6], [7., 8., 9]])
4 b = th.tensor([[1., 2], [3, 4]])
5
6 a = a.unsqueeze(0)    # 1x3x3
7 a = a.unsqueeze(0)    # 1x1x3x3
8 b = b.unsqueeze(0)    # 1x2x2
9 b = b.unsqueeze(0)    # 1x1x2x2
10
11 print(a, a.size())
12 print(b, b.size())
13
14 c = th.conv2d(a, b, stride=1, padding=1)
15
16 print(c)

```

PyTorch 中的 2D 卷积结果为

```

tensor([[[[1., 2., 3.],
          [4., 5., 6.],
          [7., 8., 9.]]]]) torch.Size([1, 1, 3, 3])
tensor([[[[1., 2.],
          [3., 4.]]]]) torch.Size([1, 1, 2, 2])
tensor([[[[ 4., 11., 18.,  9.],
          [18., 37., 47., 21.],
          [36., 67., 77., 33.]]]])

```

(下页继续)

(续上页)

```
[14., 23., 26., 9.]]])
```

注解: 对比结果可以发现, PyTorch 中的 2D 卷积实际上是 2D 相关操作, 与此类似, Tensorflow 等深度神经网络框架中的卷积均为相关操作. 但这并不影响网络的性能, 这是因为卷积核是通过网络学习的, 通过学习得到的卷积核可以看作是翻转后卷积核.

6.4.2.3 池化单元

6.4.2.3.1 经典二维池化运算

与经典二维卷积一样, 池化后的特征图大小为

$$\begin{aligned} H_o &= \left\lfloor \frac{H_i + 2 \times P_h - K_h}{S_h} + 1 \right\rfloor \\ W_o &= \left\lfloor \frac{W_i + 2 \times P_w - K_w}{S_w} + 1 \right\rfloor \end{aligned} \quad (6.7)$$

6.4.2.3.2 膨胀二维池化运算

$$\begin{aligned} H_o &= \left\lfloor \frac{H_i + 2 \times P_h - D_h \times (K_h - 1) - 1}{S_h} + 1 \right\rfloor \\ W_o &= \left\lfloor \frac{W_i + 2 \times P_w - D_w \times (K_w - 1) - 1}{S_w} + 1 \right\rfloor \end{aligned} \quad (6.8)$$

6.4.2.3.3 金字塔池化

6.4.2.3.3.1 空间金字塔池化

空间金字塔池化 (Spatial Pyramid Pooling, SPP) 由何凯明等人于 2015 年提出 [1], 如 图 6.27 所示, 对于任意大小的特征图输出, 通过使用不同池化窗口和步长, 对特征图进行池化, 将特征图划分成具有不同尺度的固定大小特征图, 特征图拉成列向量并连接, 得到固定的输出维度, 实现了多尺度网络的训练以及识别, 进而提升了图像分类和目标检测的精度.

6.4.2.3.3.2 金字塔场景池化

赵等人 [2] 于 2016 年提出一种用于语义分割的金字塔场景分析网络 (Pyramid Scene Parsing Network, PSP-Net), 其中采用 图 6.28 所示池化结构, 与 SPP 类似, 通过使用不同池化窗口和步长, 对特征图进行池化, 可以得到不同等级池化特征 (图中对应全局池化, 2×2 , 3×3 , 6×6), 与 SPP 不同的是, 空间金字塔池化后, 采用 $1 \times 1 \times N$ 的卷积核来调整通道数, 并使用上采样操作将不同等级的特征图上采样到与原始卷积输出特征图一致大小.

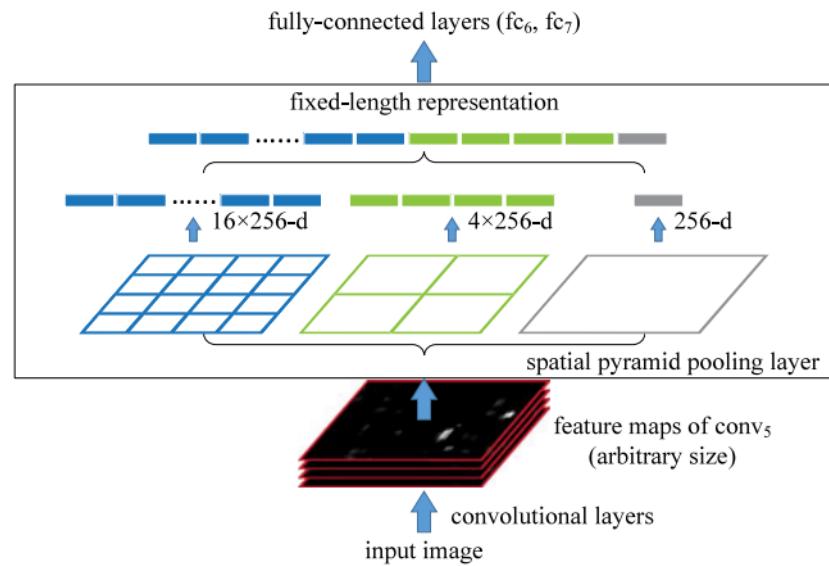


Fig. 3. A network structure with a *spatial pyramid pooling layer*. Here 256 is the filter number of the conv_5 layer, and conv_5 is the last convolutional layer.

图 6.27: 空间金字塔池化

空间金字塔池化

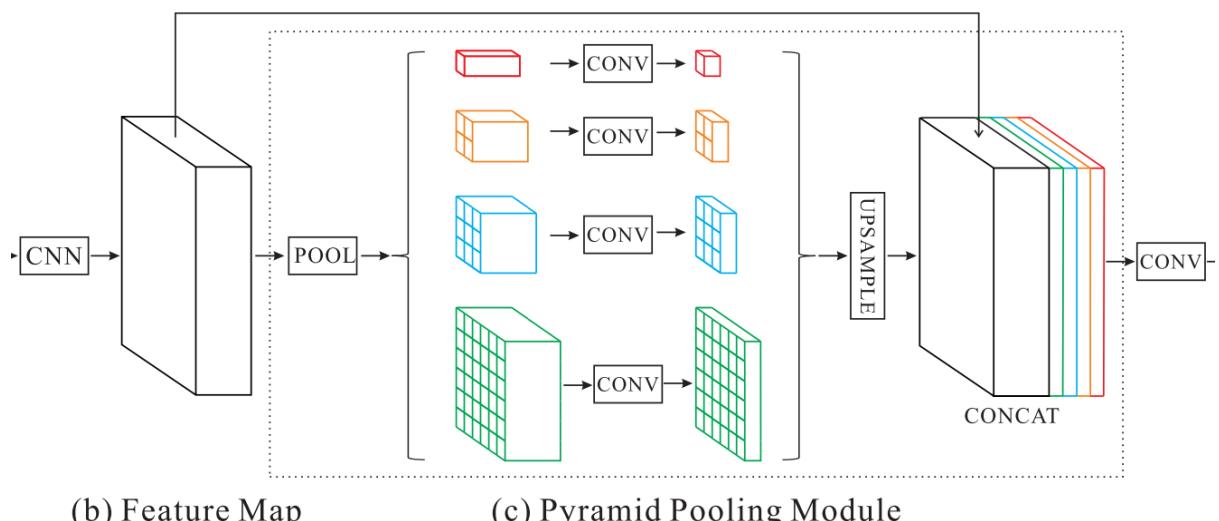


图 6.28: 金字塔池化
金字塔池化

6.4.2.3.3.3 扩张空间金字塔池化

扩张空间金字塔池化 (Atrous Spatial Pyramid Pooling, ASPP) 由陈等人 [3][4][5] 提出, 如 图 6.29 所示, 通过不同大小的卷积核, 膨胀因子和填补大小, 输出相同大小的卷积特征图, 将这些不同尺度的特征拼接再经过 1×1 卷积等到 ASPP 池化输出.

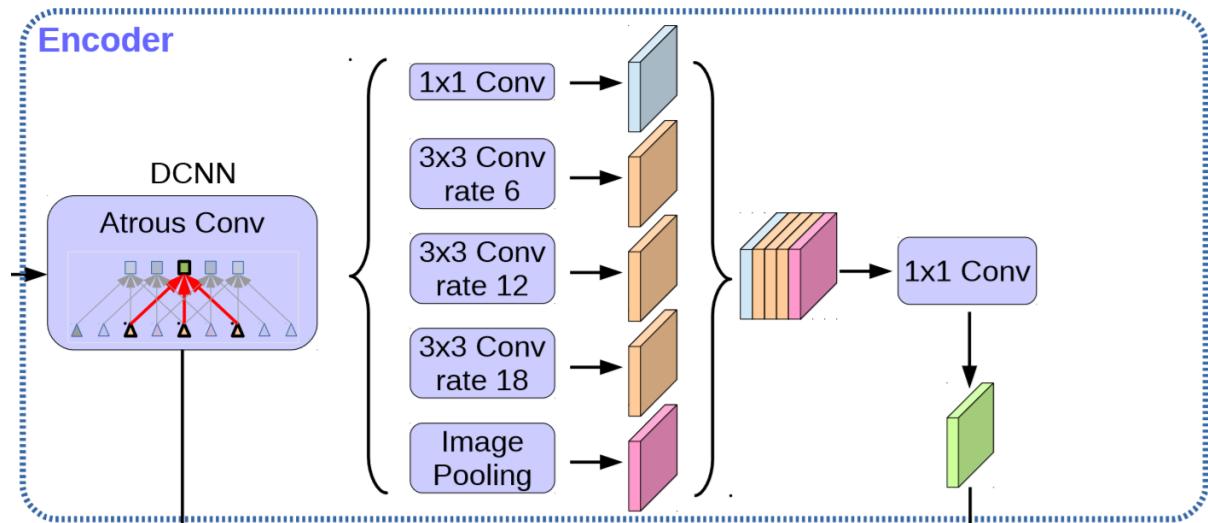


图 6.29: 扩张空间金字塔池化
扩张空间金字塔池化

6.4.2.4 正则化单元

6.4.2.5 跳跃连接

跳跃连接 (Skip Connection)

6.4.2.5.1 前向跳跃连接

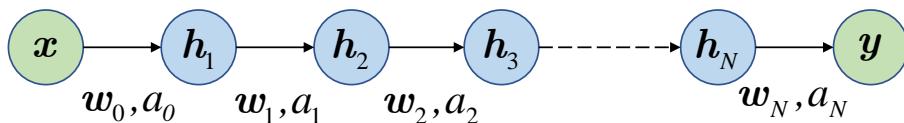
6.4.2.5.1.1 经典前向跳跃连接

6.4.2.5.1.2 结构分析

如 图 6.30 所示, w_n, a_n 分别表示第 n 层的权重与激活函数, x 为输入, y 为神经网络输出. 图 6.30 (a) 所示为无跳跃连接的网络结构, 图 6.30 (b) 所示为前向跳跃连接结构示意图, 其中, 第 p 层的输出跳跃连接至第

q 层的输出, 加和后送入下一层. 网络的前向传播过程可以表示为式.6.9.

$$\begin{aligned}
 \mathbf{y} &= a_N(\mathbf{w}_N \mathbf{h}_N) \\
 &\vdots \\
 \mathbf{h}_{q+1} &= a_q(\mathbf{w}_q \mathbf{s}_q) \\
 \mathbf{s}_q &= \mathbf{h}_p + \mathbf{h}_q \\
 &\vdots \\
 \mathbf{h}_{p+1} &= a_p(\mathbf{w}_p \mathbf{h}_p) \\
 &\vdots \\
 \mathbf{h}_2 &= a_1(\mathbf{w}_1 \mathbf{h}_1) \\
 &\vdots \\
 \mathbf{h}_1 &= a_0(\mathbf{w}_0 \mathbf{x})
 \end{aligned} \tag{6.9}$$



(a)

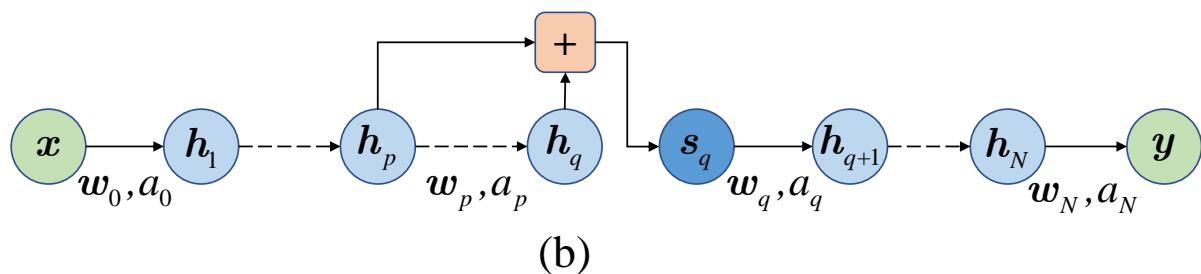


图 6.30: 前向跳跃连接示意图. 第 p 层的输出与第 q 层的输出加和后送入下一层.

6.4.2.5.1.3 梯度传播分析

利用链式求导法则, 图 6.30 (a) 所示传统神经网络的梯度传播过程可以表示为式.6.10

$$\begin{aligned}
 \frac{\partial L}{\partial \mathbf{w}_N} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{w}_N} \\
 \frac{\partial L}{\partial \mathbf{w}_{N-1}} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{w}_{N-1}} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_q} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{w}_q} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_p} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{h}_q} \times \cdots \times \frac{\partial \mathbf{h}_{p+1}}{\partial \mathbf{w}_p} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_1} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{w}_1} \\
 \frac{\partial L}{\partial \mathbf{w}_0} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial \mathbf{w}_0}
 \end{aligned} \tag{6.10}$$

同样地, 根据链式求导法则, 图 6.30 (b) 所示含跳跃连接的神经网络的梯度传播过程可以表示为式.6.11

$$\begin{aligned}
 \frac{\partial L}{\partial \mathbf{w}_N} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{w}_N} \\
 \frac{\partial L}{\partial \mathbf{w}_{N-1}} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{w}_{N-1}} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_q} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{w}_q} \\
 \frac{\partial L}{\partial \mathbf{w}_{q-1}} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_q} \times \frac{\partial \mathbf{h}_q}{\partial \mathbf{w}_{q-1}} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_p} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_{p+1}} \times \frac{\partial \mathbf{h}_{p+1}}{\partial \mathbf{w}_p} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_{p-1}} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_p} \times \frac{\partial \mathbf{h}_p}{\partial \mathbf{w}_{p-1}} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_1} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{w}_1} \\
 &\vdots \\
 \frac{\partial L}{\partial \mathbf{w}_0} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial \mathbf{w}_0}
 \end{aligned} \tag{6.11}$$

当 $k = p + 1, p + 2, \dots, q$ 时, $\frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_k} = \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_q} \times \frac{\partial \mathbf{h}_q}{\partial \mathbf{h}_{q-1}} \times \cdots \times \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}$.

当 $k = 1, 2, \dots, p$ 时, $\frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_k} = \frac{\partial \mathbf{s}_q}{\partial \mathbf{h}_p} \times \frac{\partial \mathbf{h}_p}{\partial \mathbf{h}_k} = \left(\frac{\partial \mathbf{h}_q}{\partial \mathbf{h}_p} + 1 \right) \times \frac{\partial \mathbf{h}_p}{\partial \mathbf{h}_k} = \left(\frac{\partial \mathbf{h}_q}{\partial \mathbf{h}_{q-1}} \times \cdots \times \frac{\partial \mathbf{h}_{p+1}}{\partial \mathbf{h}_p} + 1 \right) \times \frac{\partial \mathbf{h}_p}{\partial \mathbf{h}_{p-1}} \times \cdots \times \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}$.

注解: 由式.6.10 和式.6.11 可知, 与传统不含跳跃连接的神经网络相比, 从第 p 层输出跳跃连接至第 q 层输出的跳跃神经网络的梯度在第 $p+1$ 层到第 N 层保持不变, 在第 1 层到第 p 层的变大, 且增量为 $\frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_N} \times \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \times \cdots \times \frac{\partial \mathbf{h}_{q+1}}{\partial \mathbf{s}_q} \times \frac{\partial \mathbf{h}_p}{\partial \mathbf{h}_{p-1}} \times \cdots \times \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}$, ($k = 1, 2, \dots, p$).

6.4.2.5.2 后向跳跃连接

6.4.2.5.2.1 梯度传播

6.4.2.6 递归单元

6.4.2.7 残差单元

6.4.2.7.1 经典残差块分析

6.4.2.7.1.1 残差块结构

6.4.2.7.1.2 梯度传播分析

利用链式求导法则, 图 6.31 (a) 所示传统神经网络的梯度传播过程可以表示为式.6.12

$$\begin{aligned}\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial w_3} \\ \frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial w_2} \\ \frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial w_1} \\ \frac{\partial L}{\partial w_0} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial w_0}\end{aligned}\tag{6.12}$$

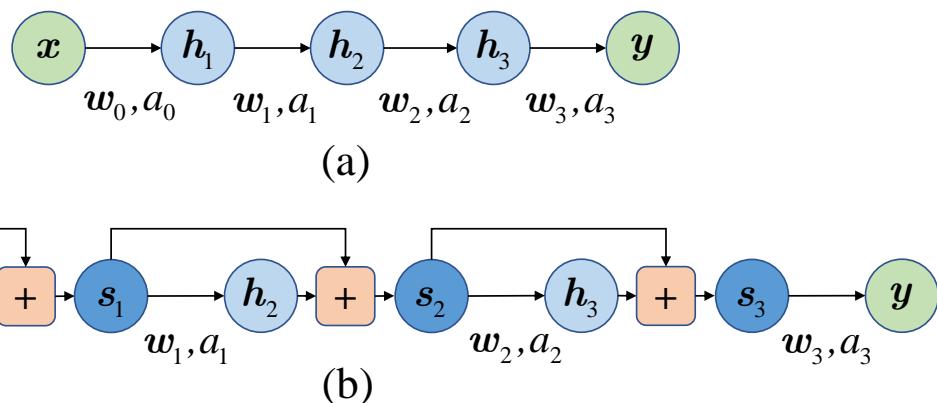


图 6.31: 经典神经网络与残差网络结构对比. (a) 含三个隐藏层的神经网络结构示意图; (b) 含三个残差块的神经网络示意图.

同样地, 根据链式求导法则, 图 6.31 (b) 所示残差神经网络的梯度传播过程可以表示为式.6.13

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}_3} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{w}_3} \\ \frac{\partial L}{\partial \mathbf{w}_2} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \frac{\partial \mathbf{s}_3}{\partial \mathbf{w}_2} \\ \frac{\partial L}{\partial \mathbf{w}_1} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \frac{\partial \mathbf{s}_3}{\partial \mathbf{s}_2} \times \frac{\partial \mathbf{s}_2}{\partial \mathbf{w}_1} \\ \frac{\partial L}{\partial \mathbf{w}_0} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \frac{\partial \mathbf{s}_3}{\partial \mathbf{s}_2} \times \frac{\partial \mathbf{s}_2}{\partial \mathbf{s}_1} \times \frac{\partial \mathbf{s}_1}{\partial \mathbf{w}_0}\end{aligned}\tag{6.13}$$

又 $\frac{\partial \mathbf{s}_n}{\partial \mathbf{w}_{n-1}} = \frac{\partial \mathbf{h}_n}{\partial \mathbf{w}_{n-1}}$, $\frac{\partial \mathbf{s}_n}{\partial \mathbf{s}_{n-1}} = \frac{\partial \mathbf{h}_n}{\partial \mathbf{s}_{n-1}} + 1$, 代入式.6.13 中得

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}_3} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{w}_3} \\ \frac{\partial L}{\partial \mathbf{w}_2} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{w}_2} \\ \frac{\partial L}{\partial \mathbf{w}_1} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \left(\frac{\partial \mathbf{h}_3}{\partial \mathbf{s}_2} + 1 \right) \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{w}_1} \\ \frac{\partial L}{\partial \mathbf{w}_0} &= \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{s}_3} \times \left(\frac{\partial \mathbf{h}_3}{\partial \mathbf{s}_2} + 1 \right) \times \left(\frac{\partial \mathbf{h}_2}{\partial \mathbf{s}_1} + 1 \right) \times \frac{\partial \mathbf{h}_1}{\partial \mathbf{w}_0}\end{aligned}\tag{6.14}$$

6.4.2.8 参考文献

6.4.3 神经网络常用损失函数

6.4.3.1 基于误差的损失函数

6.4.3.1.1 什么是误差

6.4.3.1.2 均方误差损失

6.4.3.1.3 绝对误差损失

6.4.3.1.4 光滑绝对误差损失

6.4.3.2 基于熵的损失函数

6.4.3.2.1 什么是熵

熵通常用来衡量信息量, 为香浓信息量的期望

6.4.3.2.1.1 离散变量的熵

$$H(p) = \text{E}_p[\log_2 \frac{1}{p}] = \sum_i p_i \log_2 \frac{1}{p_i} = - \sum_i p_i \log_2 p_i \quad (6.15)$$

6.4.3.2.2 交叉熵损失函数

6.4.3.2.2.1 离散变量的交叉熵

$$H(p, q) = \text{E}_p[\log_2 \frac{1}{q}] = \sum_i p_i \log_2 \frac{1}{q_i} = - \sum_i p_i \log_2 q_i \quad (6.16)$$

6.4.3.2.3 相对熵损失函数

相对熵 (relative entropy) 又称 KL 散度 (Kullback–Leibler divergence), 用于衡量两个分布的距离

6.4.3.2.3.1 离散变量的相对熵

$$D_{KL}(p||q) = \sum_i p_i \log_2 \frac{p_i}{q_i} = H(p, q) - H(p) \quad (6.17)$$

6.4.3.2.4 二值交叉熵损失函数

$$\begin{aligned} L(p, q) &= - \sum_{i \in \{1, 2\}} p_i \log_2 q_i \\ &= -p_1 \log_2 q_1 - p_2 \log_2 q_2 \\ &= -p_1 \log_2 q_1 - (1 - p_1) \log_2 (1 - q_1) \end{aligned} \quad (6.18)$$

6.4.4 简单神经网络

6.4.4.1 单层感知器

6.4.4.1.1 什么是感知器

6.4.4.1.2 单层感知器

6.4.4.1.3 实例

6.4.4.2 多层感知器

6.4.4.2.1 概念与内涵

6.4.4.2.2 实例

6.4.4.2.3 MNIST 手写体分类

实现代码, 参见文件 [demo_tf_mlp_mnist.py](#).

```
-----Start Demo-----
-----Demo loss= 1.601 Demo acc= 0.900
groundtruth labels: [3 2 0 7 5 1 8 9 4 6]
prediction labels: [3 2 0 7 4 1 8 9 4 6]
```

6.4.4.2.4 CIFAR10 物体分类

6.4.4.3 自编码神经网络

自编码神经网络 (Autoencoder Neural Network)

6.4.5 支撑矢量机

6.4.5.1 支撑矢量机

支撑矢量机 (*Support Vector Machine*) 也称支持向量机, Vladimir N. Vapnik and Alexey Ya. Chervonenkis 等人于 1963 年提出一种通过将核技巧应用于最大边缘超平面 (maximum-margin hyperplanes) 来创建非线性分类器的方法. 现今使用最多的标准的 SVM 是由 Corinna Cortes and Vapnik 等人于 1993 年提出并于 1995 年出版的 [5].

假设有训练样本集 $\mathbb{S} = \{(x_i, y_i)\}_{i=1}^N$, SVM 的优化问题为

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ & \text{subject to } y_i (\mathbf{w} \cdot \mathbf{z}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\ & \quad \xi_i \geq 0, \quad i = 1, \dots, N \end{aligned} \tag{6.19}$$

其中, \mathbf{w} 为模型参数, C 为平衡因子, ξ_i 为第 i 个样本对应误差, \mathbf{z}_i 为第 i 个样本 \mathbf{x}_i 在特征空间中的象 $\mathbf{z}_i = f(\mathbf{x}_i)$, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

SVM 中的参数 C 用于平衡最大边缘 (margin) 和错分的数量, 大的 C 使得训练的 SVM 具有窄的边缘和减少错分; 减小 C 使得 SVM 忽略更多训练样本, 分类边缘更宽.

- [libsvm](http://www.csie.ntu.edu.tw/~cjlin/libsvm/) (<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>) : a popular library of SVM learners
- [liblinear](http://www.csie.ntu.edu.tw/~cjlin/liblinear/) (<http://www.csie.ntu.edu.tw/~cjlin/liblinear/>) : a library for large linear classification including some SVMs
- [SVM light](http://svmlight.joachims.org/) (<http://svmlight.joachims.org/>) a collection of software tools for learning and classification using SVM
- [SVMJS live demo](http://cs.stanford.edu/people/karpathy/svmjs/demo/) (<http://cs.stanford.edu/people/karpathy/svmjs/demo/>) a GUI demo for JavaScript implementation of SVMs

6.4.5.2 模糊支撑矢量机

模糊支撑矢量机 (Fuzzy Support Vector Machine, FSVM) 通过在 SVM 中引入模糊性质, 使得对不干净输入更加鲁棒. [1], [2], [3].

6.4.5.2.1 FSVM 原理

现实世界中, 每个训练样本的重要性不一, 通常一些训练样本比另外一些更重要, 我们希望有意义的样本正确分类, 而不关心噪声样本是否被错分 [1]. 假设有训练样本集 $\mathbb{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, 为每一个训练样本分配一个隶属度 μ_i , 则样本集重新表示为 $\tilde{\mathbb{S}} = \{(\mathbf{x}_i, \mathbf{y}_i, \mu_i)\}_{i=1}^N$, SVM 优化问题式.6.19 变为

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \mu_i \xi_i \\ & \text{subject to } y_i (\mathbf{w} \cdot \mathbf{z}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\ & \quad \xi_i \geq 0, \quad i = 1, \dots, N \end{aligned} \tag{6.20}$$

其中, \mathbf{w} 为模型参数, C 为平衡因子, ξ_i 为第 i 个样本对应误差, \mathbf{z}_i 为第 i 个样本 \mathbf{x}_i 在特征空间中的象 $\mathbf{z}_i = f(\mathbf{x}_i)$, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

6.4.5.2.2 隶属函数选择

6.4.5.2.2.1 基于时间特性

对于序列学习, 样本的重要性与样本到达时间紧密相关, 特别是在实时信号处理中, 最新到达的样本比以往的样本更重要, 因而可以根据时间来确定隶属度 [1], 即

$$\mu_i = f_1(t_i) = \frac{1-\sigma}{t_N-t_1} t_i + \frac{t_N\sigma-t_1}{t_N-t_1},$$

其中, $t_1 < \dots < t_i < \dots < t_N$ 是训练样本到达时间序列.

6.4.5.2.2.2 基于类别中心

使用样本到类别中心的距离作为隶属度可以减少异常值的影响, 假设有 K 个类别, 用 $\bar{\mathbf{x}}_k$ 表示第 k 类的均值中心. 可定义如下隶属函数

$$\mu_i = 1 - |\bar{\mathbf{x}}_k - \mathbf{x}_i| / (r_k + \delta) \quad \text{if } y_i = k$$

其中, $k = 1, 2, \dots, K$, $\delta > 0$ 以避免 $\mu_i = 0$, r_k 为类别 k 的半径:

$$r_k = \max_{\{\mathbf{x}_i: y_i = k\}} |\bar{\mathbf{x}}_k - \mathbf{x}_i|.$$

6.4.5.2.2.3 基于训练误差

还可以根据训练误差来设置隶属度 [9], 这种方法需要先使用 SVM 训练, 将得到的误差 e 进行如下公式转换为隶属度:

$$\mu_i = f_3(e_i) = 1 - \frac{e_i}{e_{\max} + \delta}$$

其中, $e_{\max} = \max \{e_1, \dots, e_i, \dots, e_N\}$, $\delta > 0$.

6.4.5.2.3 实验及结果

6.4.5.3 参考文献

6.4.5.4 名词术语

Support Vector Machine 支撑矢量习机 ([Support Vector Machine](http://en.volupedia.org/wiki/Support_vector_machine) (http://en.volupedia.org/wiki/Support_vector_machine))

6.4.6 极速学习机

6.4.6.1 经典极速学习机

6.4.6.1.1 什么是极速学习机

极速学习机 (*Extreme Learning Machine*) 由黄¹ 等人于 2005 年提出, [2]

用于人脸识别 [3]

基于局部感受野 (Local Receptive Fields) 的极速学习机 [4]

多层极速学习机扩展 ([5])

¹ <http://www.ntu.edu.sg/home/egbhuang/index.html>

6.4.6.1.1.1 实验与分析

6.4.6.1.1.2 实验数据

参考 [6] , 数据来源于“[UCI ML<http://archive.ics.uci.edu/ml/datasets.php>](http://archive.ics.uci.edu/ml/datasets.php)”:

- [Breast Cancer Wisconsin](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)) ([http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)))
- [Statlog \(Landsat Satellite\)](http://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)) ([http://archive.ics.uci.edu/ml/datasets/Statlog+\(Landsat+Satellite\)](http://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)))
- [Wine-Quality](http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/) (<http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/>)

6.4.6.1.1.3 实验代码

6.4.6.1.1.4 实验结果

6.4.6.2 基于局部感受野的极速学习机

6.4.6.2.1 说明

本文是“ Local Receptive Fields Based Extreme Learning Machine”¹ 的学习笔记.

文章主要包含两部分内容, 极速学习机 (也有人译作极限学习机或极端学习机, Extreme Learning Machine, ELM) 和局部感受野 (Local Receptive Fields, LRF).

极速学习机 (也有人译作极限学习机或极端学习机, Extreme Learning Machine, ELM) 实际上是一种单隐层前馈神经网络 (Single-hidden Layer Feedforward Neural networks, SLFNs)², 由南洋理工大学黄广斌教授于 2004 年提出, 请参见[主页](http://www.ntu.edu.sg/home/egbhuang/index.html) (<http://www.ntu.edu.sg/home/egbhuang/index.html>). ELM 可用于特征学习 (feature learning), 聚类 (clustering), 回归 (regression) 和分类 (classification).

6.4.6.2.2 摘要内容

- 传统观点: 神经网络的隐藏层神经元需要在训练阶段迭代调整.
- ELM 理论打破了这种信条, 认为隐层神经元虽然很重要, 但不需要迭代调整. 隐藏层节点的所有参数 (权重 W 和偏置 b) 都独立于训练样例, 可以随机的 (任意连续概率分布) 生成, 这样的 ELM 依然具有普遍的逼近和分类能力 ([universal approximation](http://en.wikipedia.org/wiki/Universal_approximation_theorem) (http://en.wikipedia.org/wiki/Universal_approximation_theorem) and classification).

文章提出了一种局部连接的 ELM 的普适结构.

1. 在输入层引入局部感受野;
2. 每个隐层节点可以是几个隐层节点 (子网络, sub-network) 的组合.

在 NORB 数据集上, 与传统的深度神经网络作了对比:

¹ Huang G, Bai Z, Kasun L, et al. Local Receptive Fields Based Extreme Learning Machine[J]. Computational Intelligence Magazine IEEE, 2015, 10(2):18 - 29.

² Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks[J]. Proc.int.joint Conf.neural Netw, 2004, 2:985-990.

- 将错误率从 6.5% 降到 2.7%
- 学习速度快了 200 倍

6.4.6.2.3 引言部分

文中提到, 机器学习的成功依赖于三个关键因素:

1. 强大的计算环境 (powerful computing environments)
2. 丰富的动态数据 (rich and dynamic data)
3. 有效的学习算法 (efficient learning algorithms)

传统的诸如 BP 的训练方法的缺点:

- 大量的梯度下降搜索操作
- 慢的收敛速度
- 容易陷入局部最优
- 密集的人工干预

ELM 克服了这些缺点和限制, 不仅训练时间急剧减少, 学习的精度也非常高.

基于局部感受野的极速学习机 (Local Receptive Fields Based Extreme Learning Machine, ELM-LRF) 和卷积神经网络 (Convolutional Neural Networks, CNNs) 在局部连接上相似, 但有两点不同:

1. 局部感受野: ELM-LRF 可以灵活的使用由连续概率分布随机生成的不同形式的局部感受野; 而 CNN 使用固定的卷积隐层节点作为局部感受野.
2. 训练: CNN 使用 BP 算法; 而 ELM-LRF 的输入权重和偏置可以随机生成, 从而输出权重可以解析地计算.

6.4.6.2.4 回顾 ELM, CNN 和 HTM

6.4.6.2.4.1 极速学习机

ELM 理论³⁴⁵ 表明, 只要隐层神经元的激活函数是非线性分段连续 (nonlinear piecewise continuous) 的, 神经网络就不需要通过迭代调整网络来获得学习能力.

如上图所示, ELM 包含两步: 特征映射和特征学习.

³ Huang G B. Universal approximation using incremental constructive feedforward networks with random hidden nodes.[J]. IEEE Transactions on Neural Networks, 2006, 17(4):879 - 892.

⁴ Huang G B, Chen L, Huang G B, et al. Convex incremental extreme learning machine[J]. Neurocomputing, 2007, 70:3056–3062.

⁵ Huang G B, Chen L. Enhanced random search based incremental extreme learning machine[J]. Neurocomputing, 2008, 71(16-18):3460–3468.

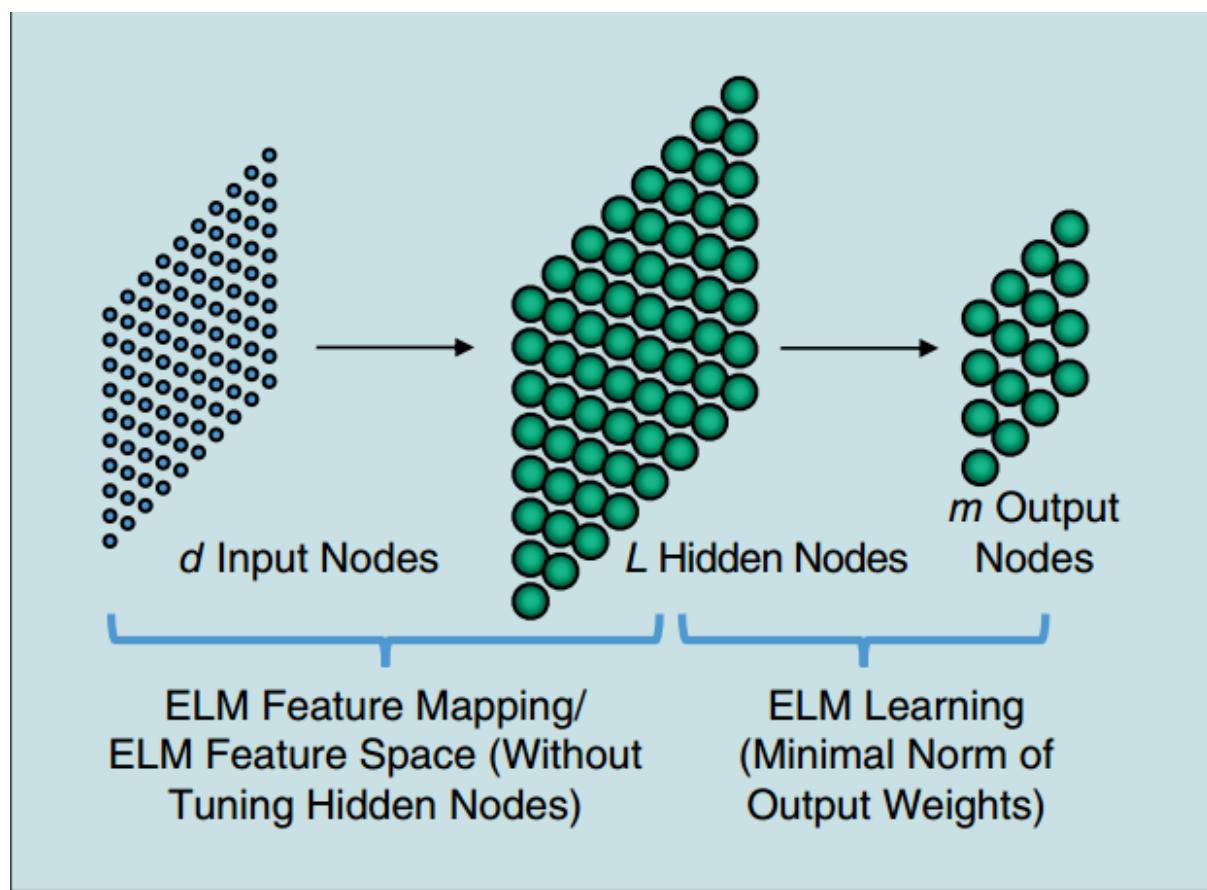


图 6.32: 图 1 ELM 的结构

6.4.6.2.4.2 ELM 特征映射

ELM 的输出函数 (output function):

$$f(\mathbf{x}) = \sum_{i=1}^L \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$$

其中, $\boldsymbol{\beta}_{L \times m} = [\beta_1, \dots, \beta_L]^T$, ($\beta_i = [\beta_{i1}, \dots, \beta_{im}]^T$) 是隐层与输出层间的输出权重矩阵, $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \dots, h_L(\mathbf{x})]$ 是隐层的输出向量.

$$h_i(\mathbf{x}) = G(\mathbf{a}_i, b_i, \mathbf{x}), \mathbf{a}_i \in \mathbb{R}^d, b_i \in R$$

其中, $G(\mathbf{a}_i, b_i, \mathbf{x})$ 是一个非线性分段连续函数.

$\mathbf{h}(\mathbf{x})$ 实际上是将 d 维的输入空间映射到 L 维的隐层随机特征空间, 所以 $\mathbf{h}(\mathbf{x})$ 是一个随机特征映射 (feature mapping).

6.4.6.2.4.3 ELM 特征学习

ELM 与传统的学习算法不同, 隐层神经元无需调整, 而且可以得到最小化训练误差和具有最小范数的解:

$$\min : \|\boldsymbol{\beta}\|_p^{\sigma_1} + C \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_q^{\sigma_2}$$

其中, $\sigma_1 > 0, \sigma_2 > 0, p, q > 0, C$ 用于控制两项的重要性.

对于给定的训练集 $(\mathbf{x}_i, \mathbf{t}_i), i = 1, 2, \dots, N$, 用 \mathbf{H} 表示隐藏层输出矩阵:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \cdots & h_L(\mathbf{x}_N) \end{bmatrix}$$

\mathbf{T} 是训练样例的目标矩阵 (target matrix, 由类标构成):

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix}$$

有很多种方法可以计算权重 $\boldsymbol{\beta}$, 如正交投影的方法、迭代的方法、和奇异值分解等等.

当 $\sigma_1 = \sigma_2 = p = q = 2$ 时, 常用的闭式解 (closed-form) 为:

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}^T (\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1} \mathbf{T}, & \text{if } N \leq L \\ (\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}, & \text{if } N \geq L \end{cases}$$

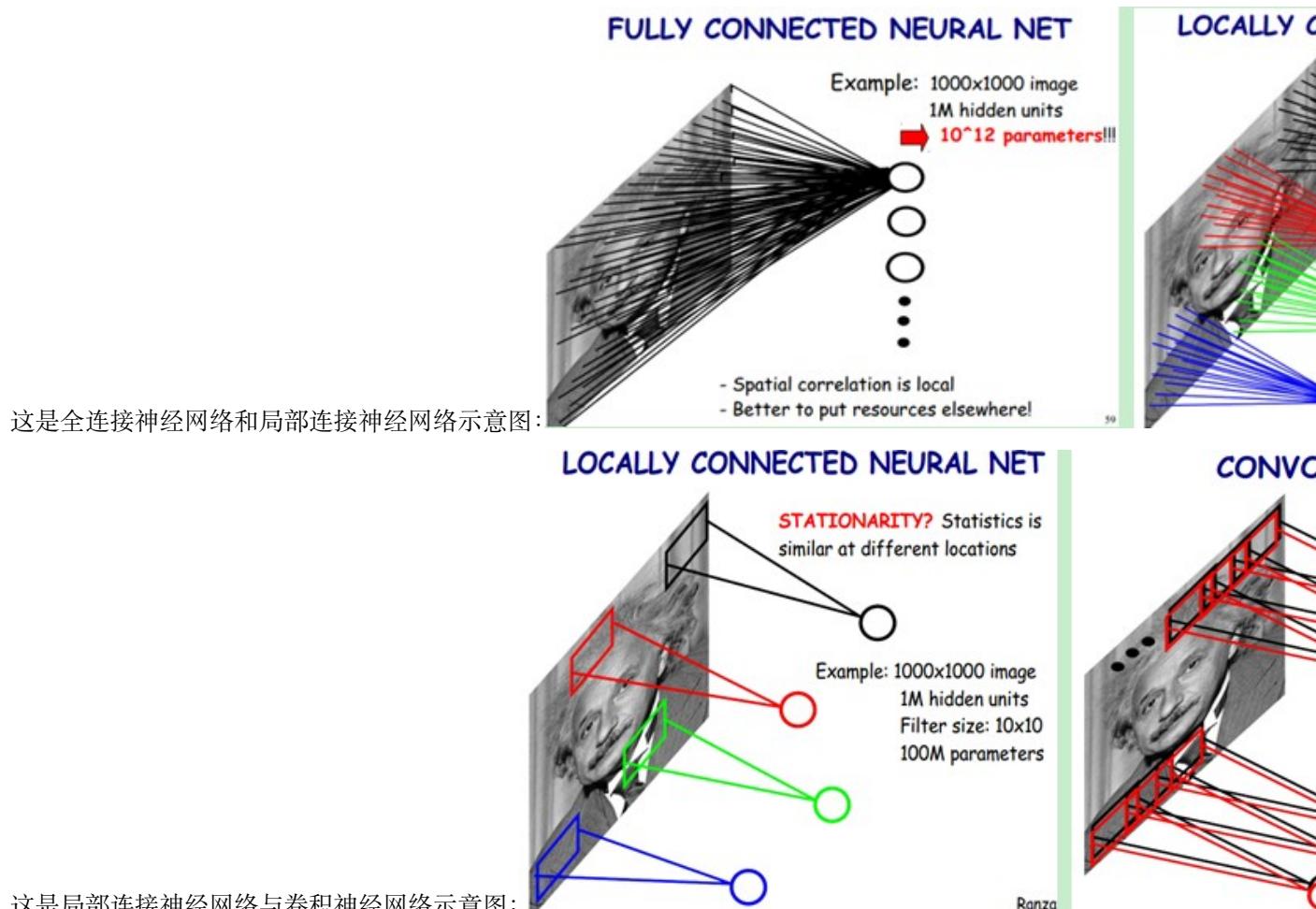
定理 1: 普适近似能力: 设激活函数为任意非常数分段连续函数 (nonconstant piecewise continuous function), 如果通过调整隐层神经元的参数可以使 SLFNs 近似任意目标函数 $f(\mathbf{x})$, 那么可以按照任意连续概率分布函数随机生成序列 $\{h_i(\mathbf{x})\}_{i=1}^L$, 能够找到适当的 $\boldsymbol{\beta}$, 使得极限 $\lim_{L \rightarrow \infty} \|\boldsymbol{\beta}_i h_i(\mathbf{x}) - f(\mathbf{x})\| = 0$ 依概率收敛于 1.

定理 2: 分类能力: 设激活函数为任意非常数分段连续函数, 如果通过调整隐层神经元的参数可以使 SLFNs 近似任意目标函数 $f(\mathbf{x})$, 那么带有随机映射 $h(\mathbf{x})$ 的 SLFNs 可以分离任何形状的任意不相交 (disjoint) 区域.

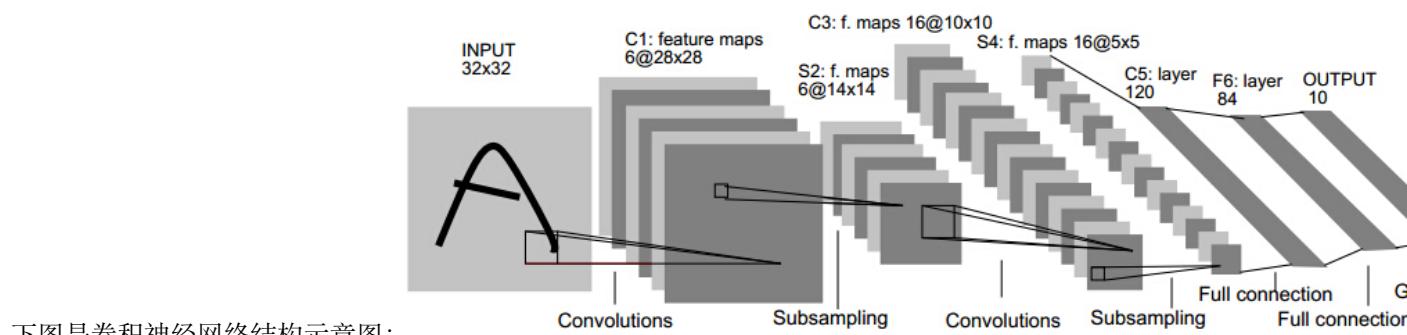
6.4.6.2.4.4 卷积神经网络

卷积神经网络（Convolutional Neural Network, CNN）是多层前馈神经网络（Multi-Layer Feedforward Neural Network, 也叫多层感知器, MLPs）的变种.

CNN 受启发与人类的视觉皮层, 输入至隐藏层采用局部连接.



这是局部连接神经网络与卷积神经网络示意图:



下图是卷积神经网络结构示意图:

CNN 包含两个基本操作: **卷积** (convolution) 和 **池化** (pooling), 通常交替排列卷积层和池化层直至获得高级的特征. 上图中的“Subsampling”其实就是池化操作.

卷积神经网络特点

- 局部连接（网络参数数目减小）
- 权值共享

- 采用 BP 训练 (包含 BP 的弊病)
- 运算量大

6.4.6.2.4.5 卷积

对于一个卷积层, 用 γ 表示该卷积层的值, 用 x 表示其前一层的值, 假设该卷积层前一层的大小是 $d \times d$, 感受野 (receptive field) 的大小是 $r \times r$, 则

$$\gamma_{i,j} = g\left(\sum_{m=1}^r \sum_{n=1}^r x_{i+m-1, j+n-1} \cdot w_{m, n} + b\right), \quad i, j = 1, \dots, (d - r + 1)$$

6.4.6.2.4.6 池化

为减少特征的维数并引入平移不变性, 在局部区域引入池化操作, 通常有平均池化和最大池化.

- 平均池化使得提取的特征对微小变形鲁棒, 与视觉感知中的复杂细胞功能类似.
- 最大池化使得提取的特征具有平移不变性.
- 池化区域通常是不重叠的.

6.4.6.2.4.7 层级实时记忆

层级实时记忆 (Hierarchical Temporal Memory (http://en.wikipedia.org/wiki/Hierarchical_temporal_memory)) , HTM 是一个在线式机器学习模型 (an online machine learning model) , 它能发现和推断出观测输入模式或序列的高层次原因. HTM 组合和扩展了贝叶斯网络、空时聚类算法中的方法, 同时利用了神经网络中常用的节点的树形层次结构.

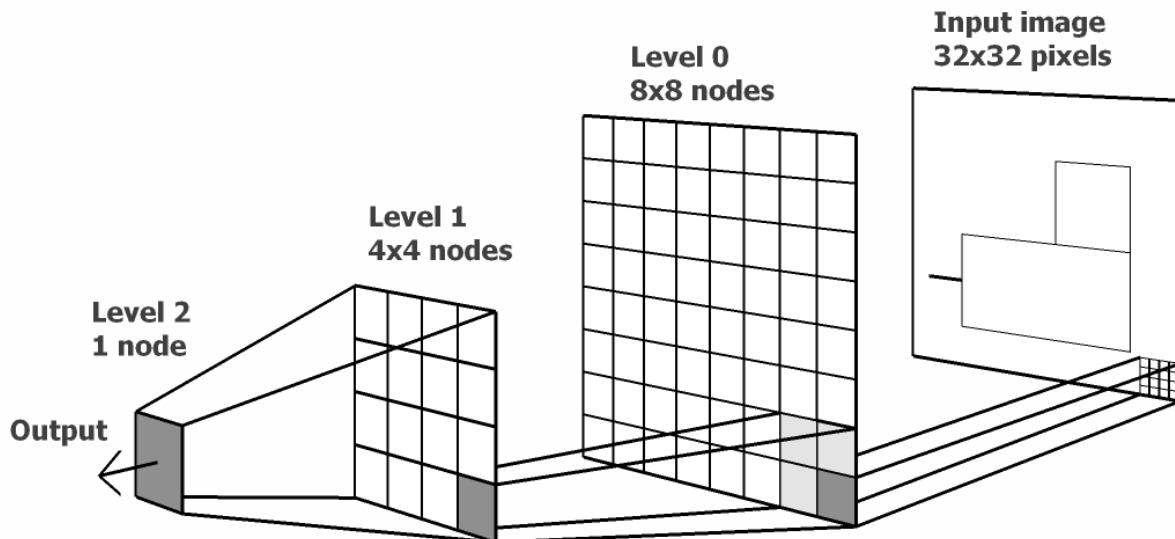


图 6.33: 层级实时记忆 (HTM) 网络结构

6.4.6.2.5 ELM-LRF 原理

- 输入与隐藏层间的连接是稀疏的, 且由相应的局部感受野 (对连续概率分布采样得到) 包围.
- 组合节点: 通过把几个隐藏层节点组合在一起, 引入平移不变性. (translational invariance) .

6.4.6.2.5.1 A. 全连接与局部连接 (Full and Local Connections)

ELM 理论证明, 隐藏层节点可以按照任意概率分布生成, 这里的随机是指:

- 输入与隐藏层节点间的连接密度是根据不同类型的概率分布随机采样得到的.
- 输入与隐藏层节点间的连接权重也是随机生成的.

如下图所示, (a) 图为隐藏层节点全连接的形式, 相关的应用研究很多, 且 ELM 在诸如遥感、时间序列分析、文本分类、行为识别等应用领域取得了最高水平的性能.

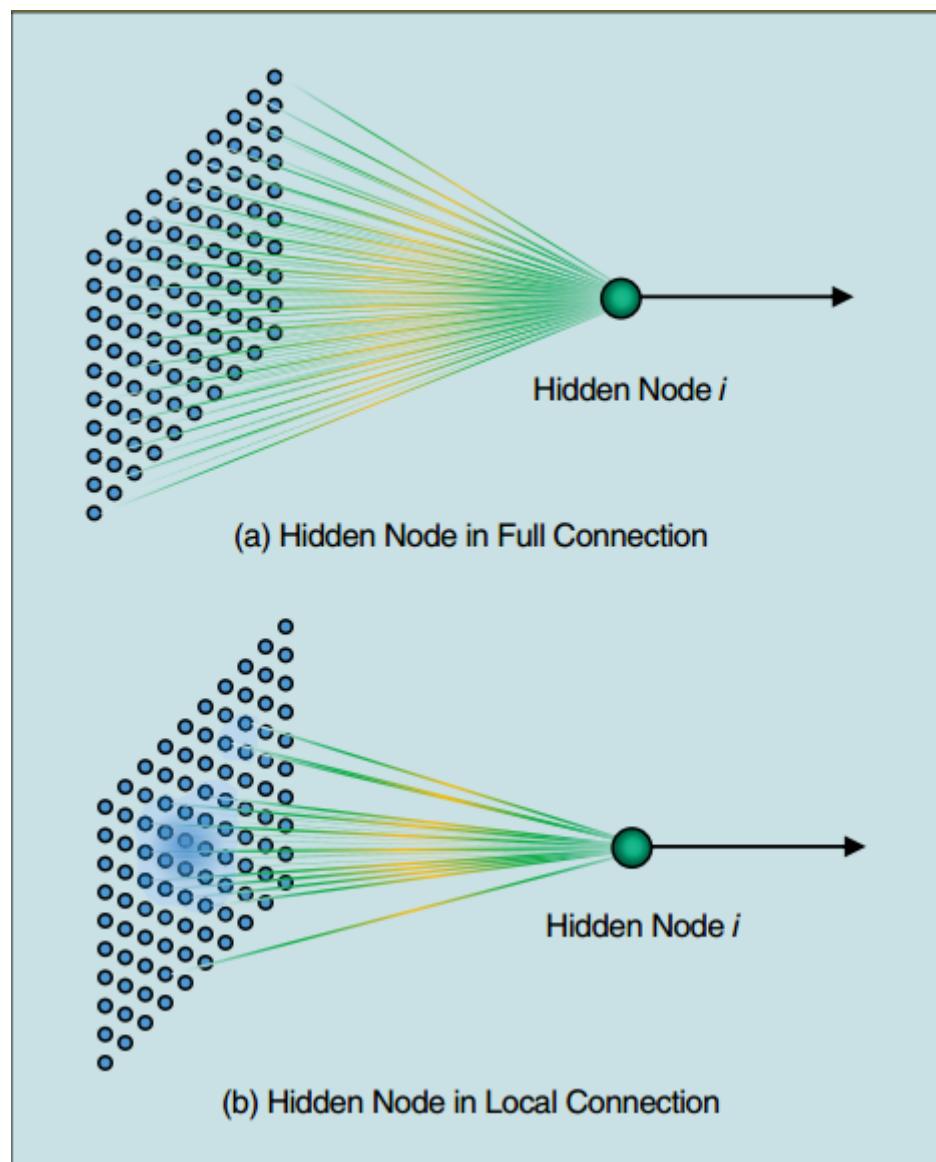


图 6.34: 图 6 全连接与局部连接

然而, 上面的工作仅关注于权重的随机, 忽略了连接也可以随机的属性. 自然图像和语言的强的局部关系, 使得全连接很不适合.

6.4.6.2.5.2 B. 基于局部感受野的 ELM

如上图中 (b) 图所示, 输入层与一个隐藏层节点 i 间的连接是根据连续概率分布随机生成的, 这种随机的连接也就构成了局部感受野.

当 ELM-LRF 应用于图像处理等相似任务时, 它学习图像的局部结构并在隐藏层生成更为有意义的表示.

6.4.6.2.5.3 C. 组合节点

ELM 理论表明, ELM 中的一个隐层节点可以是几个隐层节点的组合, 或者是节点构成的子网络. 如下图 7 所示, 组合节点 i 由一个子网络形成, 这个子网络的输出实际上是对于 3 个局部感受野的 3 个隐藏层节点的和.

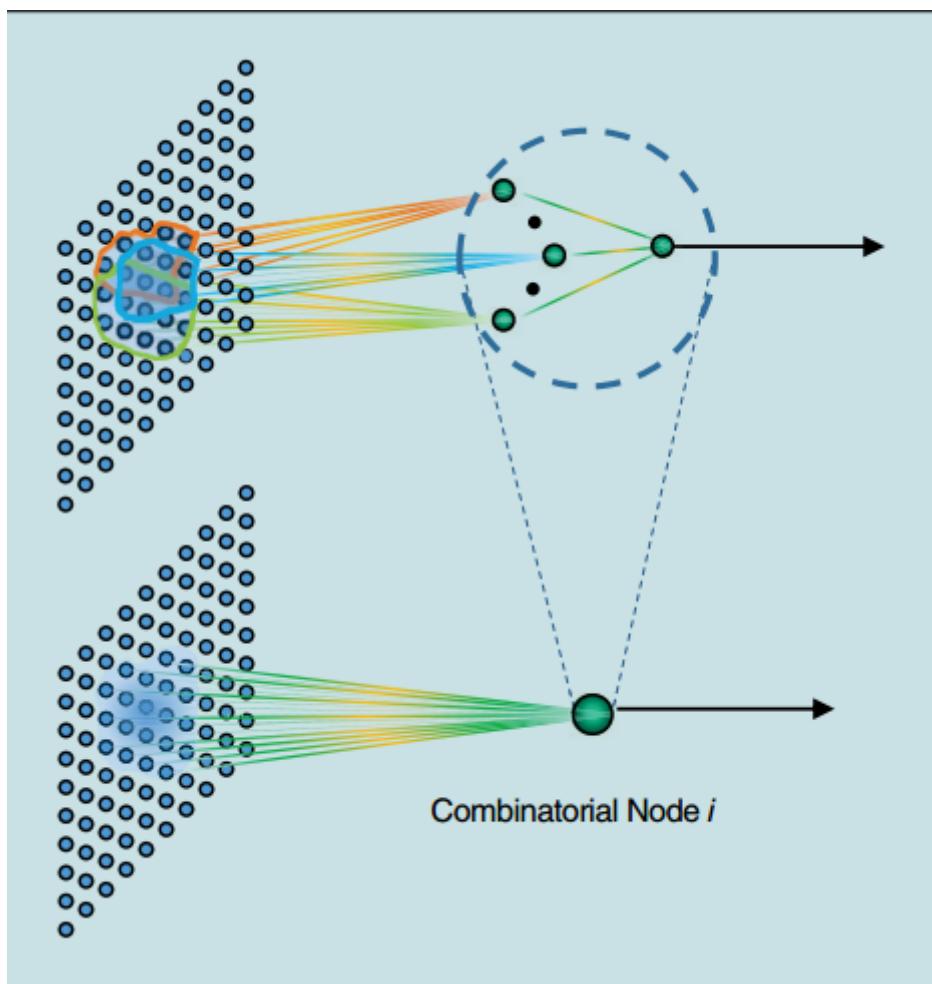


图 6.35: 图 7 ELM 的组合节点

实际上, 组合节点完成了池化的功能:

- 在一个节点生成的特征在不同的节点也有用.

- ELM-LRF 网络具有平移和旋转不变性.
- 输入与组合节点间的连接能更好的学习局部特征.

6.4.6.2.6 局部感受野的实现

6.4.6.2.6.1 A. ELM-LRF 的特殊组合节点

尽管 ELM 中可以使用各种不同的局部感受野和组合节点, 为了方便实现, 文章中采用特殊的局部感受野和组合节点如下图:

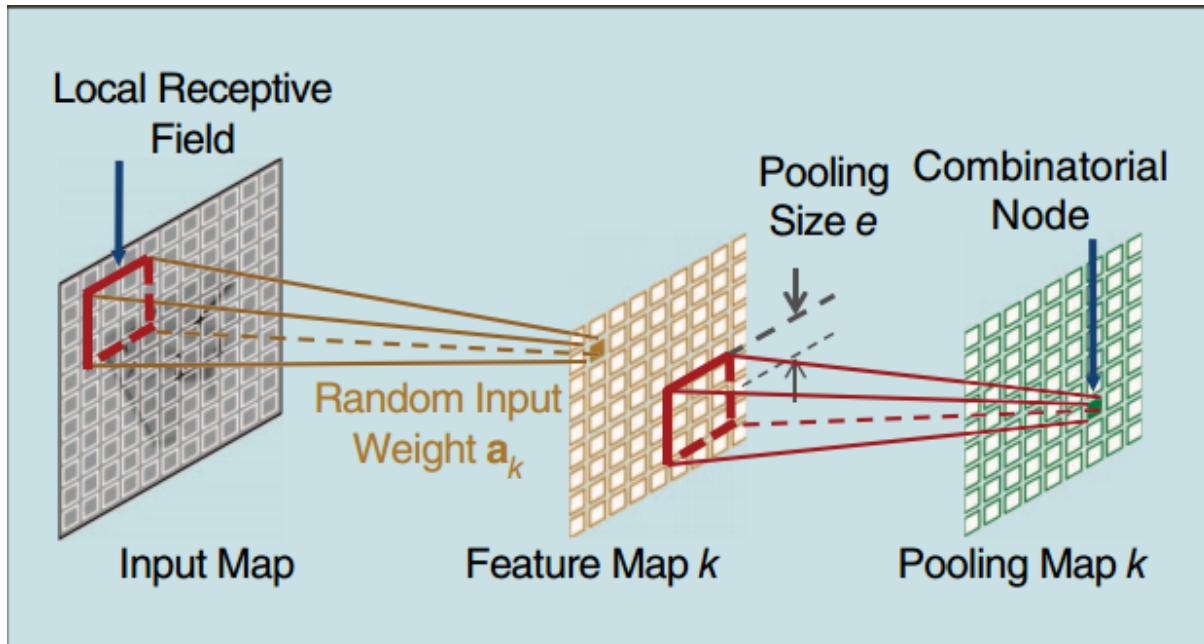


图 6.36: 图 8 文中实现的 ELM-LRF 结构

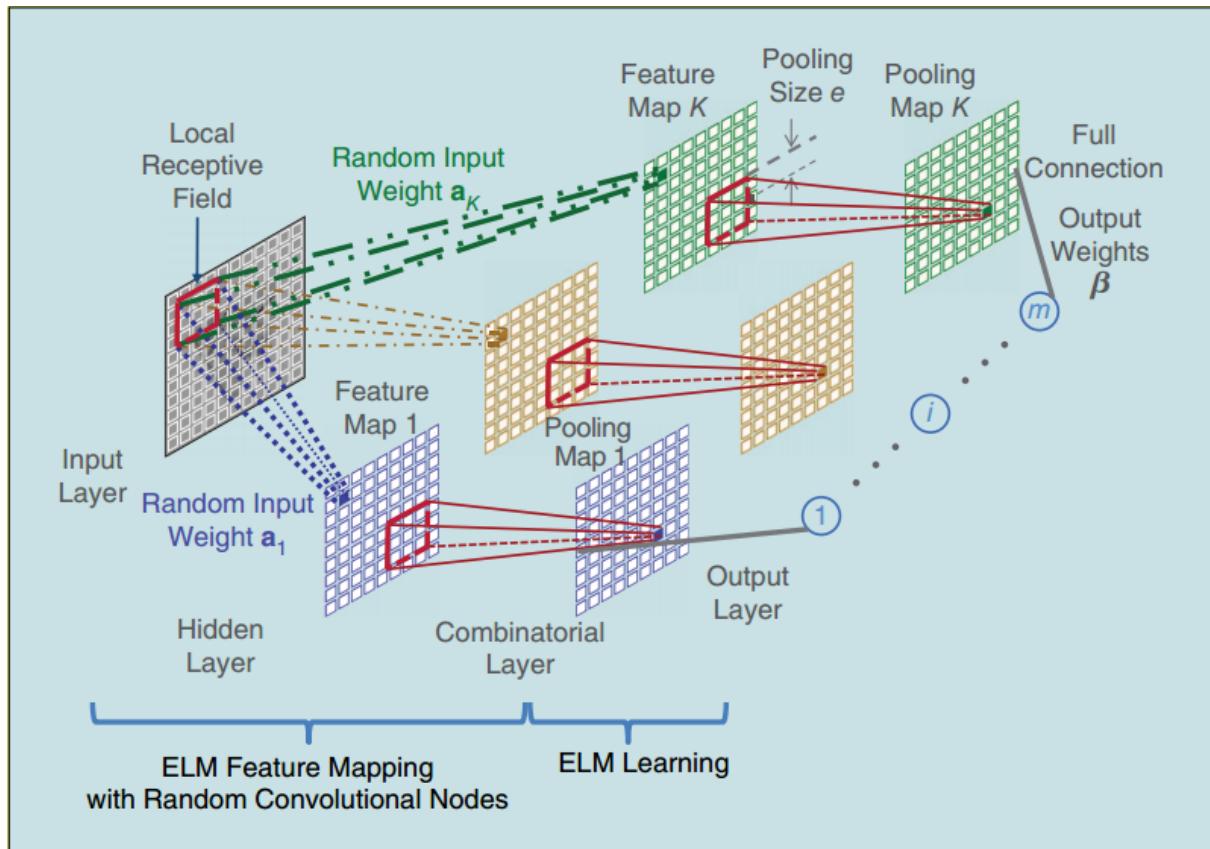
- 采样分布: 采用简单的阶梯概率函数 (Simple Step Probability Function);
- 组合节点: 平方根池化 (square/square-root pooling) 结构;
- 局部感受野: 每个隐层节点的局部感受野由距中心一定距离内的输入节点组成;
- 卷积操作: 对于不同隐藏层节点, 共享输入权重实现卷积操作.

6.4.6.2.6.2 B. 随机输入权重

为了获得输入的充分表示 (thorough representations), 采用 K 个不同的输入权重, 从而得到 K 个互异的特征图:

其中,

- 隐藏层由随机卷积节点组成;
- 同一特征图 (Feature Map) 共享同一输入权重, 不同特征图输入权重不同;
- 输入权重随机生成并正交化, 正交化的输入权重可以提取更为完备的特征.

图 6.37: 带有 K 个特征图的 ELM-LRF 网络的实现

输入权重的生成与正交化操作:

1. 随机生成初始权重 $\hat{\mathbf{A}}^{\text{init}}$. 设输入大小为 $d \times d$, 感受野大小为 $r \times r$, 那么特征图的大小为 $(d - r + 1) \times (d - r + 1)$. 注: 文章采用标准高斯分布, 且不包含偏置, 因为它不需要.

$$\hat{\mathbf{A}}^{\text{init}} \in \mathbb{R}^{r^2 \times K}, \quad \hat{\mathbf{A}}^{\text{init}} = [\hat{\mathbf{a}}_1^{\text{init}}, \hat{\mathbf{a}}_2^{\text{init}}, \dots, \hat{\mathbf{a}}_K^{\text{init}}], \quad \hat{\mathbf{a}}_k^{\text{init}} \in \mathbb{R}^{r^2}, \quad k = 1, \dots, K$$

2. 正交化初始权重 $\hat{\mathbf{A}}^{\text{init}}$. 采用奇异值分解 (SVD) 正交化, 正交化的初始权重记为 $\hat{\mathbf{A}}$, 它的每一列 $\hat{\mathbf{a}}_k$ 都是 $\hat{\mathbf{A}}^{\text{init}}$ 的正交基. 注意, 当 $r^2 < K$ 时, 先转置, 再正交化, 然后转置回来.

第 k 个特征图的输入权重是 $\mathbf{a}_k \in \mathbb{R}^{r \times r}$, 由 $\hat{\mathbf{a}}_k$ 逐列排成. 第 k 个特征图的卷积节点 (i, j) 的值 $c_{i,j,k}$ 由下式计算:

$$c_{i,j,k}(\mathbf{x}) = \sum_{m=1}^r \sum_{n=1}^r (x_{i+m-1, j+n-1} \cdot a_{m,n,k}), \quad i, j = 1, \dots, (d - r + 1) \quad (6.21)$$

6.4.6.2.6.3 C. 平方根池化 (square/square-root pooling) 结构

池化大小 e 表示池化中心到边的距离, 且池化图 (pooling map) 与特征图大小相同 ($(d - r + 1) \times (d - r + 1)$). $c_{i,j,k}$ 和 $h_{p,q,k}$, 分别表示第 k 个特征图中的节点 (i, j) 和第 k 个池化图中的组合节点 (p, q) .

$$h_{p,q,k} = \sqrt{\sum_{i=p-e}^{p+e} \sum_{j=q-e}^{q+e} c_{i,j,k}^2}, \quad p, q = 1, \dots, (d - r + 1) \quad (6.22)$$

if (i, j) is out of bound, then $c_{i,j,k} = 0$.

- 平方与求和操作: 网络引入非线性校正 (rectification nonlinearity) 和平移不变性 (translation invariance) 的特性;
- 卷积操作后紧跟平方/平方根池化结构: 使网络具有频率选择性 (frequency selective) 和平移不变性 (translation invariance);
- 因而非常适合于图像处理.

6.4.6.2.6.4 D. 基于输出权重的闭式解

池化层与输出层全连接, 输出权重 β , 采用正则化最小二乘 (Regularized Least-Squares) 法解析地计算.

对于每一个输入样例 x , 使用式.6.21 计算特征图的值, 然后使用式.6.22 计算池化图 (即组合层) 的值. 简单地连接所有组合节点的值形成一个行向量, 并把 N 个输入样例的行向量放在一起, 得到组合层矩阵 $H \in \mathbb{R}^{N \times K \cdot (d-r+1)^2}$, 输出权重矩阵计算为:

1. if $N \leq K \cdot (d - r + 1)^2$

$$\beta = H^T \left(\frac{I}{C} + HH^T \right)^{-1} T$$

2. if $N > K \cdot (d - r + 1)^2$

$$\beta = \left(\frac{I}{C} + H^T H \right)^{-1} H^T T$$

6.4.6.2.7 讨论

6.4.6.2.7.1 A. 普适近似和分类能力

1. 输入与隐藏层节点间的连接, 是根据不同类型的连续概率分布随机采样构建的, 这样的网络依然具有普适近似能力和分类能力.
2. 输入与隐藏层节点间没有连接的, 可以认为连接权重不重要以至于可以忽略, 因而仍然可以认为分布函数是连续的, 可以保持网络的普适近似与分类能力.
3. ELM 中的隐藏层节点可以是不同节点的线性或非线性组合.

由于隐藏层节点的激活函数是非线性分段连续的, 所以第 k 个池化图 $h_{p,q,k}$ 中的组合节点 (p, q) , 仍然可以表示成 ELM 隐层节点的基本形式:

$$h_{p,q,k} = G(a_{p,q}, b_{p,q}, x), p, q = 1, \dots, (d - r + 1)$$

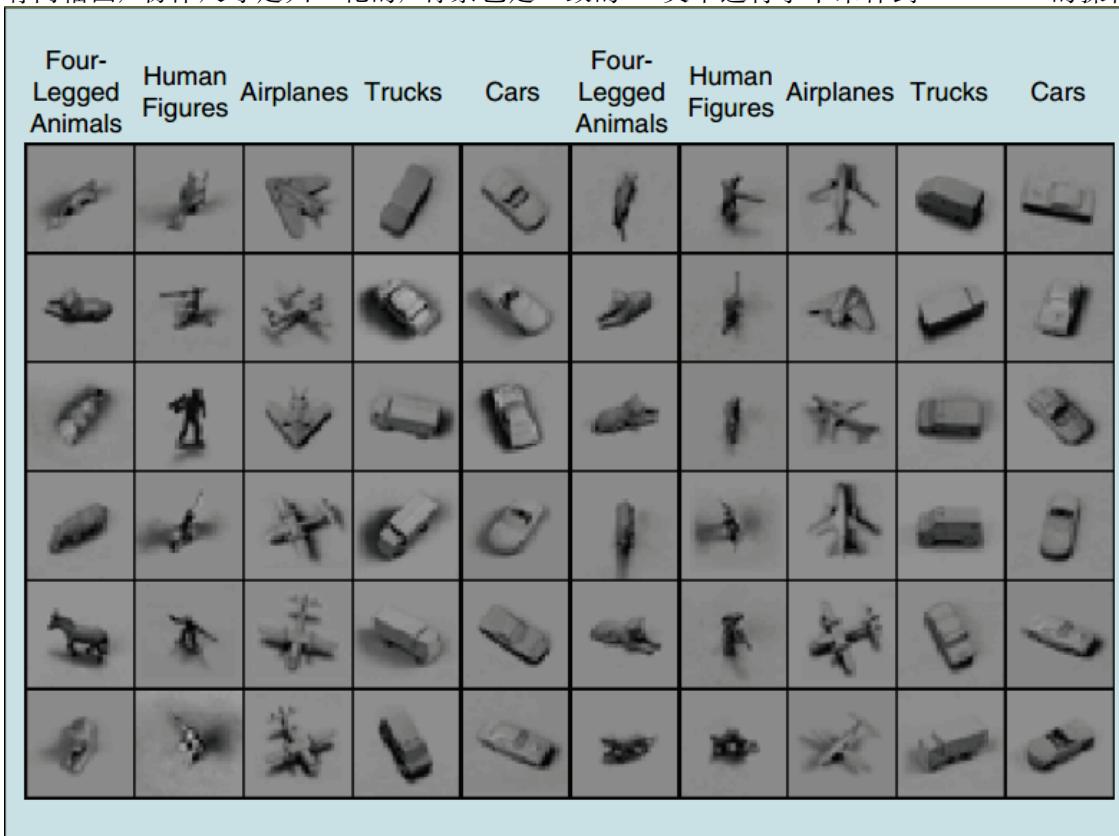
在平方根池化结构中, G 显然是非线性分段连续的, 所以 ELM-LRF 仍然保留了普适近似与分类能力, 从而可以学习输入数据更为复杂的特征.

6.4.6.2.7.2 B. ELM-LRF 与 HTM 和 CNN 的关系

- ELM-LRF 与 HTM: 在通过构造一层一层的学习模式, 来模拟大脑处理逐渐复杂的输入形式上是相似的; 然而, ELM-LRF 更为有效, 因为 ELM-LRF 网络的连接和输入权重都是随机生成的, 而 HTM 需要仔细设计网络和调整参数.
- ELM-LRF 与 CNN: 它们都直接处理原始输入, 并利用局部连接来限制网络学习诸如自然图像和语言中的空间相关性. 它们的不同是:
 1. 局部感受野: ELM-LRF 更为灵活和宽泛, 可以根据不同类型的概率分布随机采样生成, 而 CNN 只使用卷积隐藏层节点; 尽管本文仅使用随机卷积节点作为 ELM-LRF 的特殊的局部感受野, 研究其它类型的感受野也是很有价值的.
 2. 训练: CNN 中的隐藏层节点需要调整, 而通常采用 BP 算法, 这使得 CNN 面临 BP 中的琐碎问题, 如: 局部最优, 慢的收敛速度. 而 ELM-LRF 随机生成输入权重并解析地计算输出权重. 也就是计算主要是输出权重的计算, 从而 ELM-LRF 更为高效.

6.4.6.2.8 实验

1. 实验数据 ELM-LRF 与 Deep Learning 的方法进行了对比, 数据集选择目标识别数据集: NORB. NORB 包含 24300 幅训练用立体图像 (stereo image) 和 24300 幅测试用立体图像, 每个都有 5 类并且很多都进行了 3D 和光照处理. 下图是 NORB 数据集中的 60 个样例, 每个样本有两幅图, 物体尺寸是归一化的, 背景也是一致的. 文中进行了下采样到 32×32 的操作.



2. 实验平台与参数实验平台: MATLAB2013a, Intel Xeon E5-2650, 2GHz GPU, 256GB RAM. 参数: 感受野大小 $\{4 \times 4, 6 \times 6\}$; 特征图的数量 $\{24, 36, 48, 60\}$; 池化大小 $1, 2, 3, 4$; C 的值 $\{0.01, 0.1, 1, 10, 100\}$, 采用 5 倍交叉验证, 来选择参数, 最优参数如 表 6.1 所示.

表 6.1: ELMLRF 的最优参数

DATASET	# OF TRAINING DATA	# OF TESTING DATA	INPUT DIMENSIONS	RECEPTIVE FIELD	# OF FEATURE MAPS	POOLING SIZE	C
NORB	24300	24300	$32 \times 32 \times 2$	4×4	48	3	0.01

6.4.6.2.8.1 A. 测试误差

如 表 6.2 所示, ELM-LRF 要比其它微调的算法的精度更高, 而且耗时少. 与 CNN 和 DBN 的方法相比, ELM-LRF 将错误率从 6.5% 降到 2.74%.

表 6.2: 不同算法在 NORB 数据集上的测试误差

	ALGORITHMS	TEST ERROR RATES
	ELM-LRF	2.74%
	ELM-LRF (NO ORTHOGONALIZATION)	4.01%
	RANDOM WEIGHTS (ELM FEATURE MAPPING + SVM CLASSIFIER)	4.8%
	K-MEANS + SOFT ACTIVATION	2.8%
	TILED CNN	3.9%
	CNN	6.6%
	DBN	6.5%

6.4.6.2.8.2 B. 训练时间

公平起见, 其它的算法也运行在本实验平台, 如 表 6.3 所示, ELM-LRF 学习速度比其它算法快至 200 倍.

表 6.3: 不同算法在 NORB 数据集上的测试速度

	ALGORITHMS	TRAINING TIME(s)	SPEEDUP TIMES
	ELM-LRF	394.16	217.47
	ELM-LRF (NO ORTHOGONALIZATION)	391.89	218.73
	RANDOM WEIGHTS (ELM FEATURE MAPPING + SVM CLASSIFIER)	1764.28	48.58
	K-MEANS + SOFT ACTIVATION	6920.47	12.39
	TILED CNN	15104.55	5.67
	CNN5	53378.16	1.61
	DBN	85717.14	1

6.4.6.2.8.3 C. 特征图

下图显示了一个样本的 48 个特征图. 可以看出, 这些特征图的轮廓线相似的, 这是由于它们来自同一幅输入图像. 然而每个图都有自己明显突出的部分, 这就获得了原始图像的互异表示, 就原始图像的不同抽象, 使得分类变得容易和准确.

6.4.6.2.8.4 D. 随机输入权重的正交化

实验中也分析了随机输入权重的正交化的贡献. 以 48 个特征图中的卷积节点中心的值为例, `fig=RandomWeightsOrthogonalizationELMLRF` 显示了 48 个特征图中, 中心卷积节点的值在对输入权重正交化前后的变化分布.

由 `fig=RandomWeightsOrthogonalizationELMLRF` 可以看出, 正交的随机权重的分布更均匀, 特征图中的其它位置的卷积节点也是如此. 所以正交化使得物体更加线性独立和易分类的. 不过, 即使不正交化, 仍能获得 4.01% 的测试误差与传统方法相比, 减少了 38%.

6.4.6.2.9 结论

- ELM 中引入局部感受野来学习局部结构;
- 组合节点的引入使网络具有平移不变性;
- 输入权重随机生成, 然后进行正交化, 这样可以提取更为完备的特征;
- 输出权重可以解析地计算, 计算复杂度低;
- 局部感受野的形式多样;
- 随机卷积节点可以作为 ELM 的一个有效的局部感受野实现方法;
- 实验表明, 无论在精度上还是学习速度上, ELM-LRF 都远优于传统的深度学习方法.

进一步的工作:

1. ELM 的不同类型的局部感受野的影响;
2. ELM 的不同卷积节点的影响;
3. 堆栈式 ELM-LRF, 可以通过在前一组合层后采用局部连接来堆叠 ELM-LRF.

6.4.6.2.10 代码实现

6.4.6.2.10.1 Source Code

By Me: <https://github.com/antsfamily/ELM-LRF>

Another: <https://github.com/ExtremeLearningMachines/ELM-LRF>

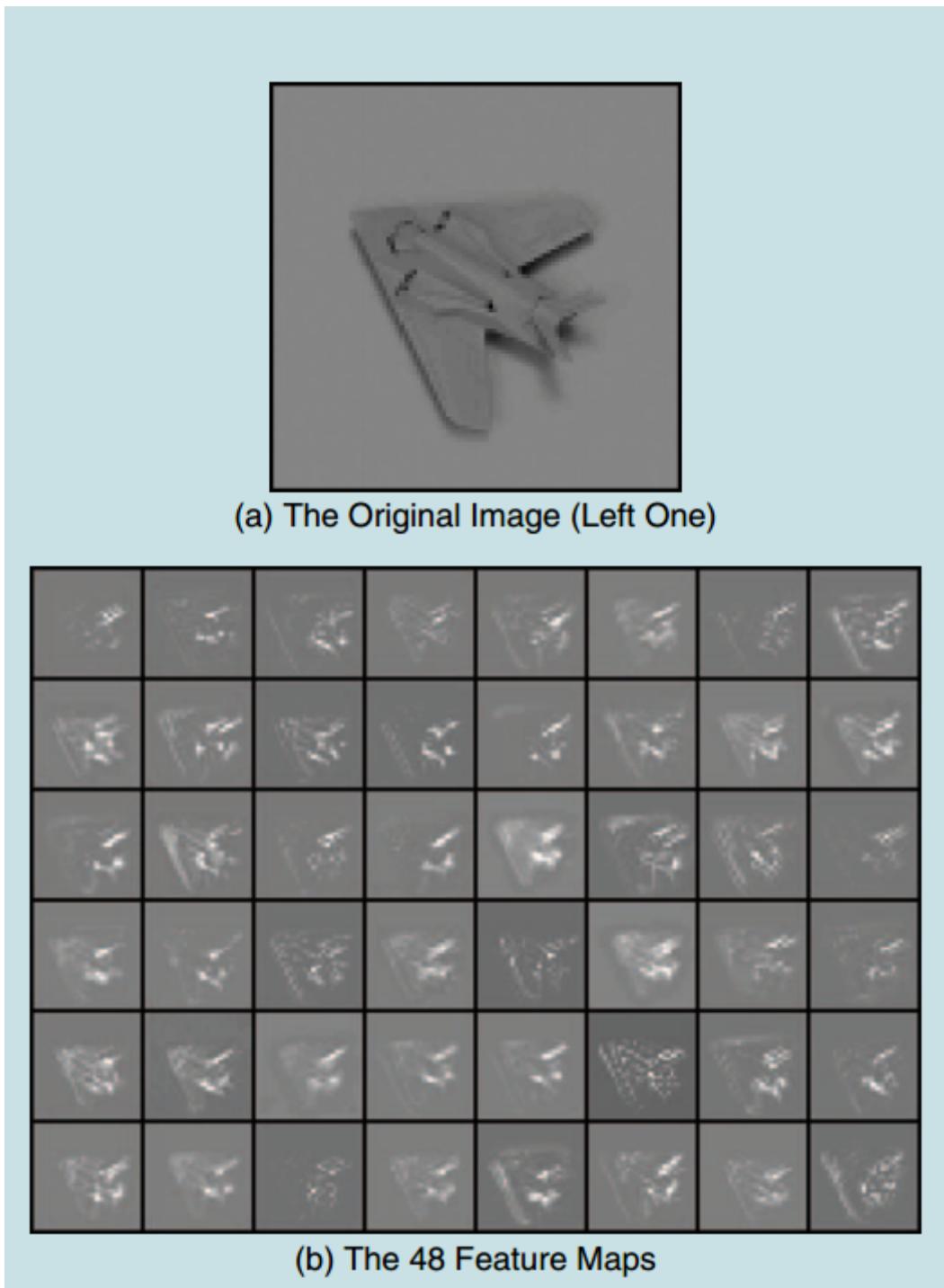


图 6.38: 图 11 一个样例: (a) 原始图像; (b) 48 个特征图

6.4.6.2.10.2 Experimental Results

6.4.6.2.10.3 MNIST 数据集

1. 数据集: MNIST, 28*28, uint8, 60K 训练, 10K 测试
2. 硬件: Intel i5-3210M 2.5GHz 双核四线程 (实验中只用单线程)
3. 软件: MATLAB
4. 参数: $C = [0.001 \ 0.01 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]$; 卷积核: 5×5 , 特征图个数: 10, 池化窗口 3×3 .

注: 由于笔记本内存仅有 4G, Matlab 无法开出 $60K \times 60K$ 的矩阵, 仅用前 10K 个作为训练.

结果如下:

```
With C = 0.001000
-----
Training error: 0.030900
Training Time: 95.156250s

Testing error: 0.035800
Testing Time: 16.250000s

With C = 0.010000
-----
Training error: 0.014400
Training Time: 94.109375s

Testing error: 0.031900
Testing Time: 14.859375s

With C = 0.100000
-----
Training error: 0.005000
Training Time: 93.234375s

Testing error: 0.031700
Testing Time: 14.750000s

With C = 0.200000
-----
Training error: 0.003200
Training Time: 92.484375s

Testing error: 0.032500
Testing Time: 15.000000s

With C = 0.300000
```

(下页继续)

(续上页)

```
-----  
Training error: 0.002500  
Training Time:92.718750s
```

```
Testing error: 0.033700  
Testing Time:14.953125s
```

```
With C = 0.400000
```

```
-----  
Training error: 0.002400  
Training Time:93.937500s
```

```
Testing error: 0.034500  
Testing Time:14.578125s
```

```
With C = 0.500000
```

```
-----  
Training error: 0.002200  
Training Time:91.828125s
```

```
Testing error: 0.035000  
Testing Time:14.828125s
```

```
With C = 0.600000
```

```
-----  
Training error: 0.002100  
Training Time:93.453125s
```

```
Testing error: 0.035400  
Testing Time:14.781250s
```

```
With C = 0.700000
```

```
-----  
Training error: 0.001900  
Training Time:92.218750s
```

```
Testing error: 0.035800  
Testing Time:14.625000s
```

```
With C = 0.800000
```

```
-----  
Training error: 0.001700  
Training Time:93.531250s
```

```
Testing error: 0.036500  
Testing Time:16.421875s
```

(下页继续)

(续上页)

```
With C = 0.900000
-----
Training error: 0.001500
Training Time: 93.343750s

Testing error: 0.037100
Testing Time: 15.359375s

With C = 1.000000
-----
Training error: 0.001400
Training Time: 95.015625s

Testing error: 0.037500
Testing Time: 15.109375s
```

哈哈, 效果不错!

6.4.6.2.10.4 NORB 数据集

2016 年 3 月 11 日补充:

一个月前升级了内存条, 今天又看到有国外童鞋 Email 问我文章中是如何处理对待 NORB 数据集的, 它是个双通道的, 即 $32 \times 32 \times 2 \times 24300$, 这个文中没有细说, 我之前实现的代码也只是单通道的, 所以今天又重新看了一下, 并更新了代码, 在[这里](https://github.com/antsfamily/ELM-LRF) (<https://github.com/antsfamily/ELM-LRF>) 下载.

6.4.6.2.10.5 论文中代码结果

先看看 <https://github.com/ExtremeLearningMachines/ELM-LRF> 代码 (这个应该是文章用的) 实验结果. 里面的代码给了两组参数, 第一组参数 (load param1) 与第二组参数 (load param5) 中感受野 (或卷积核) 大小均为 4×4 , 池化大小为 3×3 , $C = 0.01$ 不同的是特征图的数量, 第一组为 1, 第二组为 48. 第一组结果如下, 第二组根本不能跑, 内存升级到 10G 还是不行.

第一组参数运行结果 (特征图数 1):

```
train_accuracy =
0.7687

The testing begins:

test_accuracy =
0.6575
```

(下页继续)

(续上页)

```
ans =  
  
    train_time: 0.7885  
    test_time: 0.0338  
    train_accuracy: 0.7687  
    test_accuracy: 0.6575  
    time_network: 3.8226  
    time_transform_test_data: 3.4379
```

第一组参数运行结果 (特征图数 3):

```
train_accuracy =  
  
0.9627  
  
The testing begins:  
  
test_accuracy =  
  
0.8342  
  
  
ans =  
  
    train_time: 6.5180  
    test_time: 0.1000  
    train_accuracy: 0.9627  
    test_accuracy: 0.8342  
    time_network: 11.7311  
    time_transform_test_data: 10.9387
```

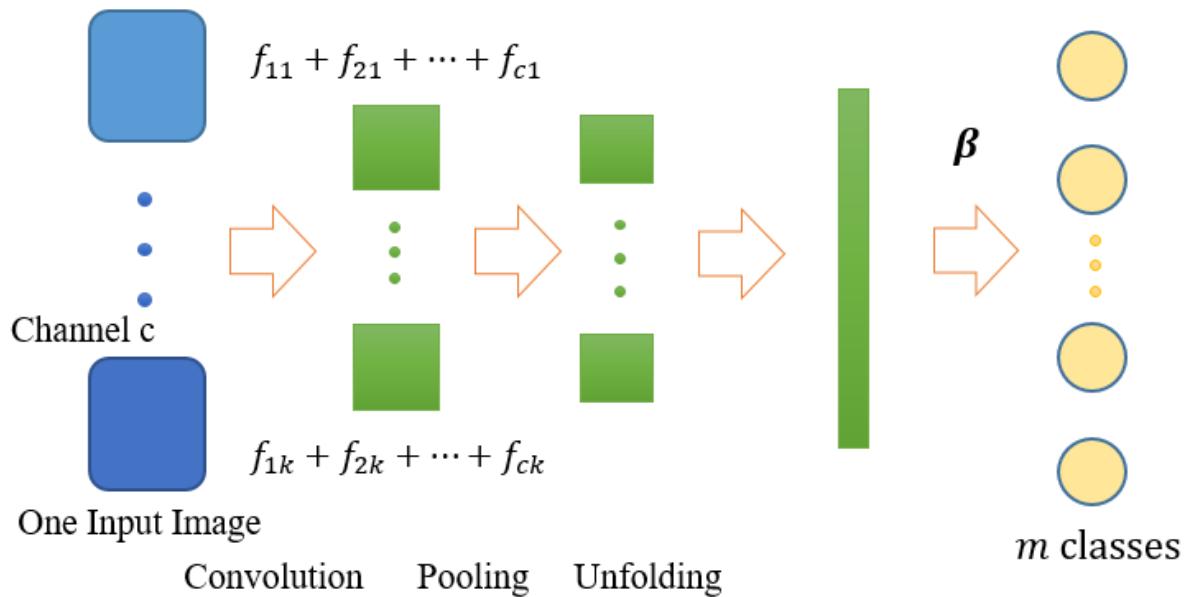
效果提高了好多, 有木有!

6.4.6.2.10.6 本人实现代码结果

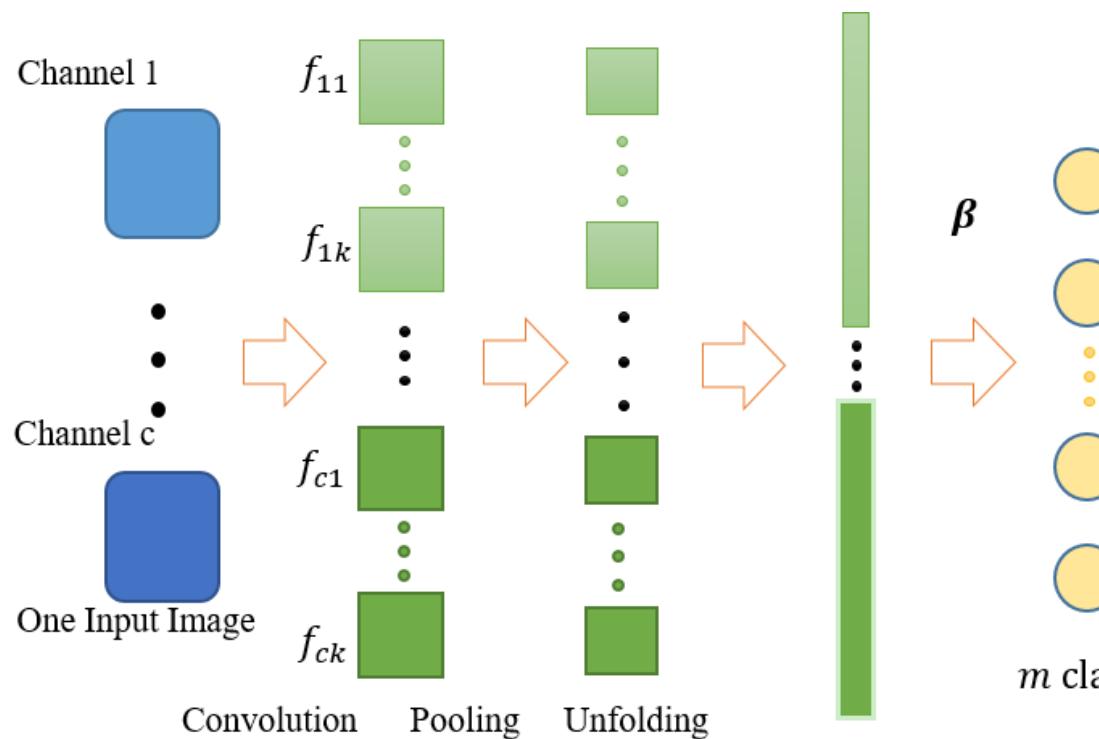
重写的代码支持多通道图像, 输入为 $H - W - N - C$ 的矩阵, 其中 C 为图像通道数.

以下图片本人原创实现了两种处理方式, 一是 “sequential” 模式, 如下:

Channel 1



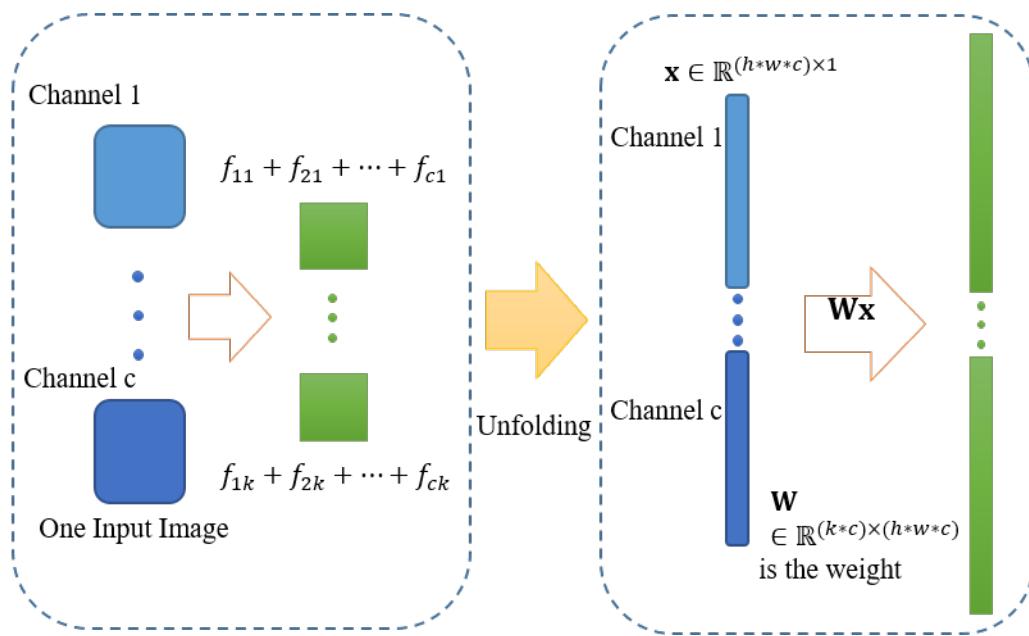
Sequential: One channel, one convolutional kernel, but convolutional feature maps of each channel are added up.



Parallel: One channel, one convolutional kernel, and convolutional feature maps of each channel are stored in parallel.

二是“parallel”模式, 如下:

论文中用的 (<https://github.com/ExtremeLearningMachines/ELM-LRF>) 代码 (这个应该是文章用的), 采用的是 sequential 模型, 且没有直接采用卷积函数计算, 而是将卷积计算转化为矩阵乘积计算 WX .



采用 parallel 模型，参数与上述第一组一致，并使 C 取不同的值，即： $C = [0.0010.010.10.20.30.40.50.60.70.80.91]$ ，结果如下：

特征图数 1：

```
>> pack
>> demo_elmlrf_NORB

With C = 0.001000
-----
Training error: 0.141934
Training Time: 9.234375s

Testing error: 0.257778
Testing Time: 3.703125s

With C = 0.010000
-----
Training error: 0.074239
Training Time: 9.015625s

Testing error: 0.208066
Testing Time: 3.718750s

With C = 0.100000
-----
Training error: 0.042140
Training Time: 8.906250s

Testing error: 0.216049
Testing Time: 3.656250s
```

(下页继续)

(续上页)

```
With C = 0.200000
-----
Training error: 0.035021
Training Time: 8.875000s

Testing error: 0.220206
Testing Time: 3.593750s

With C = 0.300000
-----
Training error: 0.031193
Training Time: 8.906250s

Testing error: 0.223580
Testing Time: 3.687500s

With C = 0.400000
-----
Training error: 0.029342
Training Time: 8.812500s

Testing error: 0.226626
Testing Time: 3.593750s

With C = 0.500000
-----
Training error: 0.027572
Training Time: 8.953125s

Testing error: 0.228354
Testing Time: 3.812500s

With C = 0.600000
-----
Training error: 0.026667
Training Time: 8.953125s

Testing error: 0.230206
Testing Time: 3.625000s

With C = 0.700000
-----
Training error: 0.026420
Training Time: 8.812500s

Testing error: 0.231811
```

(下页继续)

(续上页)

```
Testing Time: 3.687500s
```

```
With C = 0.800000
```

```
-----
```

```
Training error: 0.025926
```

```
Training Time: 8.875000s
```

```
Testing error: 0.233539
```

```
Testing Time: 3.734375s
```

```
With C = 0.900000
```

```
-----
```

```
Training error: 0.025226
```

```
Training Time: 8.750000s
```

```
Testing error: 0.234733
```

```
Testing Time: 3.656250s
```

```
With C = 1.000000
```

```
-----
```

```
Training error: 0.024609
```

```
Training Time: 8.921875s
```

```
Testing error: 0.236091
```

```
Testing Time: 3.718750s
```

特征图数 3:

```
>> demo_elmlrf_NORB
```

```
With C = 0.001000
```

```
-----
```

```
Training error: 0.027284
```

```
Training Time: 61.468750s
```

```
Testing error: 0.114856
```

```
Testing Time: 14.625000s
```

```
With C = 0.010000
```

```
-----
```

```
Training error: 0.008272
```

```
Training Time: 64.171875s
```

```
Testing error: 0.088724
```

```
Testing Time: 13.921875s
```

```
With C = 0.100000
```

(下页继续)

(续上页)

```
-----  
Training error: 0.001564  
Training Time:61.515625s  
  
Testing error: 0.104033  
Testing Time:13.281250s  
  
With C = 0.200000  
-----  
Training error: 0.001070  
Training Time:60.546875s  
  
Testing error: 0.111111  
Testing Time:12.625000s  
  
With C = 0.300000  
-----  
Training error: 0.000947  
Training Time:58.468750s  
  
Testing error: 0.116831  
Testing Time:12.765625s  
  
With C = 0.400000  
-----  
Training error: 0.000741  
Training Time:56.328125s  
  
Testing error: 0.121152  
Testing Time:12.281250s  
  
With C = 0.500000  
-----  
Training error: 0.000700  
Training Time:56.671875s  
  
Testing error: 0.123498  
Testing Time:12.765625s  
  
With C = 0.600000  
-----  
Training error: 0.000658  
Training Time:61.468750s  
  
Testing error: 0.125021  
Testing Time:13.328125s
```

(下页继续)

(续上页)

```
With C = 0.700000
-----
Training error: 0.000617
Training Time: 58.515625s

Testing error: 0.126584
Testing Time: 12.828125s

With C = 0.800000
-----
Training error: 0.000576
Training Time: 59.359375s

Testing error: 0.127984
Testing Time: 12.796875s

With C = 0.900000
-----
Training error: 0.000535
Training Time: 60.296875s

Testing error: 0.129136
Testing Time: 12.156250s

With C = 1.000000
-----
Training error: 0.000494
Training Time: 62.718750s

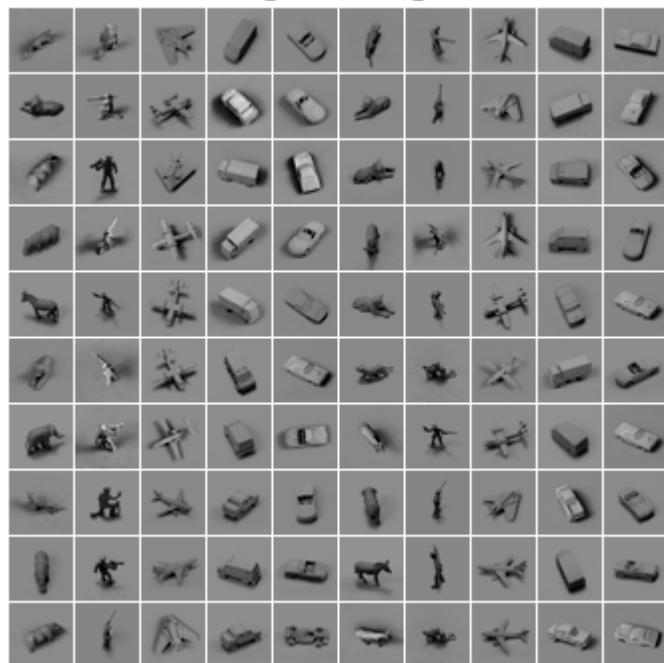
Testing error: 0.130082
Testing Time: 12.750000s
```

根据实验结果, 可知:

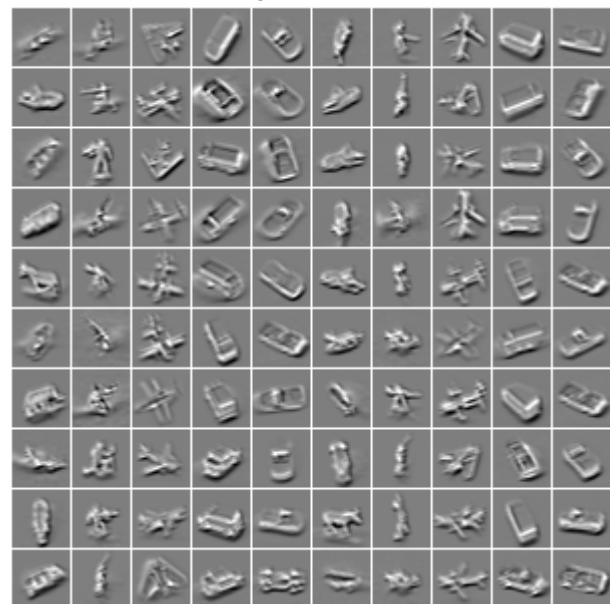
- 相同参数下, 本人实现的程序比文中略高 (精确度);
- 相同参数下, 本人实现的程比论文程序更序耗时些, 有待优化, 但容易扩展.

特征图展示:

下面分别贴出 NORB 数据集前 100 幅图像的: 原图、卷积操作后的特征图、均方根池化后的特征图. 可见隐藏层学习到了数据的特征.

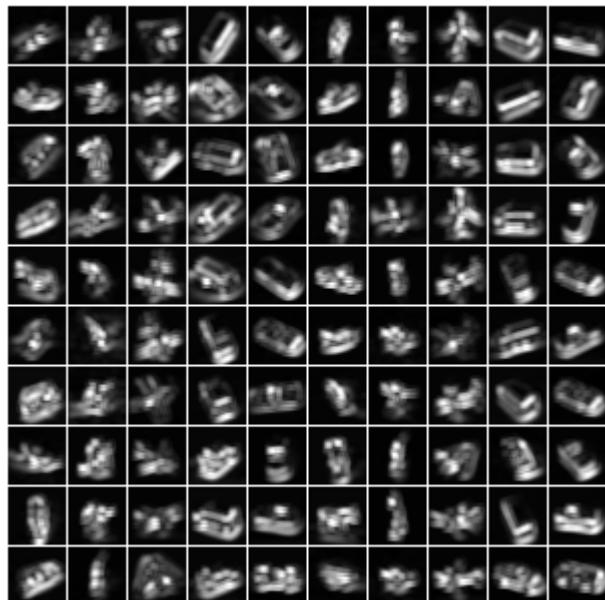
Origional images

NORB 数据集前 100 幅图像:

Feature map after convolution

NORB 数据集前 100 幅图像, 卷积后的特征图:

Feature map after pooling



NORB 数据集前 100 幅图像, 池化后的特征图:

6.4.6.2.11 参考文献

6.4.6.3 模糊极速学习机

Takagi–Sugeno–Kang (TSK) fuzzy inference system (FIS)

[7]

[6]

在线序列模糊极速学习机 [4]

正则极速学习神经模糊自适应算法 [8]

6.4.6.4 基于局部感受野的模糊极速学习机

6.4.6.4.1 简介

现实世界中, 每个训练样本的重要性不一, 通常一些训练样本比另外一些更重要, 我们希望有意义的样本被正确分类, 而不关心噪声样本是否被错分 [1]. 在基于局部感受野的极速学习机 (Local Receptive Fields based Extreme Learning Machine, ELM-LRF) 上引入模糊性, 使得模型对噪声更加鲁邦, 得到基于局部感受野的模糊极速学习机 (Local Receptive Fields based Fuzzy Extreme Learning Machine, FELM-LRF).

提示:

- 模糊集合的概念参见章节模糊集合 (页 123)
 - 有关基于局部感受野的极速学习机参见章节基于局部感受野的极速学习机 (页 358).
-

6.4.6.4.2 FELM-LRF 原理

对于分类问题, 假设有训练样本集 $\tilde{\mathbb{S}} = \{(\mathbf{x}_i, \mathbf{y}_i, u_i)\}_{i=1}^N$, 其中, N 为样本数目, $u_i \in \mathbb{R}$ 为样本 $\mathbf{x}_i \in \mathbb{R}^n$ 对应的隶属度, \mathbf{y}_i 为 \mathbf{x}_i 对应的标签, 在损失函数中引入隶属度, 从而有 FELM-LRF 的优化问题

$$\min \|\boldsymbol{\beta}\|_p^{\sigma_1} + C \|\mathbf{U}(\mathbf{H}\boldsymbol{\beta} - \mathbf{T})\|_q^{\sigma_2} \quad (6.23)$$

其中, $\sigma_1 > 0, \sigma_2 > 0$, $p, q > 0$, $\mathbf{U} = \text{diag}(u_1, u_2, \dots, u_N) \in \mathbb{R}^{N \times N}$ 为由所有训练样本的隶属度组成的对角矩阵, $\boldsymbol{\beta} \in \mathbb{R}^{L \times m}$ 输出层权重矩阵, $\mathbf{T} \in \mathbb{R}^{N \times m}$ 为由样本标签组成的 one-hot 矩阵, 满足

$$\mathbf{T}_{ij} = \begin{cases} 1, & j = \mathbf{y}_i \\ 0, & j \neq \mathbf{y}_i \end{cases}.$$

记 $\mathbf{H}_F = \mathbf{U}\mathbf{H}, \mathbf{T}_F = \mathbf{U}\mathbf{T}$, 则 FELM-LRF 的优化问题式.6.23 变为

$$\min \|\boldsymbol{\beta}\|_p^{\sigma_1} + C \|(\mathbf{H}_F\boldsymbol{\beta} - \mathbf{T}_F)\|_q^{\sigma_2} \quad (6.24)$$

当 $\sigma_1 = \sigma_2 = p = q = 2$ 时, 容易求得式.6.24 的解为

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}_F^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}_F \mathbf{H}_F^T \right)^{-1} \mathbf{T}_F, & \text{if } N \leq L \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}_F^T \mathbf{H}_F \right)^{-1} \mathbf{H}_F^T \mathbf{T}_F, & \text{if } N \geq L \end{cases} \quad (6.25)$$

其中, $\mathbf{H}_F = \mathbf{U}\mathbf{H}, \mathbf{T}_F = \mathbf{U}\mathbf{T}$.

6.4.6.4.3 隶属函数选择

关于隶属函数选择可以参考章节 [隶属函数选择](#) (页 356)

6.4.6.4.4 实验及结果

6.4.6.5 在线极速学习机

6.4.6.5.1 简介

6.4.6.5.2 OSELM 原理

下面介绍 [在线极速学习机](#) (Oniline Sequential Extreme Learning Machine, OS-ELM) 原理.

考虑如下 ELM 问题

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$$

其最小二乘解为

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}$$

如果 $\mathbf{H}^T \mathbf{H}$ 趋于奇异, 可以选择较小的隐层神经元节点数 L , 或者增加样本数量.

给定初始训练集 $\mathbb{S}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, 其中, N_0 为样本数, 且 $N_0 > L$. 从而有初始解

$$\boldsymbol{\beta}_0 = \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0, \quad \mathbf{K}_0 = \mathbf{H}_0^T \mathbf{H}_0$$

假如又可以获得训练集 $\mathbb{S}_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$, 其中 N_1 为新样本数目, 问题变为最小化

$$\left\| \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \right\|$$

其解为:

$$\boldsymbol{\beta}_1 = \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}, \quad \mathbf{K}_1 = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}$$

又因为

$$\begin{aligned} \mathbf{K}_1 &= \begin{bmatrix} \mathbf{H}_0^T & \mathbf{H}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1 \end{aligned}$$

且

$$\begin{aligned} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} &= \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= (\mathbf{K}_1 - \mathbf{H}_1^T \mathbf{H}_1) \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_1 \boldsymbol{\beta}_0 - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \end{aligned}$$

所以, $\boldsymbol{\beta}_1$ 可以表示成 $\boldsymbol{\beta}_0$ 的函数

$$\begin{aligned} \boldsymbol{\beta}_1 &= \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\ &= \mathbf{K}_1^{-1} (\mathbf{K}_1 \boldsymbol{\beta}_0 - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1) \\ &= \boldsymbol{\beta}_0 + \mathbf{K}_1^{-1} \mathbf{H}_1^T (\mathbf{T}_1 - \mathbf{H}_1 \boldsymbol{\beta}_0) \end{aligned}$$

其中, $\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1$. 依次类推, 可以得到权重 $\boldsymbol{\beta}$ 的迭代更新公式

$$\begin{aligned} \mathbf{K}_{k+1} &= \mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{K}_{k+1}^{-1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}_k), \end{aligned} \tag{6.26}$$

其中, $\mathbf{K}_k^{-1} \in \mathbb{R}^{L \times L}$, $\mathbf{H}_k \in \mathbb{R}^{N_k \times L}$, $\boldsymbol{\beta}_k \in \mathbb{R}^{L \times m}$, $\mathbf{T}_k \in \mathbb{R}^{N_k \times m}$ $k = 1, 2, \dots$. 根据输出权重的迭代更新公式.6.26 可知, 每次更新权重都需要计算一次一个 $\mathbf{K}_{k+1}^{-1} \in \mathbb{R}^{L \times L}$ 的逆.

由 Sherman-Morrison-Woodbury 等式.2.4 (参见 part-MatrixTheory 之常用总结 (页 93) 小结) 知

$$\begin{aligned} \mathbf{K}_{k+1}^{-1} &= (\mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1})^{-1} \\ &= \mathbf{K}_k^{-1} - \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{K}_k^{-1}. \end{aligned}$$

若记 $\mathbf{P}_{k+1} = \mathbf{K}_{k+1}^{-1}$, 则有如下迭代公式

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}_k) \end{aligned} \tag{6.27}$$

其中, $\mathbf{I} \in \mathbb{R}^{N_k \times N_k}$ 为单位矩阵, 当 $N_k < L$ 时, 式.6.27 可以减小计算量.

若样本以 one-by-one 的形式输入 OSELM, 则更新公式.6.27 简化为

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{h}_{k+1} \mathbf{h}_{k+1}^T \mathbf{P}_k}{1 + \mathbf{h}_{k+1}^T \mathbf{P}_k \mathbf{h}_{k+1}} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{h}_{k+1} (\mathbf{t}_{k+1}^T - \mathbf{h}_{k+1}^T \boldsymbol{\beta}_k) \end{aligned} \quad (6.28)$$

上述在线训练算法总结如下:

小技巧: 输入: 样本序列 $\mathbb{S}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, $\mathbb{S}_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$, \dots , $\mathbb{S}_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=\sum_{j=1}^k N_j+1}^{\sum_{j=1}^{k+1} N_j}$, $k = 1, 2, \dots, K$.

输出: 更新输出层权重 $\boldsymbol{\beta}_K$

1. 初始训练 ($k = 0, N_0 > L$):

计算初始样本集 \mathbb{S}_0 的隐藏层输出 \mathbf{H}_0 , 估计初始权重 $\boldsymbol{\beta}_0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$, 其中 $\mathbf{P}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$, $\mathbf{T}_0 = [\mathbf{t}_1, \dots, \mathbf{t}_{N_0}]^T$.

2. 序列学习 ($k = 1, 2, \dots, K, N_k$ 为样本数):

- 若 $N_k > 1$, 计算第 $k+1$ 个样本集 \mathbb{S}_{k+1} 的隐藏层输出 \mathbf{H}_{k+1} , 并根据式.6.27 更新权重;
- 若 $N_k = 1$, 计算第 $k+1$ 个样本集 \mathbb{S}_{k+1} 的隐藏层输出 \mathbf{h}_{k+1} , 则根据式.6.31 更新权重;

当 $k = K$ 时停止训练, 并输出权重 $\boldsymbol{\beta}_K$.

6.4.6.6 基于局部感受野的在线极速学习机

6.4.6.6.1 简介

一般来说, 隐藏层节点要足够多, ELM 算法如果要取得较好的效果, 一方面由于训练数据集往往比较大, 不能一次性送入 ELM 进行训练, 另一方面, 现实生活中, 数据往往不能一次性获得.

提示:

- 在线极速学习机参见章节在线极速学习机 (页 385)
- 有关基于局部感受野的极速学习机参见章节基于局部感受野的极速学习机 (页 358).

6.4.6.6.2 OSELM-LRF 原理

下面介绍 基于局部感受野的在线极速学习机 (Local Receptive Fields based Oniline Sequential Extreme Learning Machine, OSELM-LRF) 原理.

考虑如下 ELM 问题

$$\|\boldsymbol{\beta}\|_2^2 + C \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2^2$$

其最小二乘解为

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}^T \left(\frac{I}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}, & \text{if } N \leq L \\ \left(\frac{I}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}, & \text{if } N > L \end{cases}$$

考虑解 $(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}$.

提示: 虽然初始样本数 $N_0 < L$, 但随着样本源源不断地到来, 最终会有 $\sum_{j=1}^k N_k > L$.

给定初始训练集 $\mathbb{S}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, 其中, N_0 为样本数, 且 $N_0 > L$. 从而有初始解

$$\boldsymbol{\beta}_0 = \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0, \quad \mathbf{K}_0 = \frac{\mathbf{I}}{C} + \mathbf{H}_0^T \mathbf{H}_0$$

假如又可以获得训练集 $\mathbb{S}_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$, 其中 N_1 为新样本数目, 问题变为最小化

$$\|\boldsymbol{\beta}\|_2^2 + C \left\| \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \right\|_2^2$$

其解为:

$$\boldsymbol{\beta}_1 = \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix},$$

其中,

$$\begin{aligned} \mathbf{K}_1 &= \frac{\mathbf{I}}{C} + \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \frac{\mathbf{I}}{C} + \mathbf{H}_0^T \mathbf{H}_0 + \mathbf{H}_1^T \mathbf{H}_1 \\ &= \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1 \end{aligned}$$

又

$$\begin{aligned} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} &= \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= (\mathbf{K}_1 - \mathbf{H}_1^T \mathbf{H}_1) \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_1 \boldsymbol{\beta}_0 - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1 \end{aligned}$$

所以, $\boldsymbol{\beta}_1$ 可以表示成 $\boldsymbol{\beta}_0$ 的函数

$$\begin{aligned} \boldsymbol{\beta}_1 &= \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\ &= \mathbf{K}_1^{-1} (\mathbf{K}_1 \boldsymbol{\beta}_0 - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}_0 + \mathbf{H}_1^T \mathbf{T}_1) \\ &= \boldsymbol{\beta}_0 + \mathbf{K}_1^{-1} \mathbf{H}_1^T (\mathbf{T}_1 - \mathbf{H}_1 \boldsymbol{\beta}_0) \end{aligned}$$

其中, $\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1$. 依次类推, 可以得到权重 $\boldsymbol{\beta}$ 的迭代更新公式

$$\begin{aligned} \mathbf{K}_{k+1} &= \mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{K}_{k+1}^{-1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}_k), \end{aligned} \tag{6.29}$$

可见, 含权重正则的 OSELM 问题与不含权重正则的 OSELM 问题的权重迭代更新公式相同, 区别在于初始化权重 $\boldsymbol{\beta}_0$ 的计算. 根据[在线极速学习机](#) (页 385) 小节推导, 可知含权重正则的基于局部感受野的在线

极速学习机的权重更新公式为

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}_k)\end{aligned}\quad (6.30)$$

其中, $\mathbf{I} \in \mathbb{R}^{N_k \times N_k}$ 为单位矩阵, 当 $N_k < L$ 时, 式 6.27 可以减小计算量.

若样本以 one-by-one 的形式输入 OSELM, 则更新公式 6.30 简化为

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{h}_{k+1} \mathbf{h}_{k+1}^T \mathbf{P}_k}{1 + \mathbf{h}_{k+1}^T \mathbf{P}_k \mathbf{h}_{k+1}} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{h}_{k+1} (\mathbf{t}_{k+1}^T - \mathbf{h}_{k+1}^T \boldsymbol{\beta}_k)\end{aligned}\quad (6.31)$$

基于局部感受野的在线极速学习机训练算法总结如下:

注解: 输入: 样本序列 $\mathbb{S}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, $\mathbb{S}_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$, \dots , $\mathbb{S}_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=\sum_{j=1}^k N_j+1}^{\sum_{j=1}^{k+1} N_j}$, $k = 1, 2, \dots, K$, 最大训练代数 O .

输出: 更新输出层权重 $\boldsymbol{\beta}_O$

1. 初始训练 ($k = 0, N_0 > L$):

计算初始样本集 \mathbb{S}_0 的隐藏层输出 \mathbf{H}_0 , 估计初始权重 $\boldsymbol{\beta}_0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$, 其中 $\mathbf{P}_0 = (\frac{\mathbf{I}}{C} + \mathbf{H}_0^T \mathbf{H}_0)^{-1}$, $\mathbf{T}_0 = [\mathbf{t}_1, \dots, \mathbf{t}_{N_0}]^T$.

2. 序列学习 ($k = 1, 2, \dots, K, N_k$ 为样本数):

- 若 $N_k > 1$, 计算第 $k+1$ 个样本集 \mathbb{S}_{k+1} 的隐藏层输出 $\mathbf{H}_{k+1} \in \mathbb{R}^{N_k \times L}$, 并根据下式更新权重;

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}_k)\end{aligned}$$

- 若 $N_k = 1$, 计算第 $k+1$ 个样本集 \mathbb{S}_{k+1} 的隐藏层输出 $\mathbf{h}_{k+1} \in \mathbb{R}^{L \times 1}$, 则根据下式更新权重;

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{h}_{k+1} \mathbf{h}_{k+1}^T \mathbf{P}_k}{1 + \mathbf{h}_{k+1}^T \mathbf{P}_k \mathbf{h}_{k+1}} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \mathbf{P}_{k+1} \mathbf{h}_{k+1} (\mathbf{t}_{k+1}^T - \mathbf{h}_{k+1}^T \boldsymbol{\beta}_k)\end{aligned}$$

当 $k = K$ 时完成一代训练, 并输出权重 $\boldsymbol{\beta}_o = \boldsymbol{\beta}_K$, $o = o + 1$.

3. 随机打乱样本集 $\mathbb{S}_k, k = 1, 2, \dots, K$ 的顺序, 并重复步骤 2 至最大训练代数 $o = O$ 时停止.

6.4.6.7 参考文献

6.4.6.8 名词术语

Extreme Learning Machine 极速学习机 ([Extreme Learning Machine](http://en.volupedia.org/wiki/Extreme_learning_machine) (http://en.volupedia.org/wiki/Extreme_learning_machine))

6.4.7 基于能量的神经网络

6.4.7.1 Hopfield 神经网络

- [Discrete Hopfield Network](http://neupy.com/2015/09/20/discrete_hopfield_network.html)
- [hopfield-mnist](<https://github.com/kencyke/hopfield-mnist>): A scikit-learn implementation of hopfield network for MNIST
- [hopfield](<https://github.com/unixpickle/hopfield>): Hopfield networks in TensorFlow

6.4.7.2 玻尔兹曼机与受限玻尔兹曼机

[Deep Generative Models](https://iridescent.ink/tutorials/pdf/DeepGenerativeModels.pdf) (<https://iridescent.ink/tutorials/pdf/DeepGenerativeModels.pdf>)

6.4.8 卷积神经网络

6.4.8.1 经典卷积神经网络

卷积神经网络 (Convolutional Neural Network) [2] , 图像超分辨 [3]

6.4.8.1.1 卷积神经网络举例

6.4.8.1.1.1 LeNet

Lecun 等人于 1998 年提出 LeNet [2] , 网络结构如 图 6.39 所示

6.4.8.1.1.2 AlexNet

Alex 等人于 2012 年提出 AlexNet [4] , 网络结构如 图 6.40 所示

在 AlexNet 中, 使用了以下改进操作:

1. 使用 ReLU 作为激活函数, 并验证在较深的网络超过了 Sigmoid , 成功解决了 Sigmoid 在网络较深时的梯度弥散问题.
2. 训练时使用 Dropout 正则方法, 以避免模型过拟合. Dropout 虽有单独的论文论述, 但是 AlexNet 将其实用化, 通过实践证实了它的效果, 在 AlexNet 中主要是最后几个全连接层使用了 Dropout.
3. 使用重叠的最大池化, 此前 CNN 中普遍使用平均池化, AlexNet 全部使用最大池化, 避免平均池化的模糊化效果. 并且 AlexNet 中提出步长尺寸要比池化核的尺寸小, 这样池化层的输出之间会有重叠和覆盖, 提升了特征的丰富性.
4. 提出局部响应归一化单元, 模拟大脑中神经元的侧抑制现象.

在特定的几层经过 ReLU 非线性激活函数后, 使用了局部响应归一化 (Local Response Normalization, LRN) 单元:

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta \quad (6.32)$$

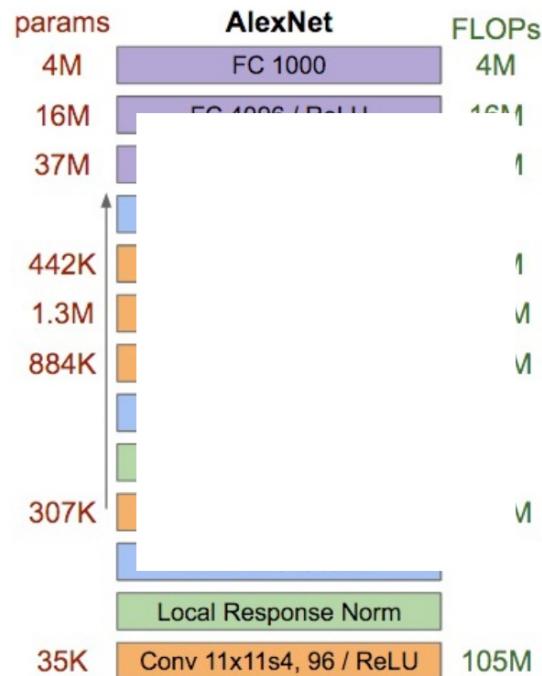


图 6.39: LeNet 结构.

LeNet 结构.

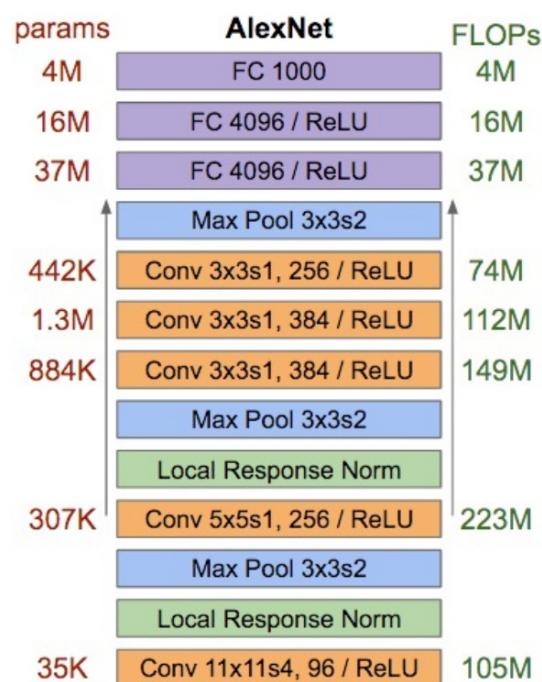


图 6.40: AlexNet 结构.

AlexNet 结构.

其中, N 为当前层中卷积核的个数, k, n, α, β 为超参数, 并使用验证集来确定其值, 文中取值为 $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$, LRN 模拟了大脑中的侧抑制现象, 对局部神经元进行激励与抑制, 使得其中响应比较大的神经元响应值变得更大, 并抑制其他响应较小的神经元, 增强了模型的泛化能力.

tensorflow 下, 主要实现代码:

6.4.8.1.1.3 VGGNet

[5]

6.4.8.1.1.4 GoogleNet

[6]

6.4.9 递归神经网络

6.4.9.1 递归神经网络

递归神经网络 (Recursive Neural Network)

6.4.10 生成式神经网络

6.4.10.1 生成式神经网络

生成式神经网络 (Generative Neural Network) [7]

6.4.11 残差神经网络

6.4.11.1 残差神经网络

残差神经网络 (Residual Neural Network)

6.4.12 分形神经网络

6.4.12.1 分形神经网络

分形神经网络 (Fractal Neural Network)

6.4.13 随机神经网络

6.4.13.1 随机神经网络

随机神经网络 (Stochastic Neural Network)

Random synaptic feedback weights support error backpropagation for deep learning

6.4.14 复杂度分析

6.4.14.1 衡量指标

通常用参数量 (params)、乘累加量 (Fused multiply-add, Madd)、浮点数运算量 (FLOPs) 以及内存读写量来衡量网络的运算复杂度. 有关 FLOPS 参见 Section-ComputationalComplexityPerformance 小节.

6.4.14.1.1 运算量计算

6.4.14.1.1.1 卷积层

卷积层的 FLOPs 可由下式计算 [1]

$$\text{FLOPs} = 2HW(C_{in}K^2 + 1)C_{out} \quad (6.33)$$

其中, H, W 分别为输入特征图的高和宽, C_{in}, C_{out} 分别为输入特征图和输出特征图的通道数, K 为卷积核的大小.

6.4.14.1.1.2 全连接层

$$\text{FLOPs} = (2I - 1)O$$

其中, I, O 分别为全连接层的输入与输出神经元节点数.

6.4.14.2 实例分析

6.4.14.2.1 SSD300 目标检测网络分析

SSD300 目标检测网络的计算量主要在卷积层, 下表给出了 SSD300 网络的各层计算量分析结果, 包括输入输出大小, 参数量 (params), 乘累加量 (Fused multiply-add, Madd), 浮点数运算量 (FLOPs) 以及内存读写量.

number	module name	input shape	output shape	params	memory(MB)	MAdd	Flops
0	vgg.0	3 300 300	64 300 300	1792.0	21.97	311,040,000.0	161,280,000.0
1	vgg.1	64 300 300	64 300 300	0.0	21.97	5,760,000.0	5,760,000.0

表 6.4 - 续上页

number	module name	input shape	output shape	params	memory(MB)	MAdd	Flops
2	vgg.2	64 300 300	64 300 300	36928.0	21.97	6,635,520,000.0	3,323,520,000.
3	vgg.3	64 300 300	64 300 300	0.0	21.97	5,760,000.0	5,760,000.0
4	vgg.4	64 300 300	64 150 150	0.0	5.49	4,320,000.0	5,760,000.0
5	vgg.5	64 150 150	128 150 150	73856.0	10.99	3,317,760,000.0	1,661,760,000.
6	vgg.6	128 150 150	128 150 150	0.0	10.99	2,880,000.0	2,880,000.0
7	vgg.7	128 150 150	128 150 150	147584.0	10.99	6,635,520,000.0	3,320,640,000.
8	vgg.8	128 150 150	128 150 150	0.0	10.99	2,880,000.0	2,880,000.0
9	vgg.9	128 150 150	128 75 75	0.0	2.75	2,160,000.0	2,880,000.0
10	vgg.10	128 75 75	256 75 75	295168.0	5.49	3,317,760,000.0	1,660,320,000.
11	vgg.11	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0
12	vgg.12	256 75 75	256 75 75	590080.0	5.49	6,635,520,000.0	3,319,200,000.
13	vgg.13	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0
14	vgg.14	256 75 75	256 75 75	590080.0	5.49	6,635,520,000.0	3,319,200,000.
15	vgg.15	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0
16	vgg.16	256 75 75	256 38 38	0.0	1.41	1,108,992.0	1,440,000.0
17	vgg.17	256 38 38	512 38 38	1180160.0	2.82	3,406,823,424.0	1,704,151,040.
18	vgg.18	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0
19	vgg.19	512 38 38	512 38 38	2359808.0	2.82	6,813,646,848.0	3,407,562,752.
20	vgg.20	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0
21	vgg.21	512 38 38	512 38 38	2359808.0	2.82	6,813,646,848.0	3,407,562,752.
22	vgg.22	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0
23	vgg.23	512 38 38	512 19 19	0.0	0.71	554,496.0	739,328.0
24	vgg.24	512 19 19	512 19 19	2359808.0	0.71	1,703,411,712.0	851,890,688.0
25	vgg.25	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0
26	vgg.26	512 19 19	512 19 19	2359808.0	0.71	1,703,411,712.0	851,890,688.0
27	vgg.27	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0
28	vgg.28	512 19 19	512 19 19	2359808.0	0.71	1,703,411,712.0	851,890,688.0
29	vgg.29	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0
30	vgg.30	512 19 19	512 19 19	0.0	0.71	1,478,656.0	184,832.0
31	vgg.31	512 19 19	1024 19 19	4719616.0	1.41	3,406,823,424.0	1,703,781,376.
32	vgg.32	1024 19 19	1024 19 19	0.0	1.41	369,664.0	369,664.0
33	vgg.33	1024 19 19	1024 19 19	1049600.0	1.41	757,071,872.0	378,905,600.0
34	vgg.34	1024 19 19	1024 19 19	0.0	1.41	369,664.0	369,664.0
35	L2Norm	512 38 38	512 38 38	512.0	2.82	0.0	0.0
36	extras.0	1024 19 19	256 19 19	262400.0	0.35	189,267,968.0	94,726,400.0
37	extras.1	256 19 19	512 10 10	1180160.0	0.20	235,929,600.0	118,016,000.0
38	extras.2	512 10 10	128 10 10	65664.0	0.05	13,107,200.0	6,566,400.0
39	extras.3	128 10 10	256 5 5	295168.0	0.02	14,745,600.0	7,379,200.0
40	extras.4	256 5 5	128 5 5	32896.0	0.01	1,638,400.0	822,400.0
41	extras.5	128 5 5	256 3 3	295168.0	0.01	5,308,416.0	2,656,512.0
42	extras.6	256 3 3	128 3 3	32896.0	0.00	589,824.0	296,064.0

表 6.4 - 续上页

number	module name	input shape	output shape	params	memory(MB)	MAdd	Flops
43	extras.7	128 3 3	256 1 1	295168.0	0.00	589,824.0	295,168.0
44	loc.0	512 38 38	16 38 38	73744.0	0.09	212,926,464.0	106,486,336.0
45	loc.1	1024 19 19	24 19 19	221208.0	0.03	159,694,848.0	79,856,088.0
46	loc.2	512 10 10	24 10 10	110616.0	0.01	22,118,400.0	11,061,600.0
47	loc.3	256 5 5	24 5 5	55320.0	0.00	2,764,800.0	1,383,000.0
48	loc.4	256 3 3	16 3 3	36880.0	0.00	663,552.0	331,920.0
49	loc.5	256 1 1	16 1 1	36880.0	0.00	73,728.0	36,880.0
50	conf.0	512 38 38	84 38 38	387156.0	0.46	1,117,863,936.0	559,053,264.0
51	conf.1	1024 19 19	126 19 19	1161342.0	0.17	838,397,952.0	419,244,462.0
52	conf.2	512 10 10	126 10 10	580734.0	0.05	116,121,600.0	58,073,400.0
53	conf.3	256 5 5	126 5 5	290430.0	0.01	14,515,200.0	7,260,750.0
54	conf.4	256 3 3	84 3 3	193620.0	0.00	3,483,648.0	1,742,580.0
55	conf.5	256 1 1	84 1 1	193620.0	0.00	387,072.0	193,620.0
56	softmax	8732 21	8732 21	0.0	0.70	550,115.0	0.0
total				26285486.0	207.65	62,782,359,651.0	31,435,153,590

SSD300 网络总的计算量见下表, 可见网络含有 26,285,486 参数, 内存占用为 207.65MB, 总乘累加次数为 62.78G, 总的浮点数运算次数为 31.44GFLOPs.

Total params	Total memory	Total MAdd	Total FLOPs	Total MemR+W
26,285,486	207.65MB	62.78GMAdd	31.44GFlops	517.64MB

6.4.15 参考文献

6.4.16 名词术语

Neural Network 神经网络 ([Neural Network](http://en.volupedia.org/wiki/Artificial_neural_network) (http://en.volupedia.org/wiki/Artificial_neural_network))

6.5 深度学习

6.5.1 简介

6.5.2 名词术语

Deep Learning 深度学习

6.6 模糊神经系统

6.6.1 简介

有关模糊的基本概念, 参见[模糊数学](#) (页 123) .

6.6.2 神经模糊

6.6.2.1 神经模糊基础

提示:

- 模糊数学的概念参照[模糊数学](#) (页 123)
 - 模糊集合的概念参照[模糊集合](#) (页 123)
 - 模糊推理的概念参照[模糊推理](#) (页 132)
 - 神经生物学的概念参照[volume-BasicNeurobiology](#)
 - 神经网络的概念参照[神经网络](#) (页 333)
-

6.6.3 经典模糊神经网络

6.6.3.1 自适应神经模糊推理系统

自适应神经模糊推理系统 (Adaptive Neuro-Fuzzy Inference System, ANFIS) 也称 自适应网络模糊推理系统 (Adaptive Network-based Fuzzy Inference System, ANFIS) 由 Jang 于 1993 年提出 [1] , ANFIS 是基于[Takagi-Sugeno 模糊系统](#) (页 134) 和神经网络的自适应模糊推理系统. ANFIS 将神经网络与模糊推理相结合, 从而具备自学习与模糊推理能力. Isik 等人 [2] 于 1997 年提出递归神经模糊系统 (Recurrent neuro-fuzzy systems), Petr 等人 [3] 于 2005 年提出具备在线学习能力的 ANFIS.

警告: ANFIS 与 Takagi-Sugeno 不同点在于隶属函数和权重通过优化算法学习得到.

6.6.3.2 基于 ELM 的模糊推理系统

6.6.3.2.1 SLFN 与 ANFIS 的等效性

参见 [4]

单层前馈神经网络 (Single-Layer Feedforward Network, SLFN)

6.6.4 参考文献

6.6.5 名词术语

Fuzzy System 模糊系统 ([Fuzzy System](http://en.volupedia.org/wiki/Fuzzy_system) (http://en.volupedia.org/wiki/Fuzzy_system))

6.7 课程学习

6.7.1 引言

6.7.1.1 什么是课程学习

在 Curriculum Learning 中，课程由先前知识预先确定，并在此之后保持固定。因此，这种方法在很大程度上依赖于先前知识的质量而忽略了关于学习者的反馈。

6.7.2 名词术语

Curriculum Learning 课程学习 ([Curriculum Learning](http://en.volupedia.org/wiki/) (<http://en.volupedia.org/wiki/>) , CL)

6.8 自步学习

6.8.1 引言

6.8.1.1 什么是自步学习

在 Self-paced Learning 中，课程是动态决定的，以适应学习者的学习节奏。但是，Self-paced Learning 无法处理先前的知识，使其容易过度拟合。

Curriculum Learning 和 Self-paced Learning 代表了最近提出的学习制度，其受到人类和动物学习过程的启发，这些学习过程逐渐从训练中的简单复杂样本开始。这两种方法具有相似的概念学习范式，但在具体的学习方案上有所不同。

注解：与模糊机制类似

6.8.2 自步学习基础

6.8.3 自步函数

6.8.3.1 自步函数设计准则

自步函数 $g(v_i, \lambda)$ 一般需要满足以下条件 [2]:

1. $g(v_i, \lambda)$ 在区间 $[0, 1]$ 上是凸函数, 以此保证 v_i^* 是唯一的;
2. $v_i^*(l_i, \lambda)$ 关于 l_i 是单调递减的, 这样可以引导模型选择较容易地样本;
3. $v_i^*(l_i, \lambda)$ 关于 λ 是单调递增的, 大的 λ 可以容忍更大的损失, 可以逐渐把复杂样例包含进来.

6.8.3.2 常用自步函数

文献 [2] 中总结了一些常见的自步函数, 设有 N 个样本, 样本容易度向量记为 $\mathbf{v} \in [0, 1]^N$, 自步函数记为 $f(\mathbf{v}; \lambda)$, 其中, $= \frac{1}{k}$, 最优解记为 $\mathbf{v}^* = [v_1^*, v_2^*, \dots, v_N^*]$.

6.8.3.2.1 二值加权

二值加权自步函数的表达式为

$$f(\mathbf{v}, k) == -\|\mathbf{v}\|_1 = -\sum_{n=1}^N v_n \quad (6.34)$$

其最优解为

$$v_n^* = \begin{cases} 1, & l_n < \lambda \\ 0, & l_n \geq \lambda \end{cases} \quad (6.35)$$

6.8.3.2.2 线性加权

线性加权自步函数的表达式为

$$f(\mathbf{v}, \lambda) = \lambda \left(\frac{1}{2} \|\mathbf{v}\|_2^2 - \sum_{n=1}^N v_n \right) \quad (6.36)$$

其最优解为

$$v_n^* = \max\{1 - l_n/\lambda, 0\} \quad (6.37)$$

6.8.3.2.3 对数加权

对数加权自步函数的表达式为

$$f(\mathbf{v}, \lambda) = \sum_{n=1}^N \left(\zeta v_n - \frac{\zeta^{v_n}}{\log \zeta} \right) \quad (6.38)$$

其中, $\zeta = 1 - \lambda, 0 < \lambda < 1$

其最优解为

$$v_n^* = \begin{cases} 0, & l_n \geq \lambda \\ \log(l_n + \zeta) / \log \xi, & l_n < \lambda \end{cases} \quad (6.39)$$

6.8.3.2.4 混合加权

混合加权自步函数的表达式为

$$f(\mathbf{v}, \lambda) = -\zeta \sum_{n=1}^N \log(v_n + \zeta / \lambda) \quad (6.40)$$

其中, $\zeta = \frac{1}{k' - k} = \frac{\lambda' \lambda}{\lambda - \lambda'}, k' > k > 0, 0 < \lambda' < \lambda$

其最优解为

$$v_n^* = \begin{cases} 1, & l_n \leq \lambda' \\ 0, & l_n \geq \lambda \\ \zeta / l_n - \zeta / \lambda, & \text{otherwise} \end{cases} \quad (6.41)$$

6.8.3.3 可学习自步函数

6.8.4 参考文献

6.8.5 名词术语

Self-paced Learning 自步学习 ([Self-Paced Learning](http://en.volupedia.org/wiki/) (<http://en.volupedia.org/wiki/>) , SPL)

CHAPTER 7

第七卷雷达信号处理

7.1 引言

雷达 (Radar) 是 无线电探测与测距 (radio detection and ranging) 一词的缩写.

涉及连续波雷达, 调频连续波雷达, 脉冲雷达, 合成孔径雷达, 极化雷达等.

7.1.1 雷达种类

针对高分辨观测应用需求, 研制智能超分辨 SAR 成像演示验证系统, 在传统 SAR 成像理论技术与人工智能等理论技术基础上, 突破现有 SAR 成像设备系统中的波形设计、斑点噪声抑制、压缩成像、感兴趣目标成像与超分辨成像技术, 实现低散斑噪声 SAR 成像、高压缩比 SAR 成像、超分辨 SAR 成像与认知 SAR 成像技术, 为新一代新体制智能化 SAR 成像 (认知 SAR 成像) 系统设计与研制提供理论论证与技术支撑。

7.1.2 推荐资源

- [Radar tutorial](http://www.radartutorial.eu/index.en.html) (<http://www.radartutorial.eu/index.en.html>) : SAR, PolSAR, FMCW, CW, MIMO…
- [电子工程技术教程](https://www.st-andrews.ac.uk/~www_pa/Scots_Guide/intro/electron.htm) (https://www.st-andrews.ac.uk/~www_pa/Scots_Guide/intro/electron.htm) : 模拟数字, 音频, 无线电与相干 (Antennas, Modulation, Radar), 信息与测量 (Sampling, Sensors, Amplifiers, Compression…)
- [海军武器工程概论](https://fas.org/man/dod-101/navy/docs/es310/syllabus.htm) (<https://fas.org/man/dod-101/navy/docs/es310/syllabus.htm>) : RADAR, SONAR

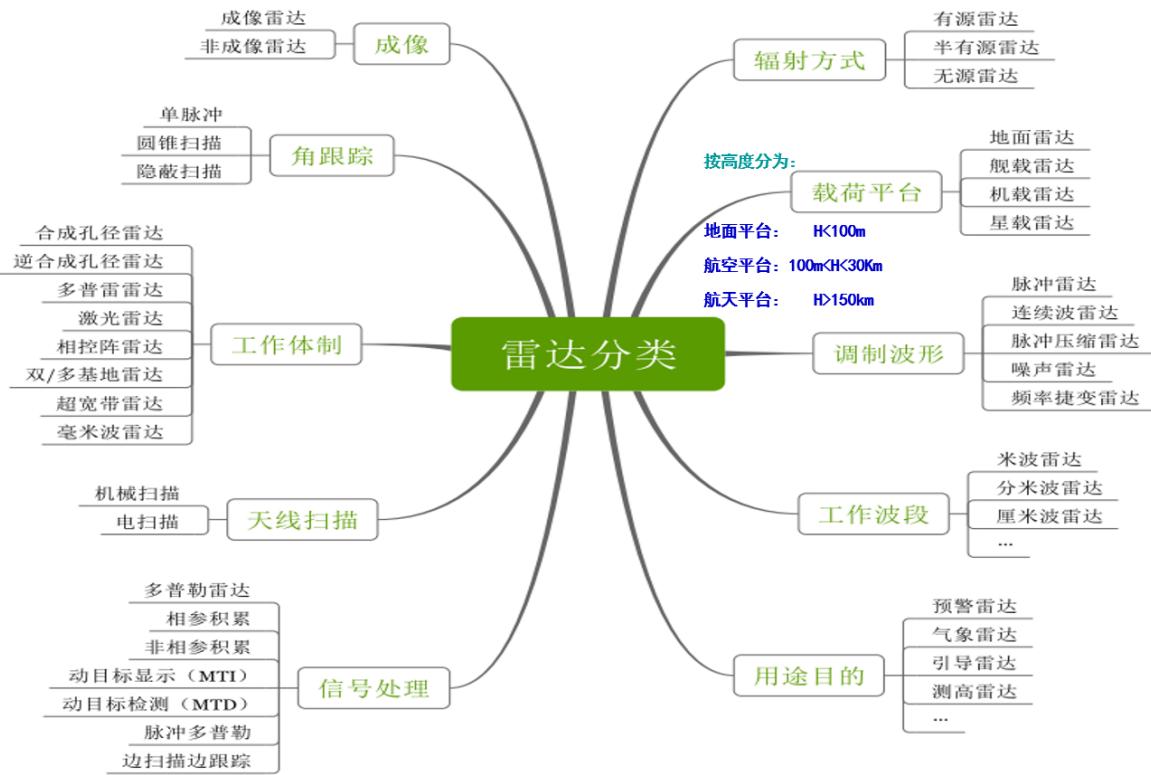


图 7.1: 雷达种类

雷达有多种, 可根据调制波形, 工作模式, 扫描模式等分类

7.2 辐射源信号分析

7.2.1 雷达辐射源信号

7.2.1.1 概述

有关雷达的概念参见 volume-SignalProcessingOfRadar 之 volume-IntroductionSignalProcessingOfRadar 小节.

7.2.1.1.1 雷达辐射源信号描述方式

雷达辐射源描述方式也称辐射源描述字 (Emitter Description Word, EDW) 或脉冲描述字 (Pulse Description Word, PDW) 指雷达辐射源脉冲参数构成的数字化描述字, 常见的类型有

- 脉冲重复间隔 (Pulse Repetition Interval, PRI)
- 脉冲重复频率 (Pulse Repetition Frequency, PRF)
- 脉冲到达时间 (Time Of Arrival, TOA)
- 脉冲到达方向 (Direction Of Arrival, DOA)
- 脉冲宽度 (Pulse Width, PW)
- 脉冲幅度 (Pulse Amplitude, PA)

- 信号载频 (Radio Frequency, RF): 载频一般用 carrier frequency 表示, 但文献里常用 RF 表示
参见文献 [1] p6.

7.2.1.1.1 脉冲重复间隔

常见的脉冲重复间隔形式有: 1) 固定 PRI; 2) 参差 PRI; 3) 抖动 PRI 4) 参差抖动 PRI.

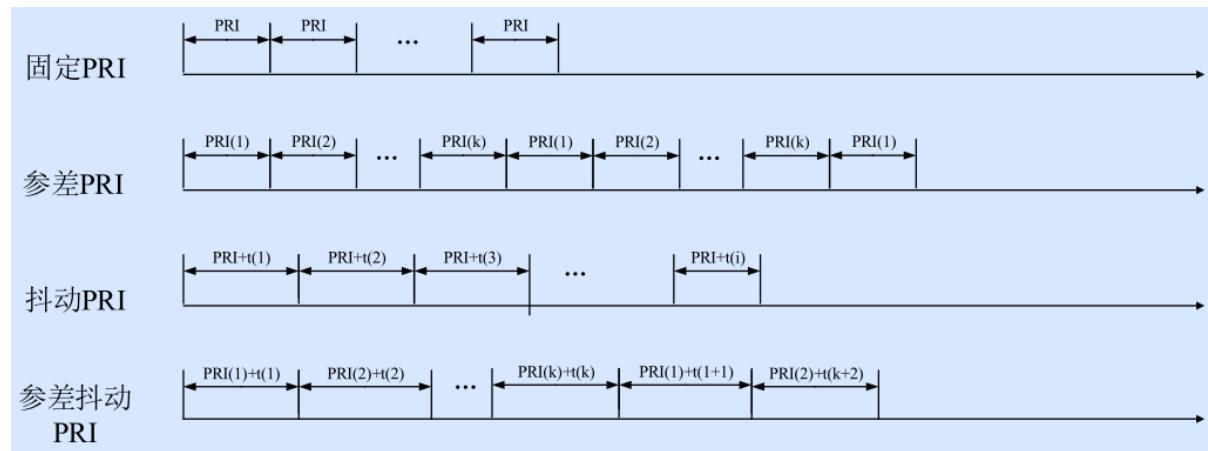


图 7.2: 常见脉冲重复间隔形式 (引自 [1])

7.2.1.2 多雷达辐射源信号模型

7.2.1.2.1 雷达辐射源信号描述方式

雷达侦查侦察系统对截获周围电磁环境中的不同辐射源脉冲信号进行测量从而获取信号的特征参数, 用于后续的分选识别等任务。传统的雷达辐射源信号识别技术主要依赖于截获的雷达辐射源脉冲信号的脉冲描述字 PDW 对雷达辐射源信号进行描述, 它包括了以下几个全脉冲信号的特征参数: 载波频率 (RF), 脉冲到达角 (AOA), 脉冲到达时间 (TOA) 以及通过该参数计算得到的脉冲宽度 (PW) 和脉冲重复频率 (PRF) 等。

脉冲重复间隔 (*Pulse Repetition Interval*, PRI) 又称脉冲重复周期

[1]

7.2.1.3 单雷达信号模型

7.2.1.3.1 发射信号

7.2.1.3.2 接收信号

7.2.1.3.3 噪声信号

7.2.1.3.4 杂波反射信号

7.2.1.3.5 合成信号

7.2.1.4 雷达辐射源信号类型

7.2.1.4.1 概述

7.2.2 辐射源信号仿真

7.2.3 辐射源信号分选

7.2.3.1 引言

空间中充斥着各种各样的雷达信号,在同一时间会有多种雷达信号到达侦查机,这些信号在时域上交叠在一起,雷达信号分选既是从交叠的信号中将各部雷达信号分离出来,其概念示意图如图 7.3 所示.

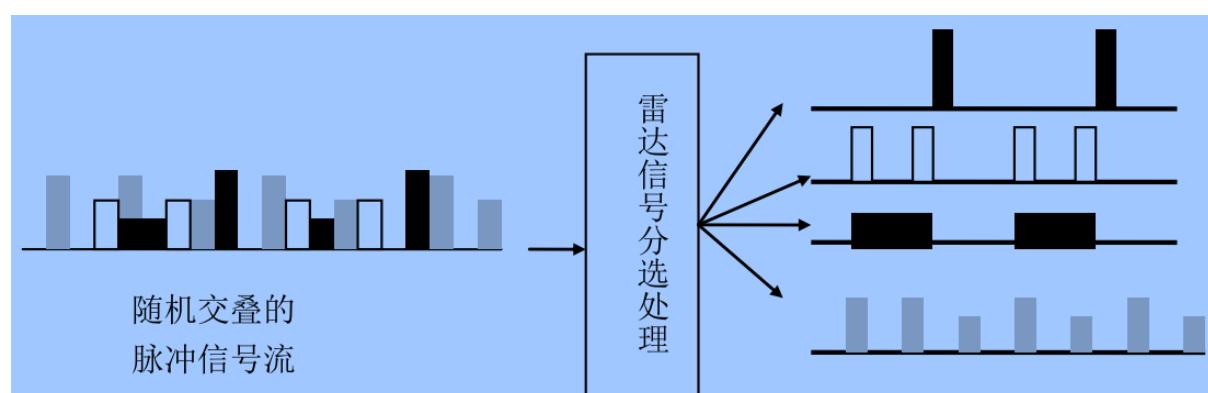


图 7.3: 雷达信号分选概念示意图 (引自 [1])

7.2.3.2 传统方法

7.2.3.2.1 一般集合

7.2.4 参考文献

7.2.5 名词术语

Direction Of Arrival 到达方向 (Direction Of Arrival, DOA) 即波达方向, 指波束到达方向.

Pulse Repetition Interval 脉冲重复间隔 (Pulse Repetition Interval, PRI) 即相邻两个脉冲到达时间差, 可分为固定 PRI, 参差 PRI, 抖动 PRI, 参差抖动 PRI.

Time Of Arrival 到达时间 (Time Of Arrival, TOA) 即波达时间, 指波束到达时间.

7.3 脉冲雷达

7.4 连续波雷达

7.4.1 引言

分为连续波雷达 (单频点, 正弦调制) 和调频连续波雷达 (变频)

7.4.2 系统概述

7.4.2.1 三角波泄漏

7.4.2.1.1 三角波泄漏分析

发射机信号泄漏是 FMCW 毫米波雷达比较严重的问题.

7.4.2.1.1.1 泄漏原因

三角波泄漏的根本原因是由于 **VCO 调频的非线性**, 调频的同时存在着调幅. 这时的发射信号不再是一纯粹的调频波, 而是调幅调频波, 该调幅调频波被单端混频器的 GaAs 肖特基势垒二极管和后面的低通滤波器构成的包络检波电路所检波, 其调幅信号被检波出来, 叠加在有用信号上, 形成泄漏三角波.

7.4.2.1.1.2 解决方法

- 要从根本上抑制三角波泄漏, 应该尽量提高 VCO 的性能, 改进 RF 中 VCO 的线性度和减小扫描功率非线性, 尽量减小调频时存在的寄生调幅.
- 在中频电路中, 可采用固定的高通或带通滤波器来抑制三角波的泄漏. 但因三角波频率和有用中频信号很接近, 高通或带通的实现比较困难. 也可以采用 **频域动态压缩方法**, 它不仅有效抑制三角波泄漏, 而且在一定程度上可以替代 AGC, 简化电路设计, 提高系统灵敏度, 适用于近距离探测. 用由电容和电感组成的近线性巴特沃斯高通滤波器可以实现.
- 数字处理部分, 可采用信号时频分析、短时 FFT 分析和 **多种自适应处理方法** 等来有效抑制三角波泄漏, 提高系统性能.

7.4.2.1.1.3 频域动态压缩方法

源于频域对消思想, 它利用泄漏三角波和有用中频信号在频域中的不同, 抑制三角波信号, 增强有用信号; 同时利用 FMWC 毫米波雷达中频频率(幅度)与距离的关系, 对不同频率的有用信号进行不同的压缩处理, 使得其最终各个频率对应的输出信号幅度相同, 一定程度上达到替代 AGC 的目的. 下图为一种压缩网络与其对应的幅频特性

7.4.2.1.1.4 自适应处理方法

注解: 关于自适应滤波器中的参考信号: [自适应滤波器中的参考信号\(期望信号\)是如何获取的](#) (<https://www.zhihu.com/question/28603557>)

在许多情况下, 一个宽带信号既受到周期性干扰的污染, 又没有外部参考输入可以利用. 此时, 可以直接从原始输入引出, 接入一具有固定延迟的延迟线, 则可得到类似得参考输入支路. 延迟 Δ 必须选得足够长, 以使参考输入中的宽带信号分量和原始输入中的宽带信号分量去相关. 而干扰分量因其周期性, 将仍然是彼此相关的. 经过对消器后, 在系统输出中去掉了原始输入中的可预测部分而留下了其中的不可预测分量.

7.4.2.1.2 三角波泄漏自适应对消验证

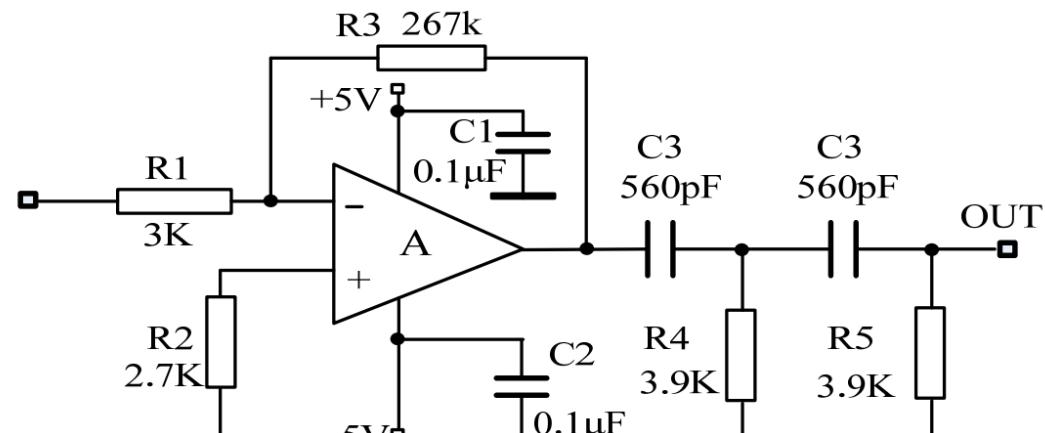
7.4.2.1.2.1 实现方式及结果对比

7.4.2.1.2.2 自己实现

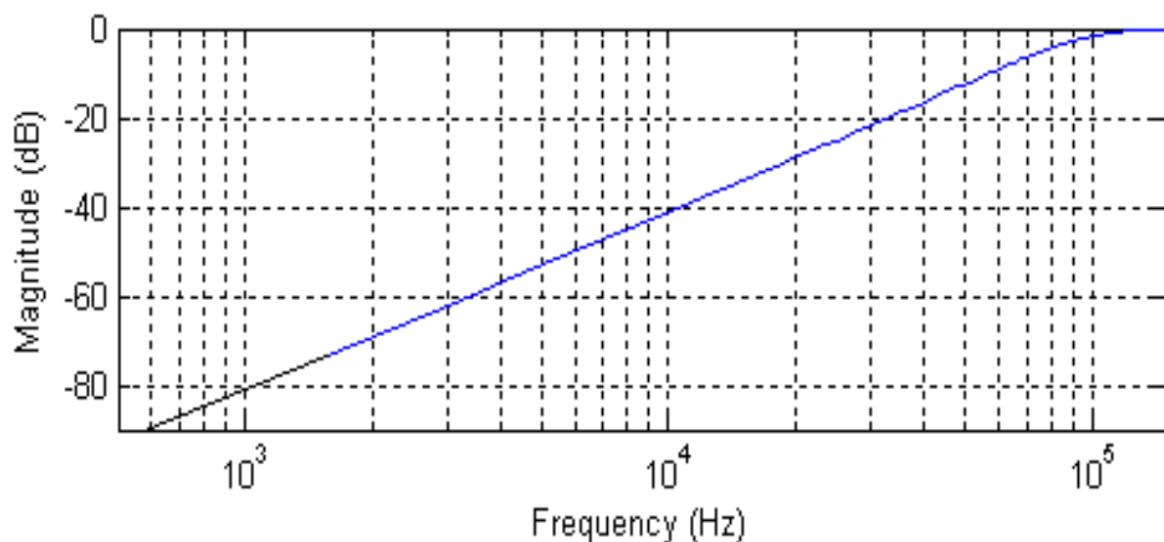
采用最小均方 (Least Mean Square, LMS) 误差准则实现参数调优, 核心代码如下:

```
function [ output, err, w ] = adaptfilterlms( input, desired, param )  
  
LEN = length(input);  
  
% 设计自适应滤波器  
M = param.M; % 滤波器阶数
```

(下页继续)



IF preamplifier and compressive network



Compress network transport characteristic

图 7.4: 频域压缩网络及其幅频特性

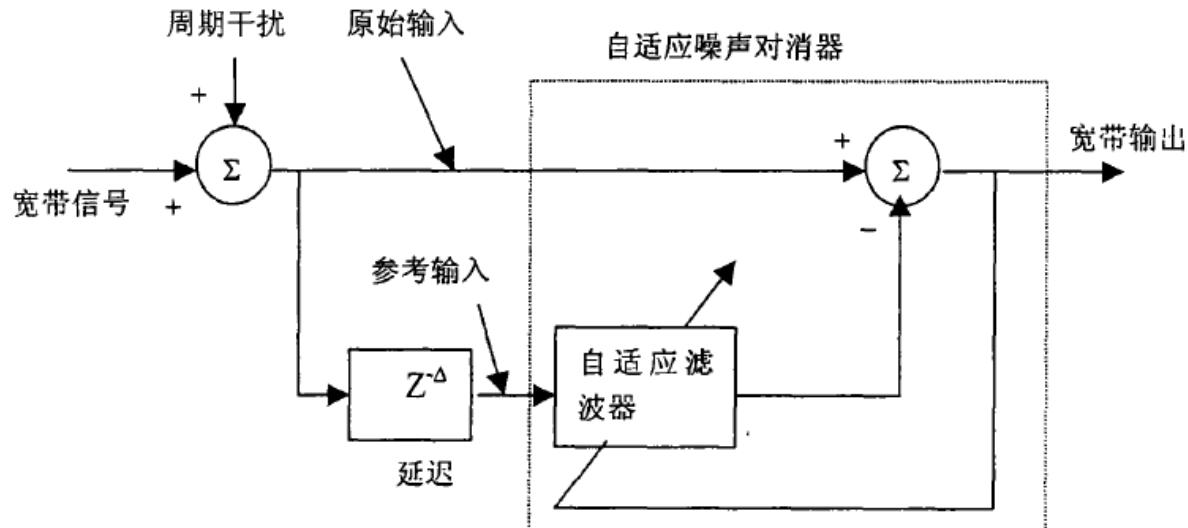


图 6.4 没有外部参考输入的周期干扰对消器原理

图 7.5: 没有参考输入的周期干扰自适应对消

(续上页)

```

u = param.mu; % step

w = ones(M, 1); % weight

output = zeros(LEN, 1);
err = output;

for k = M:LEN
    wn(:, k-M+1) = w;
    output(k) = input(k-M+1:k)' * w;
    err(k) = desired(k) - output(k);
    w = w + 2*u*err(k) .* input(k-M+1:k); % update
end
end

```

7.4.2.1.2.3 matlab 实现

这里借用 MATLAB DSP 库中提供的 `dsp.LMSFilter` 函数实现。核心代码如下：

```

function [ output, err, w ] = adaptivefilter( noised, ref, param, method )

if nargin < 4
    h = adaptfilt.lms(param.M, param.mu);

```

(下页继续)

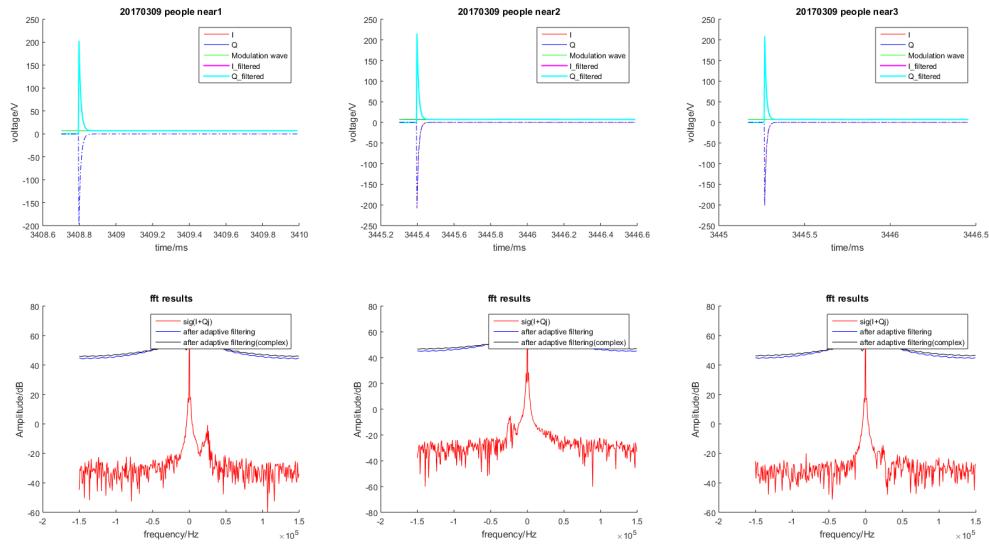


图 7.6: 自适应滤波迭代过程 (虚线为误差)

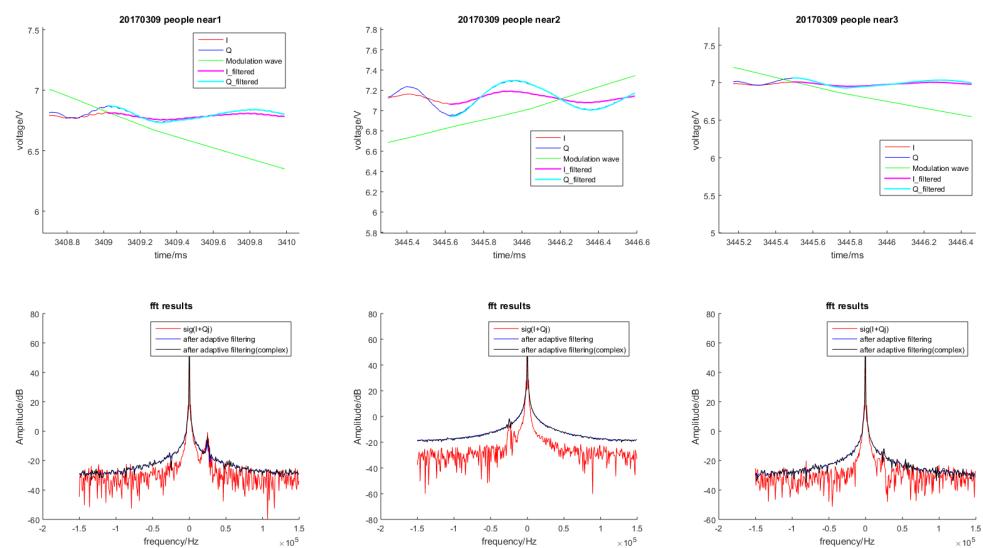


图 7.7: 自适应滤波稳态过程 (虚线为误差)

(续上页)

```

[output, err] = filter(h, ref, noised);
else
    hlms = dsp.LMSFilter('Length', param.M, ...
    'Method', method, ...
    'AdaptInputPort', true, ...
    'StepSizeSource', 'Input port', ...
    'WeightsOutputPort', true);

    [output, err, w] = step(hlms, ref, noised, param.mu, 1);
end

end

```

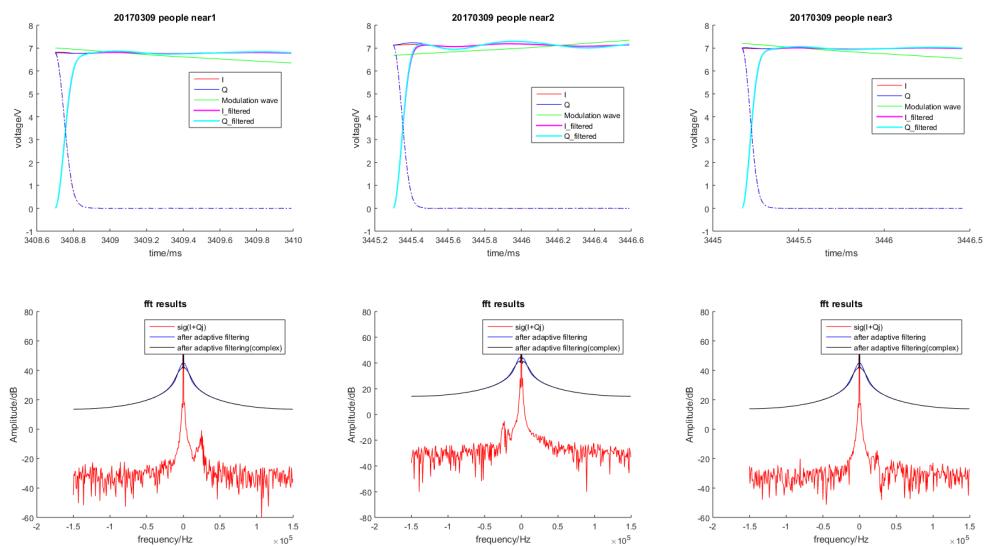


图 7.8: 自适应滤波迭代过程 (虚线为误差)

以下仅示意稳态后的结果.

7.4.2.1.2.4 结果对比

7.4.2.1.2.5 滤波效果分析

7.4.2.1.2.6 参考信号

- 三角波调制信号重复频率为 $1/4e-3 = 250Hz$;
 - 滤波器阶数: $param.M = 30$;
 - 步长: $param.mu = 0.0001$.
1. 带噪雷达回波时延信号
 2. 调制信号

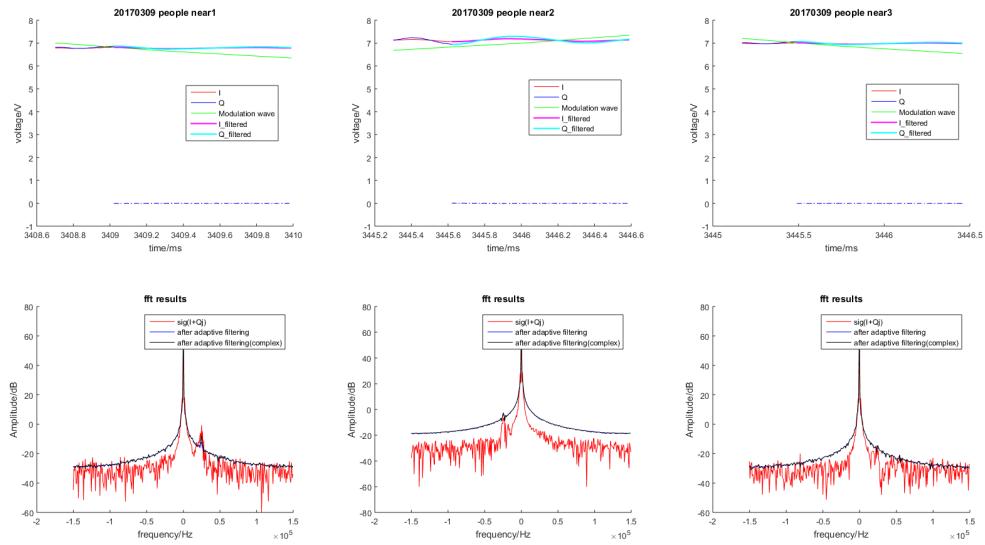


图 7.9: 自适应滤波稳态过程 (虚线为误差)

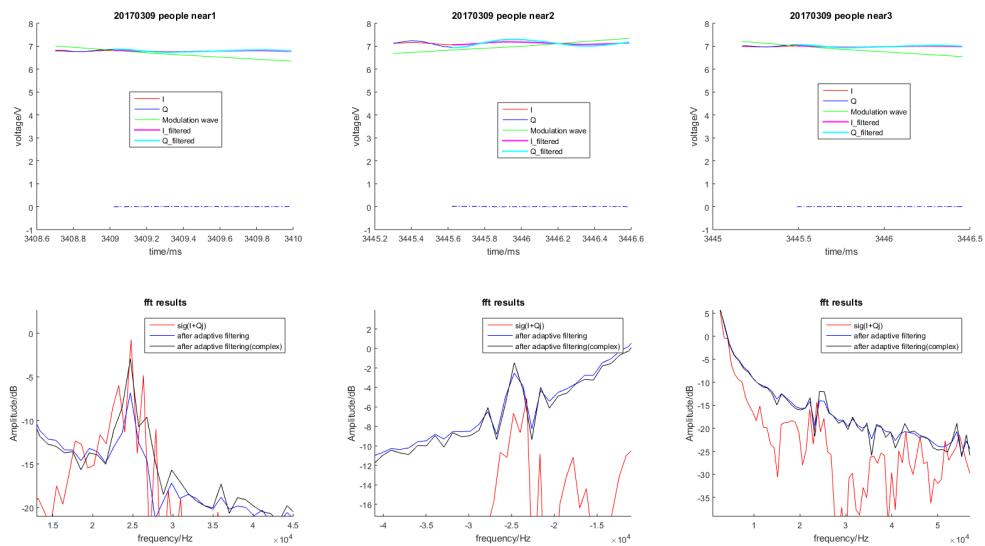


图 7.10: 自己编程实现验证结果图示

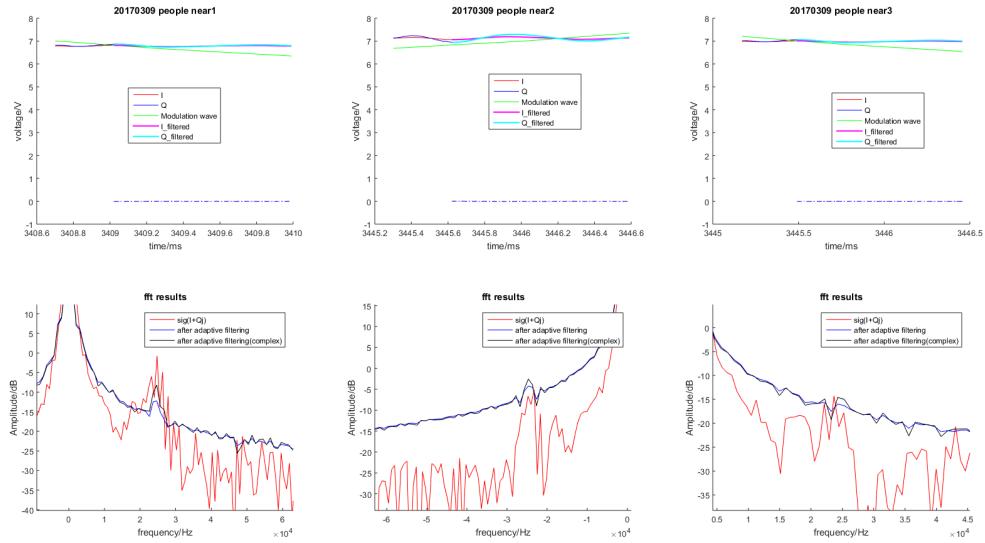


图 7.11: 采用 MATLAB 自带函数库验证结果图示

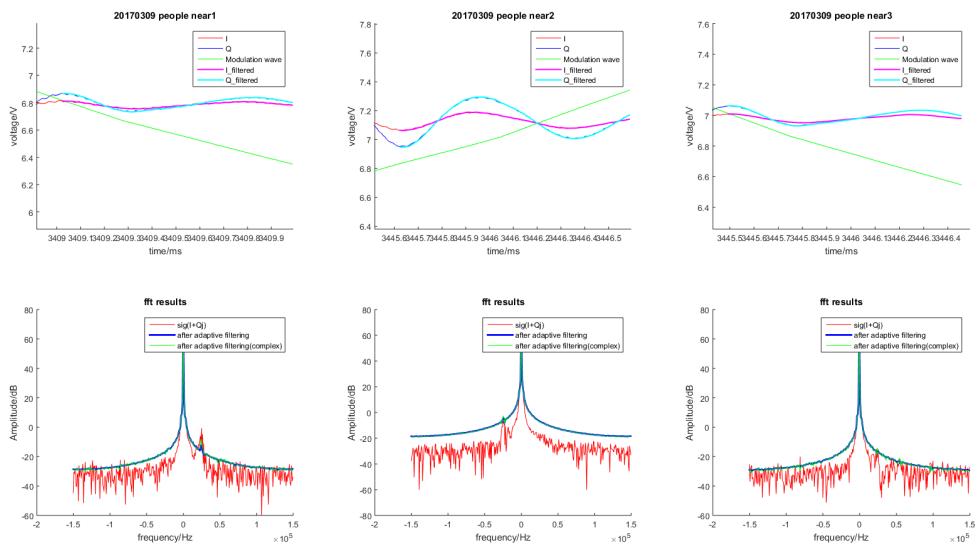


图 7.12: 基于 LMS 的自适应滤波器滤波效果 (参考信号为带噪雷达回波时延信号)

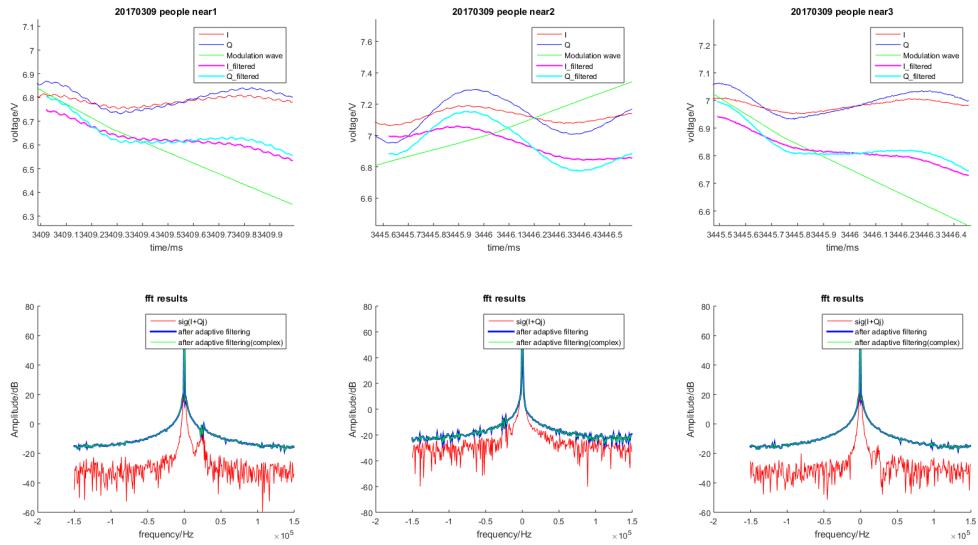


图 7.13: 基于 Normalized LMS 的自适应滤波器滤波效果 (参考信号为带噪雷达回波时延信号)

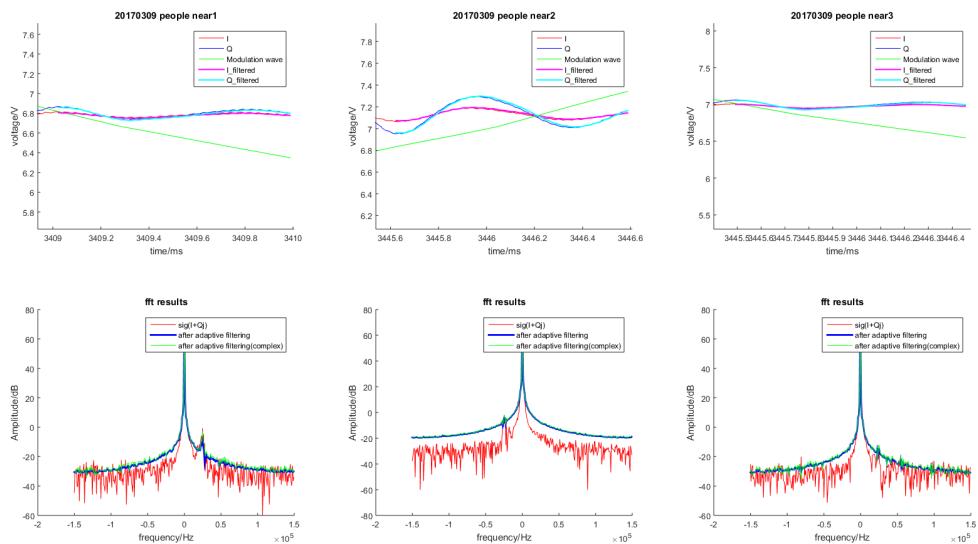


图 7.14: 基于 LMS 的自适应滤波器滤波效果 (参考信号为调制信号)

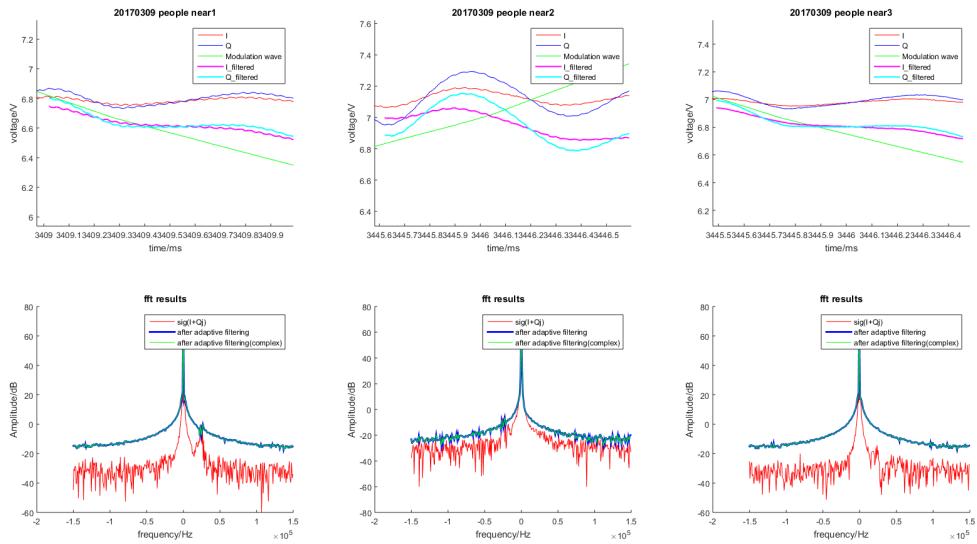


图 7.15: 基于 Normalized LMS 的自适应滤波器滤波效果 (参考信号为调制信号)

7.4.2.1.2.7 权重计算方法

1. 正弦信号

- 实验中, 以正弦信号为原始干净信号, 幅度为 $A = 2$, 频率为 $f_s = 200\text{Hz}$;
- 噪声信号为三角波信号, 频率为 $f_n = 3\text{e}2$;
- 参考输入信号为噪声信号, 幅度为其两倍;
- 实验了基于 **LMS** 和 **Normalized LMS** 的权重更新算法;

当滤波器阶数及步长因子分别取为 $\text{param.M} = 1$; $\text{param.mu} = 0.0001$; 时, 迭代更新过程结果图如下:

自适应滤波稳态后结果图如下:

2. 雷达回波信号

对于雷达回波信号, 参考信号取为带噪宽带信号的时延信号, 效果图如下:

7.4.2.1.2.8 滤波器阶数

- 三角波调制信号重复频率为 $1/4\text{e}-3 = 250\text{Hz}$;
- 参考输入信号: 带噪雷达回波时延信号;
- 权重更新算法: **LMS**;
- 步长: $\text{param.mu} = 0.0001$.

1. 当滤波器阶数取为 $\text{param.M} = 1$ 时

自适应滤波结果如图所示:

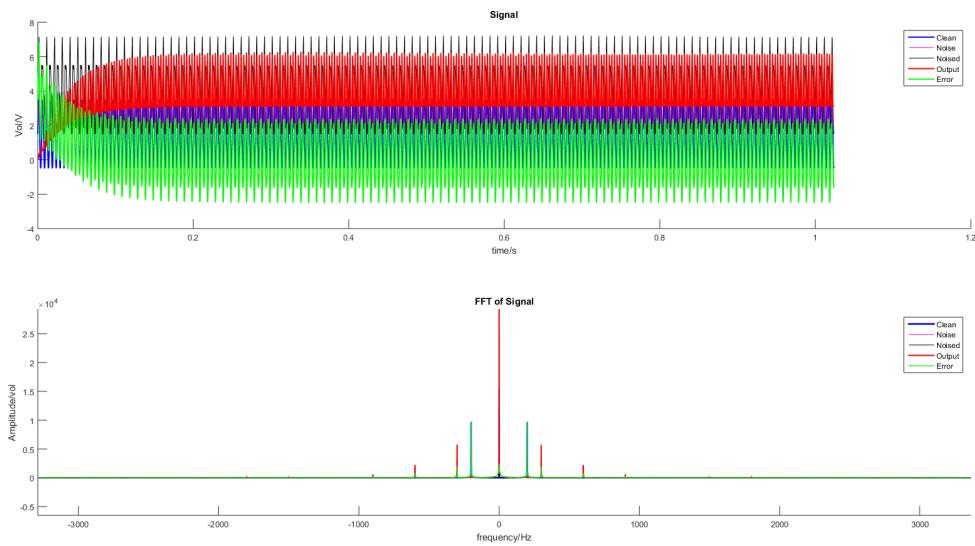


图 7.16: 基于 LMS 的自适应滤波器对带噪正弦信号的滤波效果 (迭代过程图)

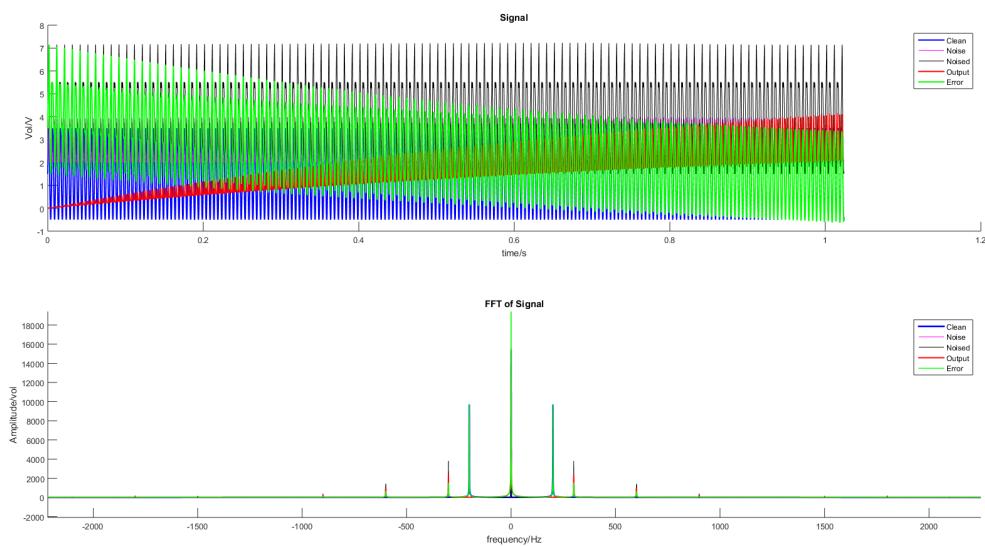


图 7.17: 基于 Normalized LMS 的自适应滤波器对带噪正弦信号的滤波效果 (迭代过程图)

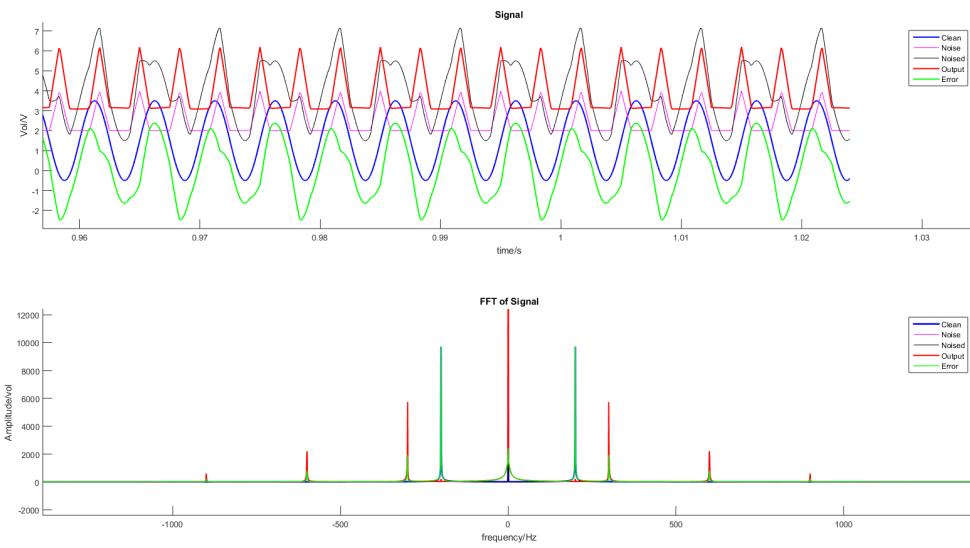


图 7.18: 基于 LMS 的自适应滤波器对带噪正弦信号的滤波效果 (稳态图)

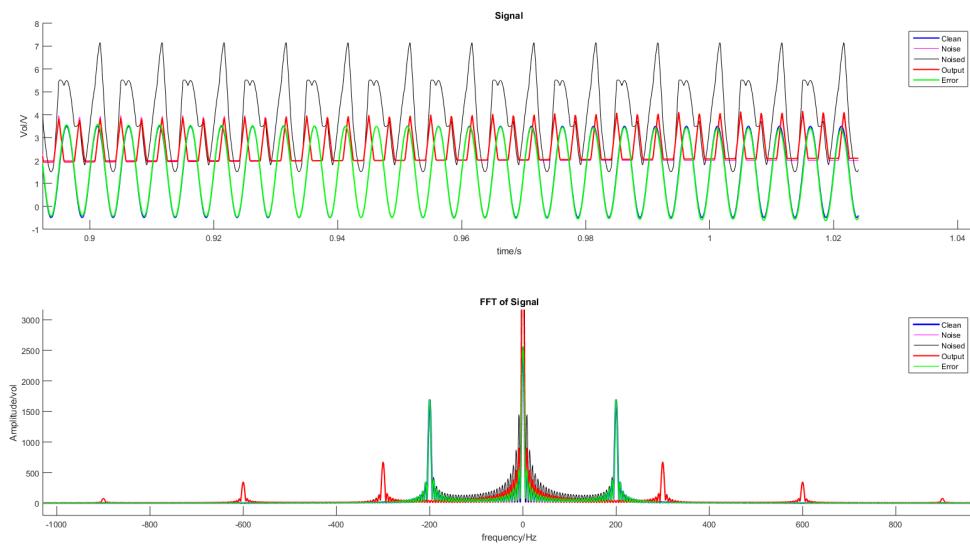


图 7.19: 基于 Normalized LMS 的自适应滤波器对带噪正弦信号的滤波效果 (稳态图)

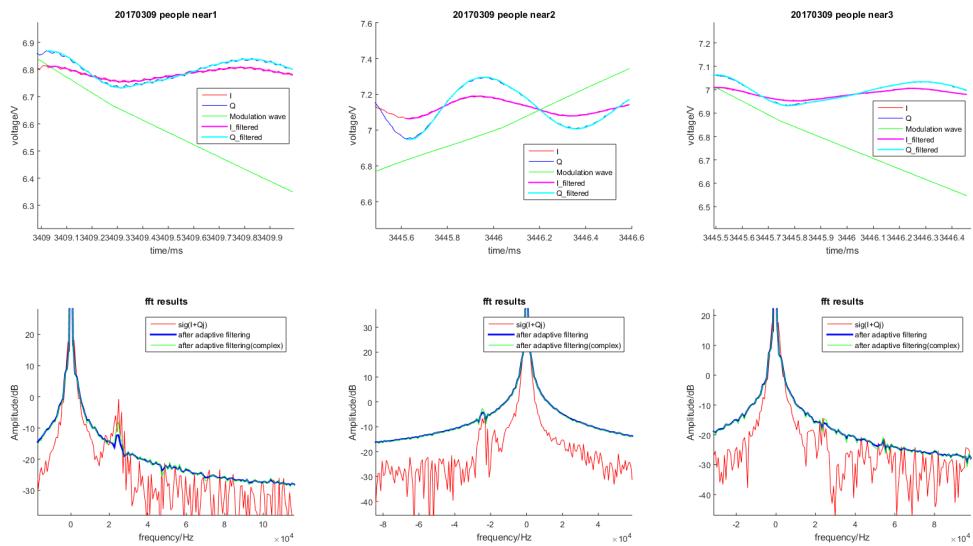


图 7.20: 基于 LMS 的自适应滤波器对雷达回波宽带信号的滤波效果 (稳态图)

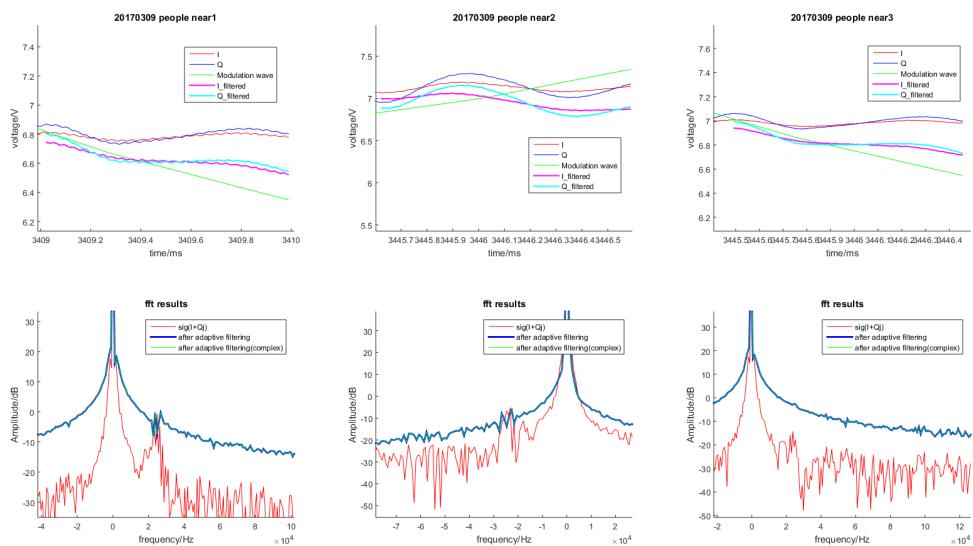


图 7.21: 基于 Normalized LMS 的自适应滤波器对雷达回波宽带信号的滤波效果 (稳态图)

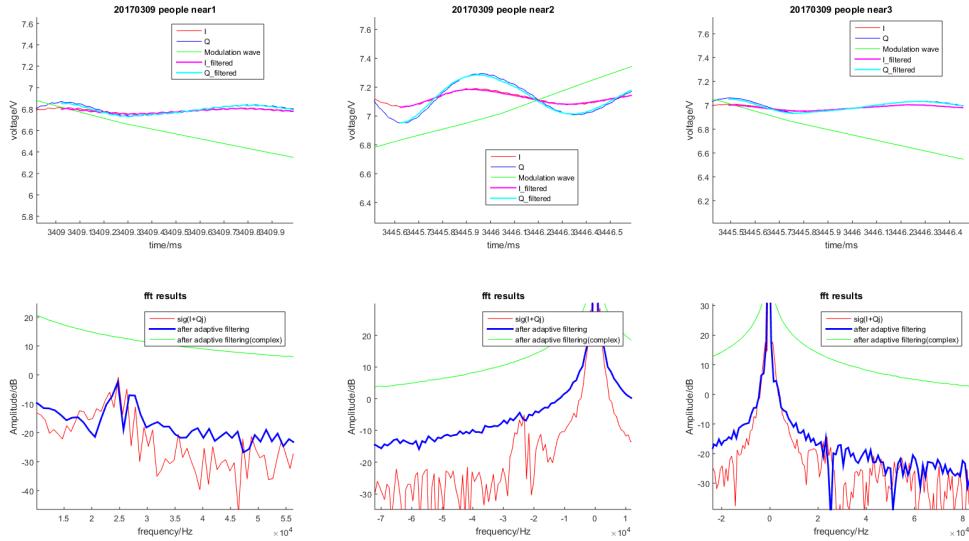


图 7.22: 滤波器阶数取为 1

2. 当滤波器阶数取为 $\text{param.M} = 10$ 时

自适应滤波结果如图所示:

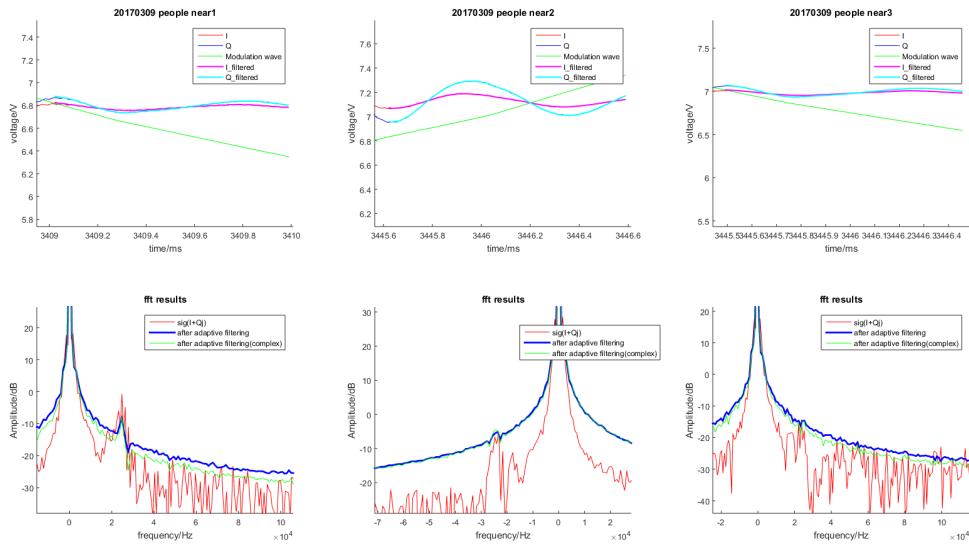


图 7.23: 滤波器阶数取为 10

3. 当滤波器阶数取为 $\text{param.M} = 30$ 时

自适应滤波结果如图所示:

4. 当滤波器阶数取为 $\text{param.M} = 50$ 时

自适应滤波结果如图所示:

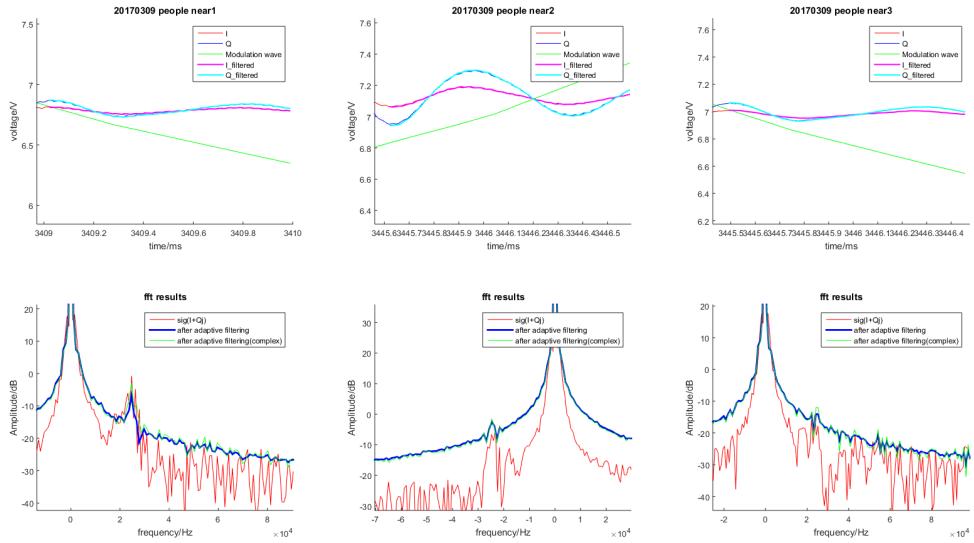


图 7.24: 滤波器阶数取为 30

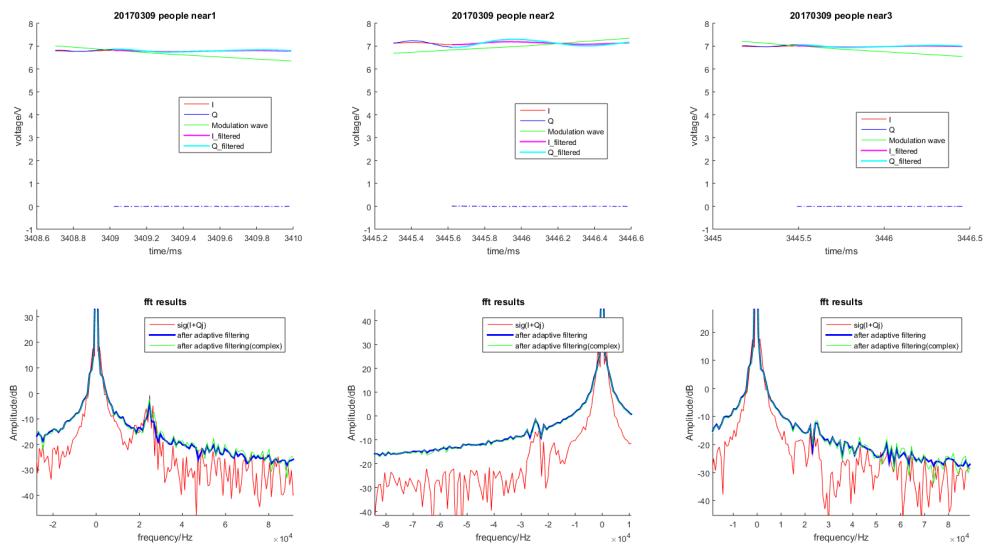


图 7.25: 滤波器阶数取为 50

7.4.2.1.2.9 步长因子

- 三角波调制信号重复频率为 $1/4e-3 = 250Hz$;
- 参考输入信号: 带噪雷达回波时延信号;
- 权重更新算法: LMS ;
- 滤波器阶数取为: param.M = 30 .

1. 当步长因子为 param.mu = 0.001 时

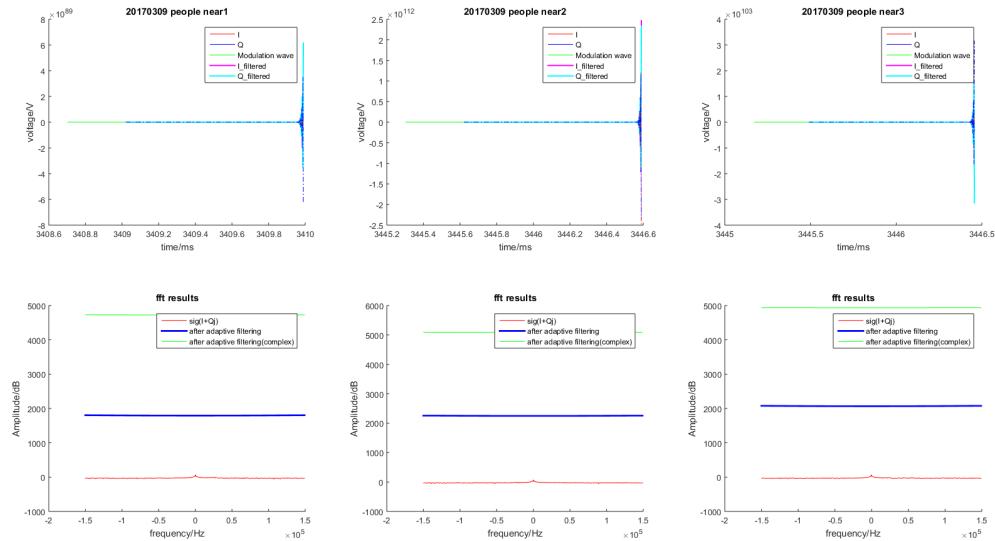


图 7.26: 步长因子为 0.001

2. 当步长因子为 param.mu = 0.0001 时

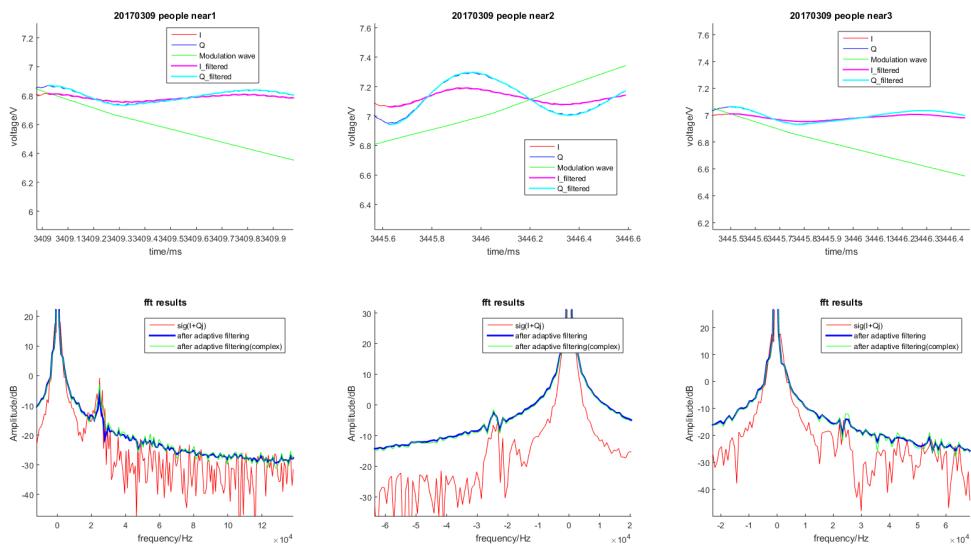


图 7.27: 步长因子为 0.0001

3. 当步长因子为 $\text{param.mu} = 0.0003$ 时

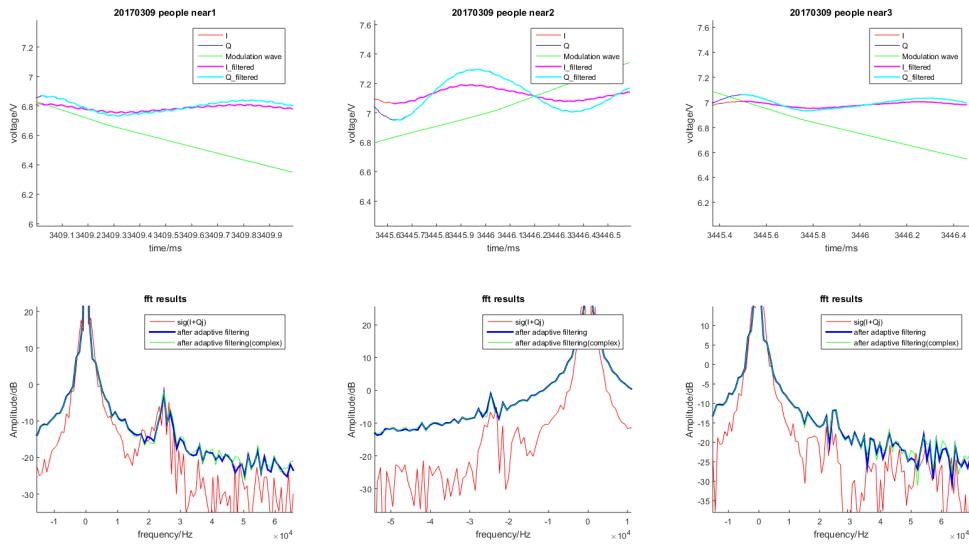


图 7.28: 步长因子为 0.0003

4. 当步长因子为 $\text{param.mu} = 0.00001$ 时

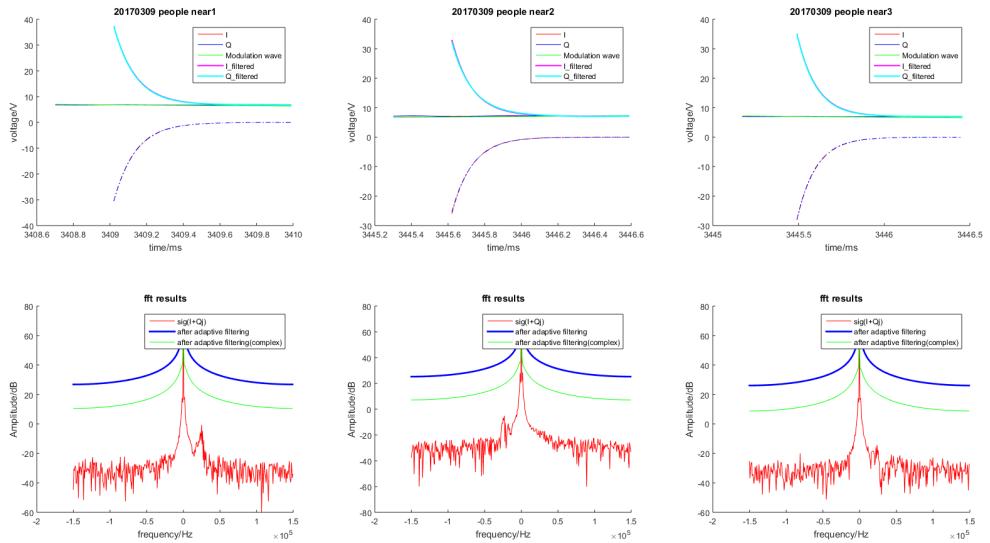


图 7.29: 步长因子为 0.00001

结论: 步长越大, 收敛越快, 但若步长过大, 会导致很大误差!

7.4.2.1.3 如何使用学习好的权重

7.4.2.1.3.1 利用学习好的权重滤波

核心代码:

```
LEN = length(input);
w = param.w;
M = param.M;
u = param.mu;

output = zeros(LEN,1);
err = output;

for k = M:LEN
    output(k) = input(k-M+1:k)'*w;
    err(k) = desired(k) - output(k);
end
```

效果如下:

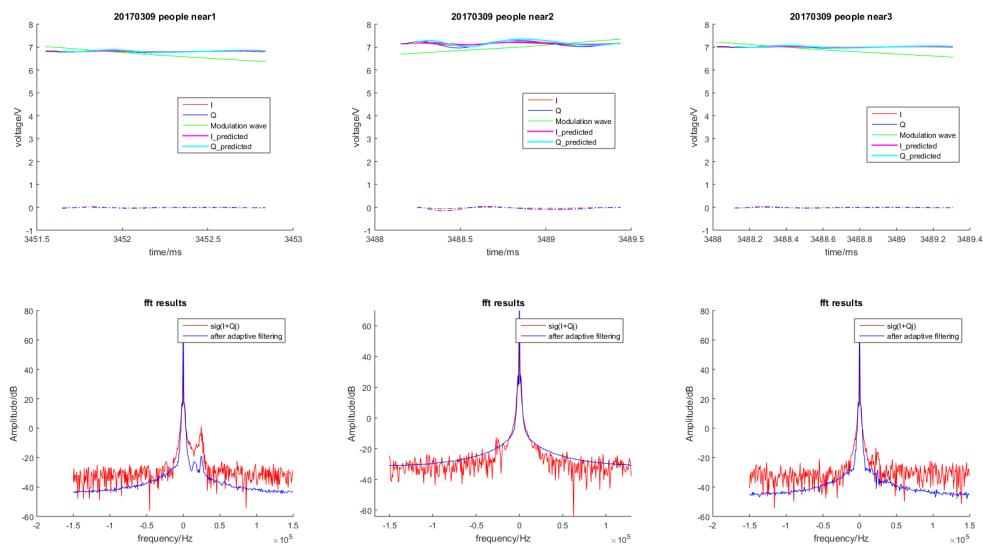


图 7.30: 直接利用学习好的权重滤波

7.4.2.1.3.2 基于学习好的权重再滤波

核心代码:

```

LEN = length(input);
w = param.w;
M = param.M;
u = param.mu;

output = zeros(LEN, 1);
err = output;

for k = M:LEN
    output(k) = input(k-M+1:k)' * w;
    err(k) = desired(k) - output(k);
    w = w + 2*u*err(k).*input(k-M+1:k); % update
end

```

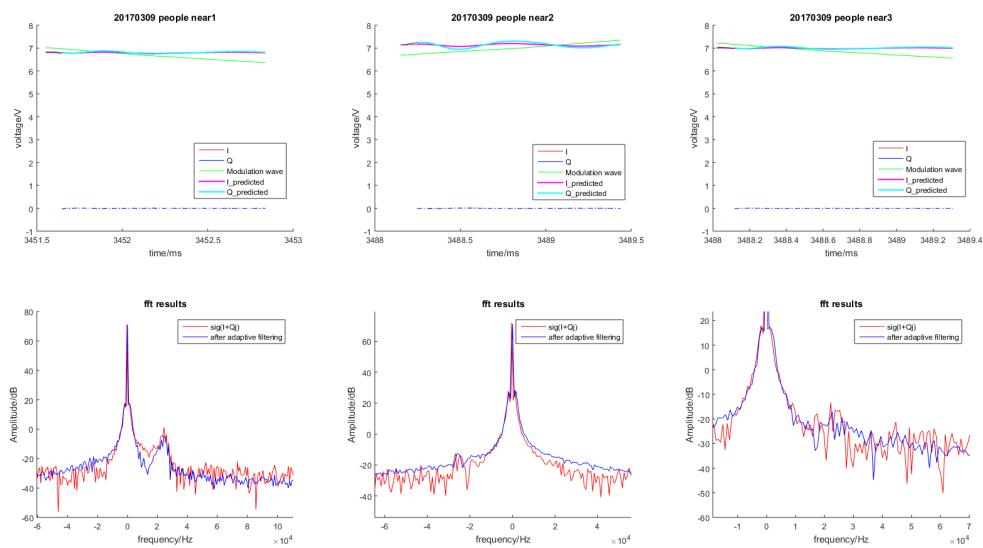


图 7.31: 使用学习好的权重初始化自适应滤波器

7.4.3 探测

7.4.3.1 简介

主要介绍连续波雷达与调频连续波雷达测速, 测距与测角原理.

7.4.3.2 距离与速度测量

7.4.3.2.1 三角波测距与测速

频率调制连续波 (Frequency Modulated Continuous Wave, FMCW) 测距与测速, 发射一信号频率呈现三角波变化的调频连续波, 利用时延及多普勒效应实现测距与测速。其原理可由下图给出:

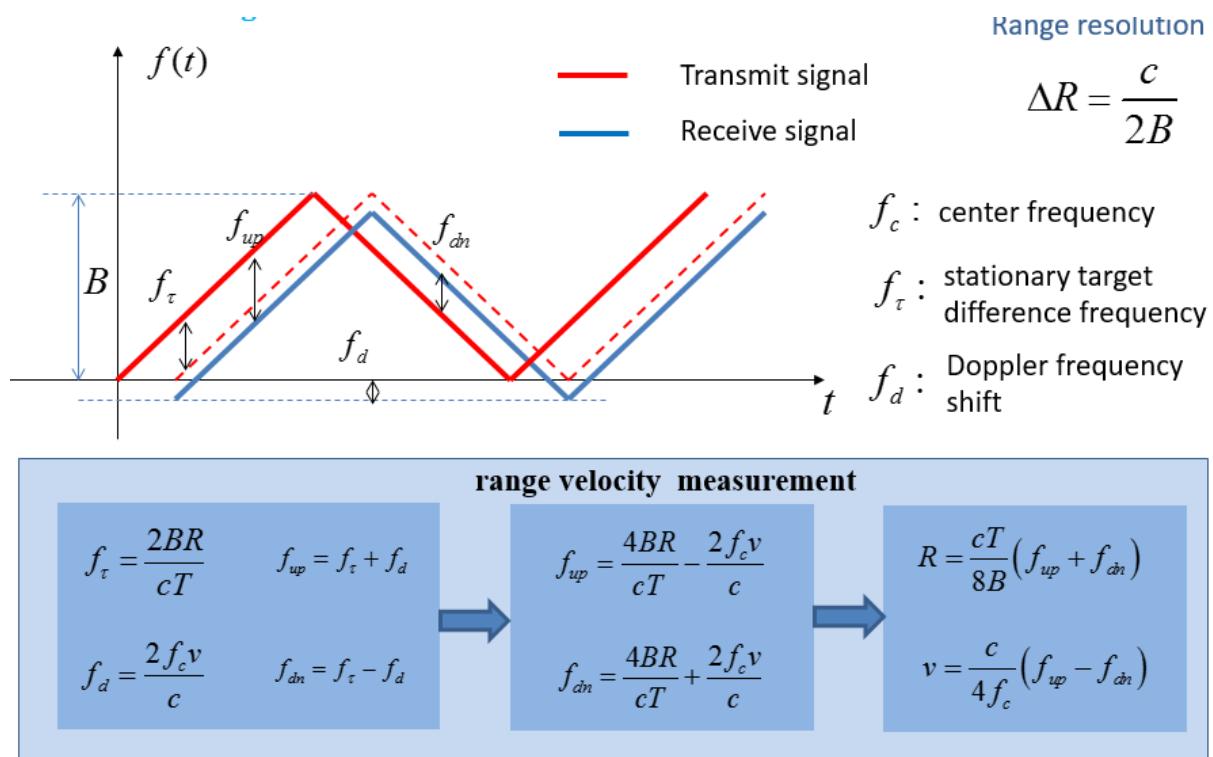


图 7.32: 三角波测距与测速

其中:

- c 为光速;
- B 为调制带宽;
- T_m 为调制周期;
- R 为目标与雷达间的距离;
- f_c 为载频;
- f_τ 为静止目标差频;
- f_d 为运动目标产生的多普勒频移;

7.4.3.2.2 锯齿波测距

锯齿波测距原理示意图如下:

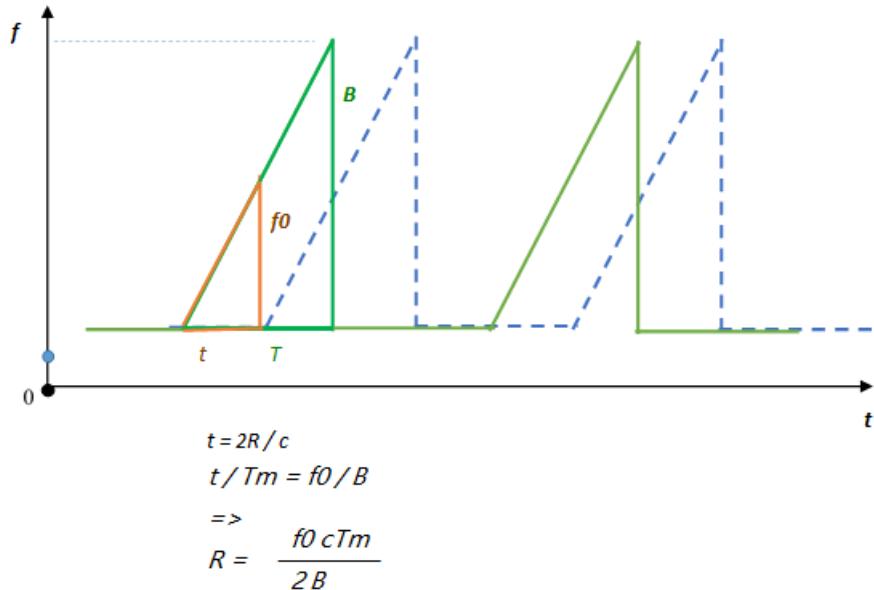


图 7.33: 锯齿波测距

其中:

- f_0 为目标静止时的差频;
- f_d 为多普勒差频, 可通过 FFT 获得;
- B 为调制带宽;
- T_m 为调制周期;
- c 为光速;
- R 为距离;
- v 为速度;

7.4.3.2.3 距离速度模糊

高脉冲重复频率信号

距离模糊: 目标回波延迟时间大于脉冲重复周期, 同一距离

- 倍乘法解模糊
- DFS 解模糊
- 三角波变频法

在多普勒速度不模糊时, MTD 求得的结果准确

- 复合体制测速;

7.4.3.2.3.1 测速模糊

由香浓定理, 要满足测速不模糊, 需要采样频率 f_r 不小于两倍的由速度产生的多普勒频率, 即满足 $f_r < 2f_{d_{max}}$, 由此可以推导出采样周期(即脉冲重复周期), 具体计算过程如下:

为保证速度不模糊, 需满足 :

$$f_r \geq 2 f_{d_{max}} \quad \text{——香浓定理}$$

$$\Rightarrow f_{d_{max}} \leq \frac{1}{2} f_r = \frac{1}{2T_r}$$

$$\Rightarrow T_r \leq \frac{1}{2f_{d_{max}}}$$

$$\text{又 } f_d = \frac{1}{2\pi} \frac{d\varphi}{dt} = \frac{2}{\lambda} v_r$$

$$T_r \leq \frac{1}{2f_{d_{max}}} \leq \frac{\lambda}{4v_{r_{max}}} = \frac{c}{4f_0 v_{r_{max}}}$$

举例 :

$$f_d = 2v_r/(c/f_0) = \begin{cases} 160, & v_r = 1m/s, f_0 = 24GHz \\ 512, & v_r = 1m/s, f_0 = 77GHz \end{cases}$$

$$T_r \leq \begin{cases} 3.125ms, & v_r = 1m/s, f_0 = 24GHz \\ 0.976ms, & v_r = 1m/s, f_0 = 77GHz \end{cases}$$

图 7.34: MTD 多普勒测速模糊分析

7.4.3.2.3.2 解决方案

距离速度解耦

采用复合体制, 即 FMCW + CW, 如调制波采用梯形波;

7.4.3.2.3.3 多目标配对

- MTD 可以实现多目标中, 同一目标的距离与速度的配对 (计算量、存储量较大)
- 典型梯形波配对法 (存在虚假目标情况)
- 变周期梯形波配对法 (计算量稍大)

7.4.3.3 角度测量

- 相位法
- 振幅法

7.4.3.3.1 简介

角度测量是探测“目标——天线中心线的连线”与法线的夹角, 多用于目标定位。设计有一根发射天线 TX, 两根接收天线 RX1/RX2, 并有 I1、I2、Q1、Q2 四个工作通道, 采用比相法来实现此功能。

参考文档: [雷达原理角度测量](<https://wenku.baidu.com/view/d44192cee45c3b3567ec8bb7.html>)

7.4.3.3.2 相位差测角

7.4.3.3.2.1 相位差法测角原理概览

相位差法测角原理可总结如下图所示:

7.4.3.3.2.2 相位差法测角原理详解

相位差测角原理示意图如下图所示:

相位差法测角算法如下图所示:

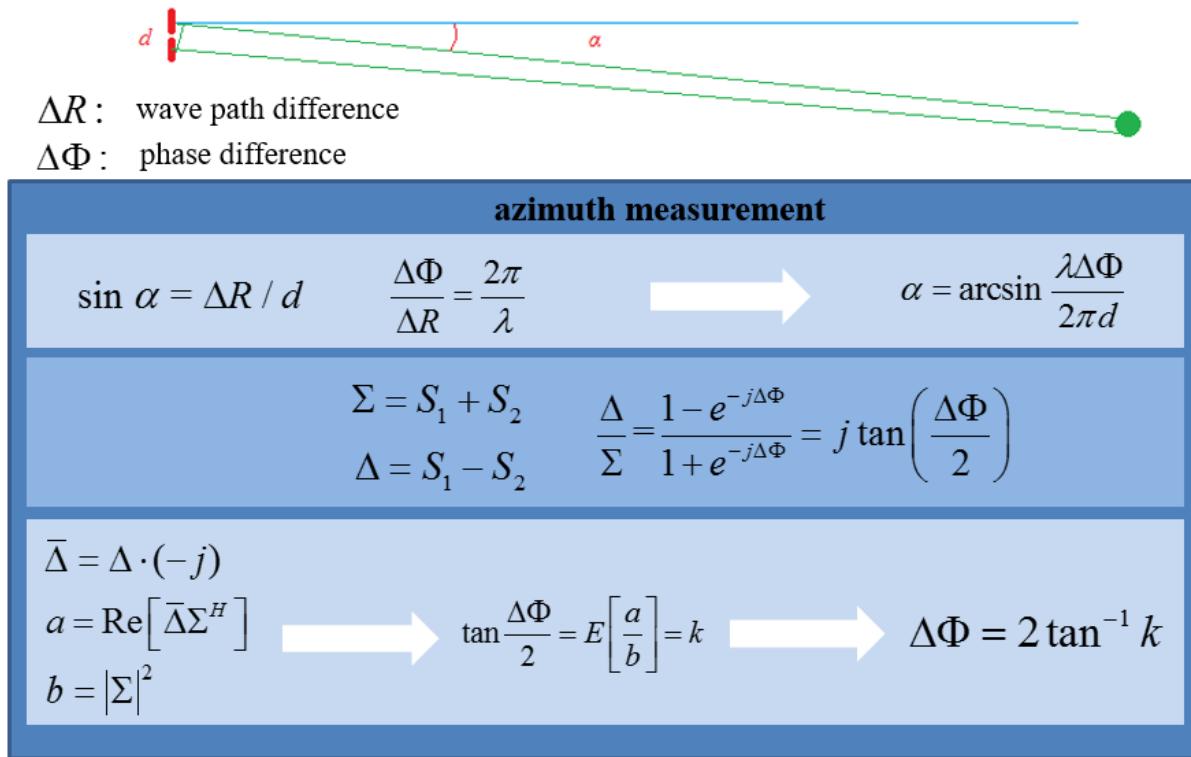
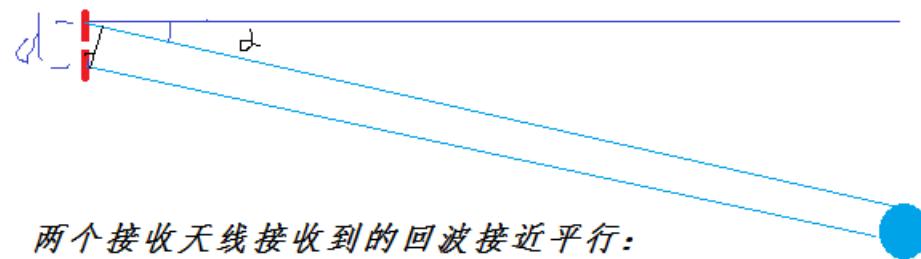


图 7.35: 相位差法测角原理总览



两个接收天线接收到的回波接近平行:

1. 天线间距很小;
2. 目标距天线举例一般较远;

所以, 目标偏离中心法线
的角度, 满足关系式:

电池波在发射时, 相位是一致的, 且没有波程差, 当经
过目标反射回来的电池波存在波程差 ΔR , 相位差 Φ ,
由电池波波长和相位关系知:

$$\frac{\Delta \Phi}{\Delta R} = \frac{2\pi}{\lambda}$$

$$\Rightarrow \alpha = \arcsin \frac{\lambda \Delta \Phi}{2\pi d}$$

图 7.36: 相位差测角原理示意图

$$\begin{aligned}\Sigma(\phi) &= S_1 + S_2 \\ \Delta(\phi) &= S_1 - S_2\end{aligned}$$

$$\frac{\Delta}{\Sigma} = \frac{1 - e^{-j\phi}}{1 + e^{-j\phi}} = j \tan\left(\frac{\phi}{2}\right)$$

我们用其模值来计算角度大小：

实际上，可以直接求

$$2 \arctan \left| \frac{\Delta}{\Sigma} \right| = |\phi| \quad \text{或} \quad \varphi = 2 \operatorname{real}(\operatorname{atan}(-j \frac{\Delta}{\Sigma}))$$

$$\varphi = 2 \operatorname{atan}(\operatorname{real}(-j \frac{\Delta}{\Sigma}))$$

算法步骤：

(1) 差信号移相 90 度： $\bar{\Delta}(t) = \Delta(t) \cdot (-j)$

(2) 计算差共轭与和信号乘积的实部： $a(t) = \operatorname{Re}[\bar{\Delta}(t) \Sigma^H(t)]$ ，由于噪声影响。

(3) 计算和信号的功率： $b(t) = |\Sigma(t)|^2$

(4) 计算误差信号的期望： $\tan \frac{\phi}{2} = E \left[\frac{a(t)}{b(t)} \right] = k$

(5) 计算角度： $\phi = 2 \tan^{-1} k$ ，若为正数，正角度；若为负，负角度。

图 7.37: 相位差法测角算法

7.4.3.3.2.3 相位差法测角范围

相位差法测角利用多个天线所接收到的回波信号间的相位差测角，测角范围与波长及天线间距相关，计算方法如下图所示

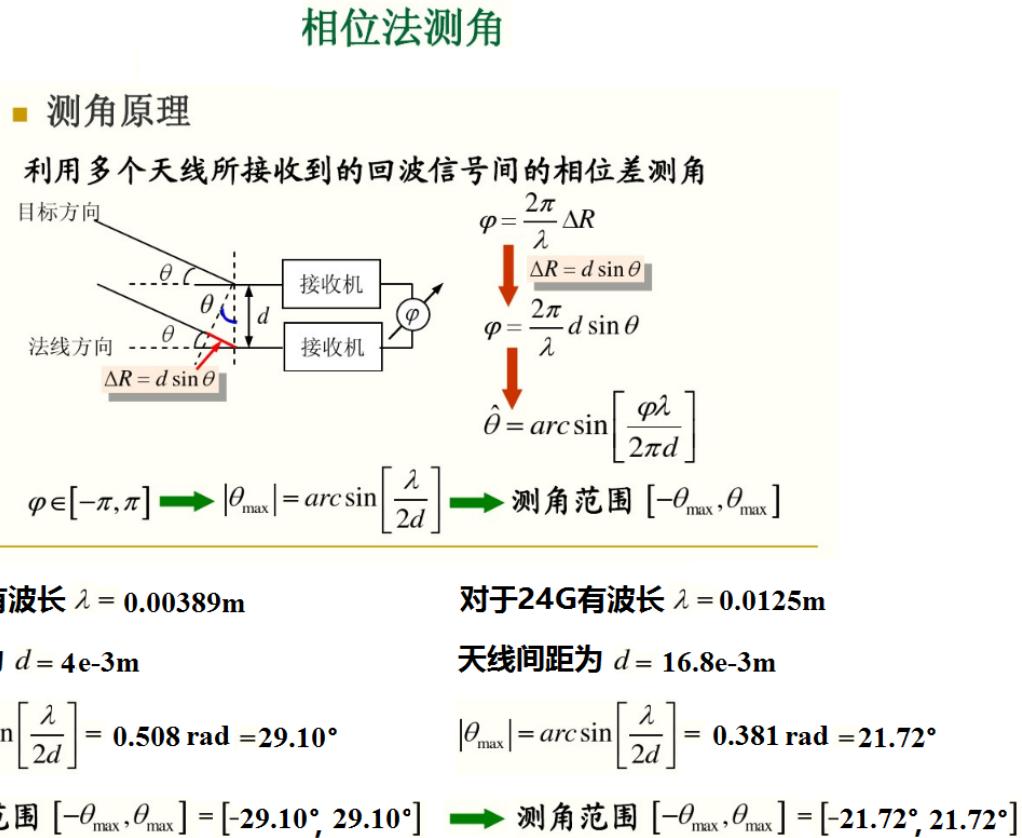


图 7.38: 相位差法测角范围分析

7.4.3.3.2.4 相位差法测角误差及多值性

相位差法测角误差分析如下，可见该方法存在测角模糊问题。

一种解决方法是采用三天线法测角，原理如下图

7.4.4 成像

7.4.4.1 简介

主要介绍连续波雷达与调频连续波雷达测速，测距与测角原理。

■ 测角误差与多值性问题

测角误差

$$\varphi = \frac{2\pi}{\lambda} d \sin \theta \rightarrow d\varphi = \frac{2\pi}{\lambda} d \cos \theta d\theta \rightarrow d\theta = \frac{\lambda}{2\pi d \cos \theta} d\varphi$$

$\frac{\lambda}{d}$ ↓ ↑ $\frac{d}{\lambda}$ ↑ ↓ $\varphi = \frac{2\pi}{\lambda} d \sin \theta$ 可能超出 2π

↑ 鉴相精度

↑ 实际读数

\rightarrow 测相模糊 $\rightarrow \varphi = 2\pi N + \psi$

[提高测相精度要求 $\frac{\lambda}{d}$ 尽量小]

[测相不模糊要求 $\frac{\lambda}{d}$ 尽量大] 矛盾！！！

图 7.39: 相位差法测角误差分析

解决方法 三天线法测角

$$\varphi_{12} = \frac{2\pi}{\lambda} d_{12} \sin \theta < 2\pi \quad \text{无测角模糊}$$

测角精度差

$$\varphi_{13} = \frac{2\pi}{\lambda} d_{13} \sin \theta = 2\pi N + \psi \quad \text{测角模糊}$$

测角精度高

$$\frac{\varphi_{13}}{\varphi_{12}} = \frac{d_{13}}{d_{12}} = \frac{2\pi N + \psi}{\varphi_{12}} \rightarrow N = \text{INT}\left[\varphi_{12} \cdot \frac{d_{13}}{d_{12}} / 2\pi\right]$$

$$\theta = \arcsin\left[\frac{\varphi_{13} \lambda}{2\pi d_{13}}\right] \quad \varphi_{13} = 2\pi N + \psi$$

图 7.40: 三天线法测角

7.4.4.2 单频连续波雷达成像

7.4.4.2.1 一般集合

7.4.4.2.1.1 概念

7.4.5 识别

7.4.6 参考文献

7.5 合成孔径雷达

7.5.1 合成孔径雷达简介

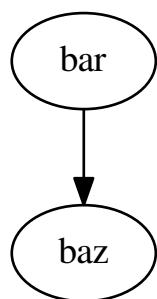
7.5.1.1 What is ?

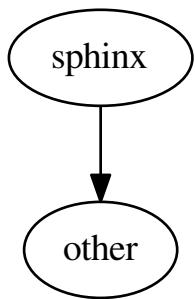
合成孔径雷达 (*Synthetic Aperture Radar*, SAR) 的概念最早由 Carl Wiley 于 1951 年提出.

dSARsim (<http://af-projects.it/dsarsim>) :Dummy Synthetic Aperture Radar Simulator

<https://pypi.org/project/GDAL/>

<https://gdal.org/>

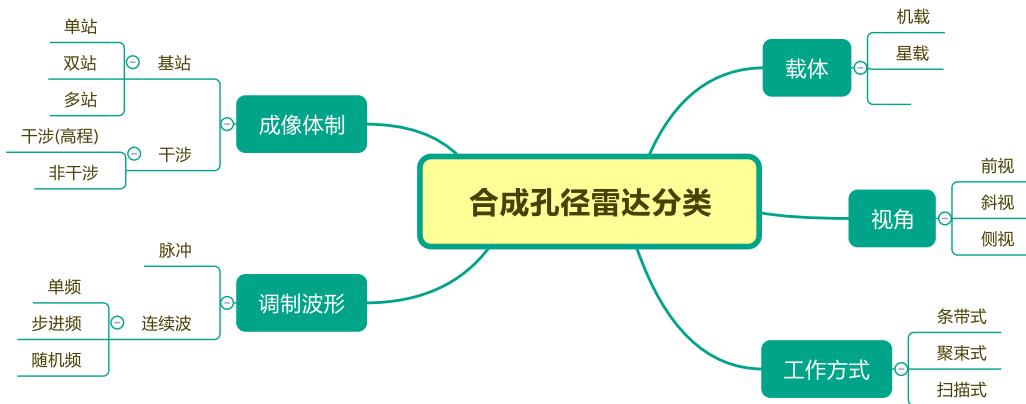




7.5.2 SAR 系统概述

7.5.2.1 相关概念汇总

7.5.2.1.1 SAR 分类

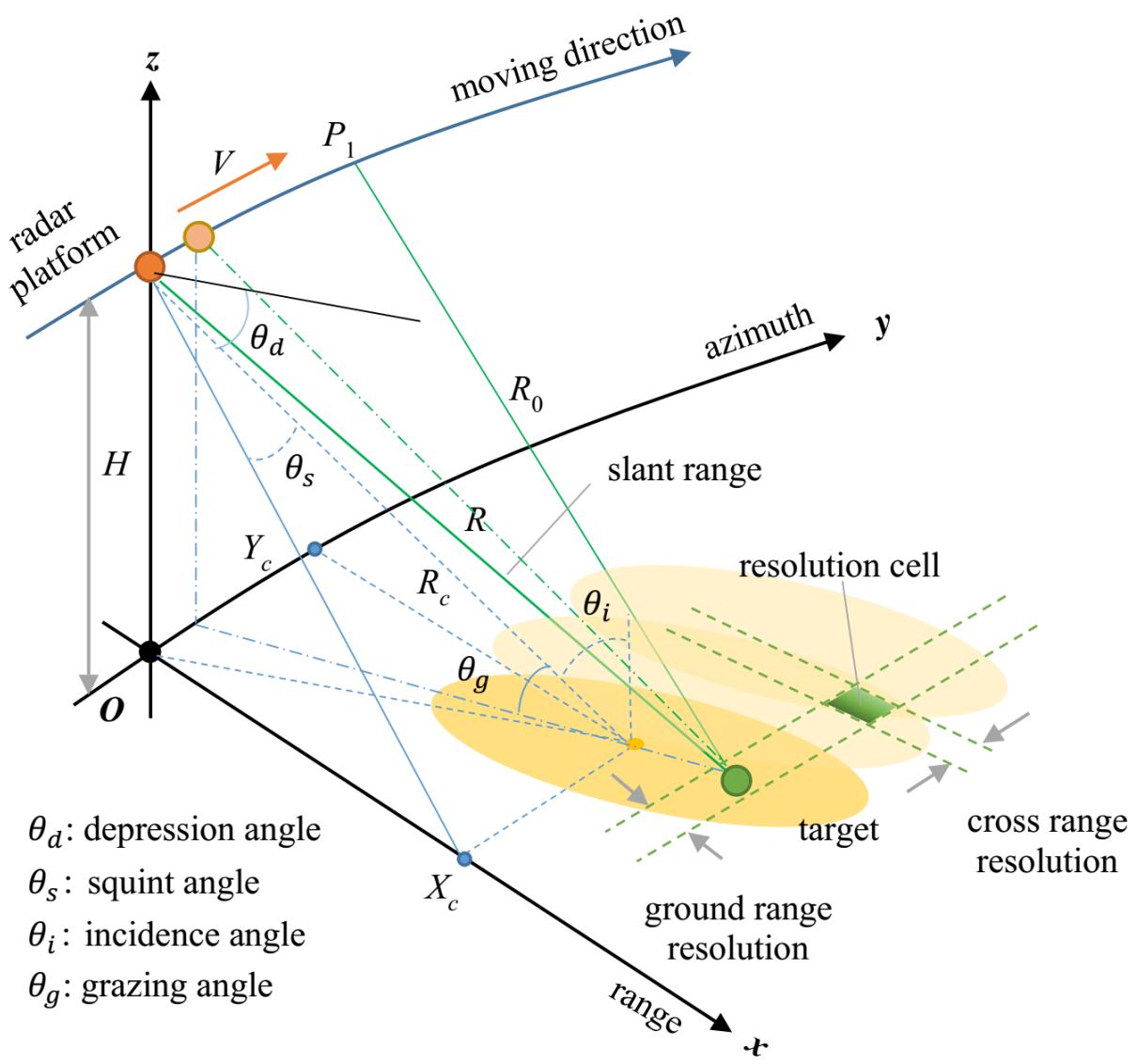


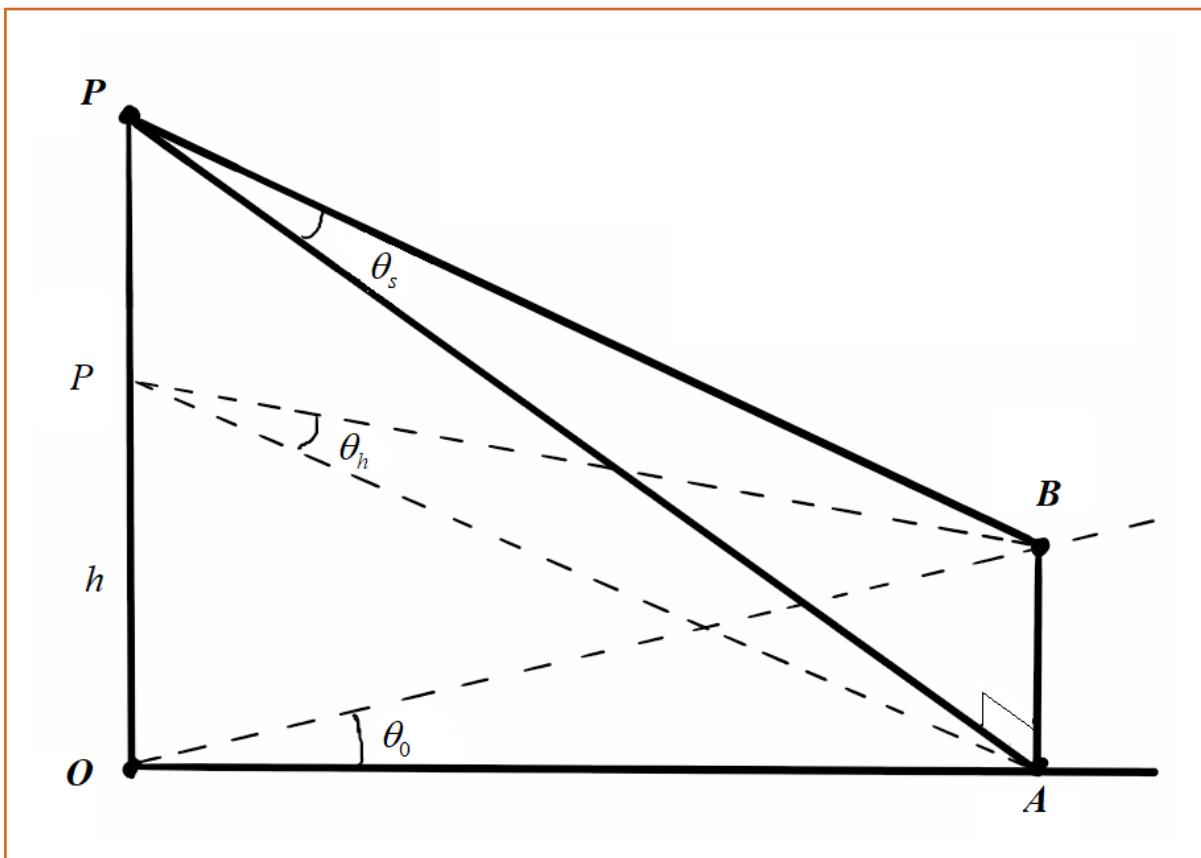
Synthetic-aperture radar (http://en.volupedia.org/wiki/Synthetic-aperture_radar)

7.5.2.1.2 SAR 成像几何关系

警告: 注意上图中的斜视角定义为波束中心线方向与 $x - O - z$ 平面间的夹角. 如下图所示几何关系
知 θ_h 与 θ_0 满足如下关系

$$\tan\theta_h = \frac{1}{\sqrt{\left(\frac{OP}{AB}\right)^2 + \frac{1}{\tan^2\theta_0}}}.$$





如果雷达在距离向上不作扫描, 那么雷达成像区域范围与波束宽度有关, 所以超出波束覆盖范围的地方是没法成像的?

提示:

- 倾角 (*Depression Angle*) : 如图中 θ_d ;
- 斜视角 (*Squint Angle*) : 如图中 θ_s , 即波束中心线与零多普勒线之间的夹角
- 入射角 (*Incidence Angle*) : 也叫视角 (look angle), 即波束照射方向与地平面的法线间的夹角, 如图中 θ_i ;
- 猎地角 (*Grazing Angle*) : 即波束照射方向与地平面间的夹角, 如图中 θ_g .
- 轨道倾角 (*Orbital Inclination*) : 某一轨道相对于另一参考平面的倾斜角度, 图中未标出.
- 目标被波束中心照射一般是指波束中心线经过目标的时刻.

7.5.2.1.3 符号列表

- C : 光速 ([Light speed](http://en.volupedia.org/wiki/Speed_of_light) (http://en.volupedia.org/wiki/Speed_of_light));
- f_0 : 中心频率 ([Center frequency](http://en.volupedia.org/wiki/Center_frequency) (http://en.volupedia.org/wiki/Center_frequency));
- f_c : 载波频率 ([Carrier frequency](http://en.volupedia.org/wiki/Carrier_frequency) (http://en.volupedia.org/wiki/Carrier_frequency)), 指载波的中心频率/频率;
- $\lambda = \frac{C}{f_c}$: 波长 ([Wavelength](http://en.volupedia.org/wiki/Wavelength) (<http://en.volupedia.org/wiki/Wavelength>));
- D 或者 La : 天线真实孔径长度;
- L_s : 合成孔径长度 (Synthetic aperture length);
- T_s : 合成孔径时间 (Synthetic aperture time);
- R : 雷达与目标间的垂直距离/瞬时斜距;
- $\alpha = \lambda/D$: 天线波束宽度 (Antenna beam width);
- H : 平台高度;
- V : 平台运动速度;
- T_p : 脉冲宽度 (pulse width);
- B : 带宽 (band width);
- K_r : 线性调频率 ([Linear chirp rate](http://en.volupedia.org/wiki/Chirp#Rate) (<http://en.volupedia.org/wiki/Chirp#Rate>)), $K_r = B/T_p$;

合成孔径与真实孔径: $L_s = \frac{\lambda}{D} \cdot R$

合成孔径时间与合成孔径长度: $L_s = VT_s$

实孔径距离向分辨率: $\rho_r = \frac{C}{2K_r T_p} = \frac{C}{2B}$

实孔径方位向分辨率: $\rho_a = \alpha \cdot R = \frac{\lambda \cdot R}{D}$

合成孔径方位向分辨率: $\rho_a = \alpha_h \cdot R = \frac{\lambda}{2L_s} \cdot R = \frac{D}{2}$

因为合成孔径雷达发射和接收共用一副天线且信号的距离差是双程差, 进一步锐化了波束, 所以合成孔径雷达的有效半功率波束宽度近似为相同长度实孔径的一半, 它的半功率波束宽度为: $\alpha_h = \frac{\lambda}{2L_s}$

SAR 利用脉冲压缩技术获得高的距离向分辨率, 利用合成孔径原理获得高的方位向分辨率, 从而获得高分辨率雷达图像.

SAR 回波信号经距离向脉冲压缩后, 雷达的距离分辨率由雷达发射信号带宽决定: $\rho_r = \frac{C}{2B_r}$, 式中 ρ_r 表示雷达的距离分辨率, B_r 表示雷达发射信号带宽, C 表示光速.

同样, SAR 回波信号经方位向合成孔径后, 雷达的方位分辨率由雷达方位向的多谱勒带宽决定: $\rho_a = \frac{v_a}{B_a}$, 式中 ρ_a 表示雷达的方位分辨率, B_a 表示雷达方位向多谱勒带宽, v_a 表示方位向 SAR 平台速度. 在小斜视角的情况下, 方位分辨率近似表示为 $\rho_a = \frac{D}{2}$, 其中 D 是方位向合成孔径的长度.

7.5.2.1.4 SAR 数据获取

设 $t = 0$ 时刻, 天线发射一脉宽为 T_p 的脉冲, 于 t_1 时刻脉冲起始边缘最先到达近距地面点, 于 t_2 时刻脉冲结束边缘到达远距地面点, t_1 时刻到达地面的波束, 于 $2t_1$ 时刻返回到天线, t_2 到达地面的波束于 $2t_2$ 时刻返回到天线. 接收机在稍早于 $2t_1$ 的时刻开始采样, 于稍晚于 $2t_2$ 时刻的时候停止采样. 由此, 雷达距离向的采样时间为 $T_{sr} = 2t_2 - 2t_1$

2-D Raw Data Matrix

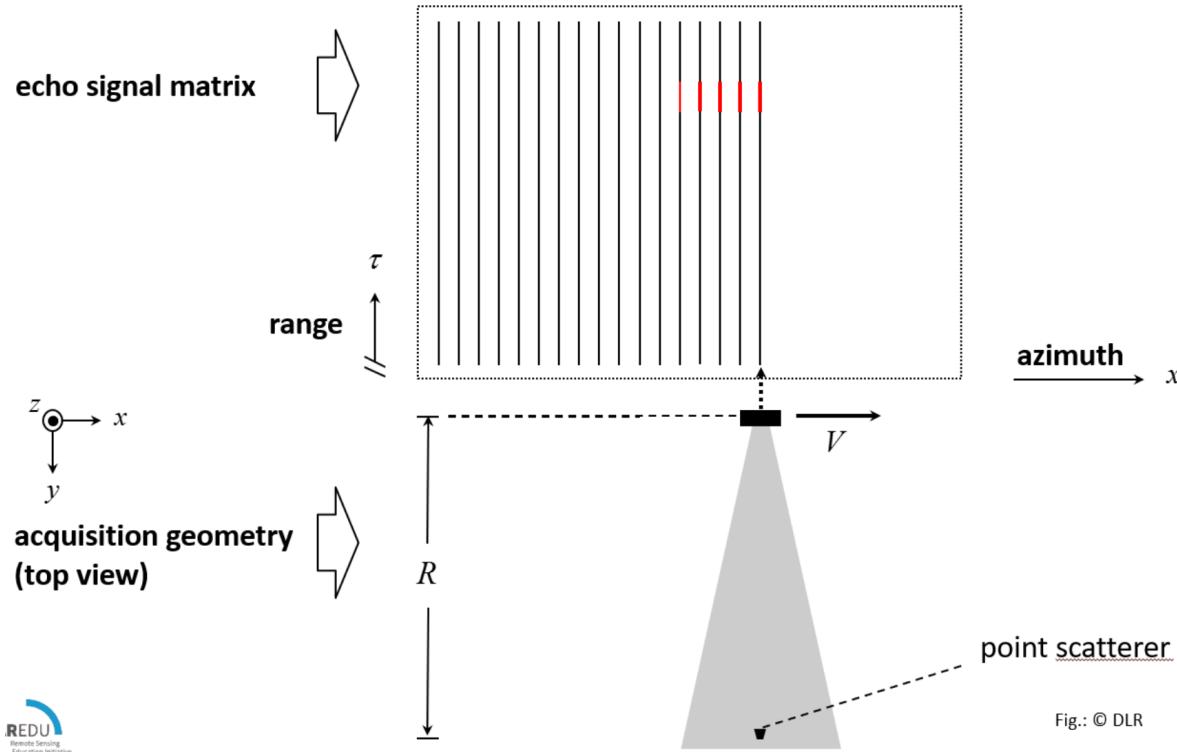


Fig.: © DLR

图 7.41: 成像与原始数据存储

成像与原始数据存储

7.5.2.1.4.1 采样参数的选取

记方位向采样频率即脉冲重复频率 (PRF) 为 F_{sa} , 方位向采样时间为 T_{sa} , 距离向采样频率为 F_{sr} , 距离向采样时间为 T_{sr} , 近地点脉冲起始到达时刻为 t_1 , 远地点脉冲起始远离时刻为 t_2 , 则需要满足以下准则 (参见文献 [1] pp.90):

1. 奈奎斯特采样率: 即 $F_{sa} > \alpha_a B_a$, 过采样率因子一般需满足 $\alpha_a \in [1.1, 1.4]$, 太低会引起方位模糊, 且 $\alpha_a > \alpha_r$.
2. 距离测绘带宽度: 即 $1/F_{sa} - T_p > T_{sr} = 2t_2 - 2t_1$, PRF 需足够低, 以使得测绘带内的脉冲回波都可以被采集到, 若过大, 不同脉冲回波出现在同一接收窗内, 会产生距离模糊, 若距离模糊过大且 PRF 不能降低, 需要减小俯仰方向的波束宽度 (减小了测绘带宽).
3. 接收窗起始时间: 在星载情况下, 某时刻发射的脉冲需要经过好几个脉冲间隔才能被接收, 这一被接收的时刻受 PRF 的影响.

4. 星下点回波: 来自星下点处的地面反射能量往往比较大, 导致图像上出现亮条纹(镜像反射且每个距离单元覆盖较大的面积) 距离模糊, 通过选择合适的 PRF 可以使得星下点回波不在接收窗内.

提示: 在星载情况下需要考虑这些准则, 机载情况下一般可以忽略.

7.5.2.1.5 SAR 成像模式

Advanced SAR Modes: e.g. TerraSAR-X

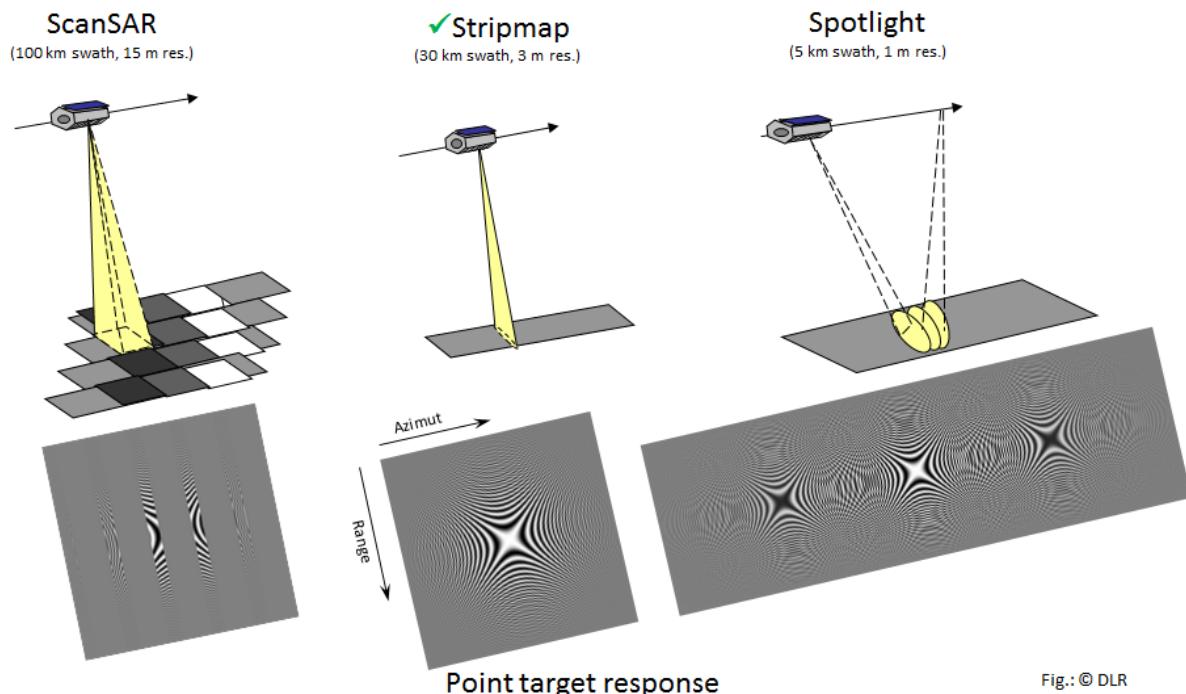


Fig.: © DLR

图 7.42: SAR 成像模式

SAR 成像模式

7.5.2.1.6 斑点噪声 (Speckle)

诸如 SAR、声纳这样的相干成像系统经常遭受一种称为散斑的乘性噪声。当相干辐射照射的物体具有与成像波长相比粗糙的表面时, 会出现散斑。它是由在每个分辨单元内的小反射器散射的相干回波的建设性和破坏性干涉引起的。斑点噪声严重影响后续目标辨识等性能, 通过在成像模型中引入正则可以减轻斑点噪声, 即对恢复图像施加正则, 常用的正则有, ℓ_1 , $\ell_{1/2}$, TV 正则等等 [3].

有关正则化技术参见:ref:“

7.5.2.1.7 公式总结

7.5.2.1.8 术语解释

7.5.2.1.8.1 Foot Print

雷达波束在地面上的脚印 (footprint), 即指真实孔径照射的区域. 分距离向与方位向的两种脚印.

距离向的脚印 (Along range footprint size) 大小可由下式近似计算

$$R_{AR} \approx \frac{\lambda}{L_r} \frac{H}{\cos^2 \theta_d}$$

方位向的脚印 (Cross range footprint size) 大小可由下式近似计算

$$R_{CR} \approx \frac{\lambda}{L_a} \frac{H}{\cos \theta_d}$$

其中, λ 为雷达发射波长, L_r, L_a 分别为天线距离向与方位向的天线孔径, H 为平台高度, θ_d 为雷达下视角.

7.5.2.2 SAR 系统组成

7.5.2.2.1 硬件系统

7.5.2.2.2 脉冲时序与采样

如图 7.47 与 图 7.45 所示, 设雷达于 t_k 时刻发射第 k 个脉冲波束, 脉冲宽度为 T_p . 设经过 t_{near} 时间, 发射脉冲前边缘于 $t_{A_k} = t_k + t_{near}$ 时刻到达近地点 A ; 经过 t_{far} 时间, 发射脉冲前边缘于 $t_{B_k} = t_k + t_{far}$ 时刻到达远地点 B , 前边缘到达远地点的脉冲又经过 $t_{far} + T_p$ 时间, 其后边缘到达雷达. 雷达接收机在稍早于 $2t_{A_k}$ 的时刻 $2t_{A_k} - \delta_{A_k}$ 开始采样, 并于稍晚于 $2t_{B_k} + T_p$ 的时刻 $2t_{B_k} + T_p + \delta_{B_k}$ 结束采样, 则采样时间为 $T_r = (2t_{B_k} + T_p + \delta_{B_k}) - (2t_{A_k} - \delta_{A_k})$. 其中, $\delta_{A_k} > 0, \delta_{B_k} > 0, (k = 0, 1, \dots, N_a)$ 为很小的正数, 故采样时间

$$T_r \approx 2t_{far} + T_p - 2t_{near}. \quad (7.1)$$

如图 7.45 所示, 第 $k+1$ 个脉冲必须在第 k 个脉冲到达雷达后才可以发射, 因而有

$$t_{k+1} + 2t_{near} > t_k + 2t_{far} + T_p,$$

从而有

$$T_{as} = \frac{1}{F_{as}} > 2t_{far} - 2t_{near} + T_p = T_r, \quad (7.2)$$

其中, F_{as} 为方位向采样频率, T_{as} 为方位向采样周期, T_r 为距离向采样时间.

表 4.3 主要合成孔径等式概览

参数	符号	表达式	单位
距离带宽		$ K_r T_r$	Hz
距离分辨率	ρ_r	$0.886 \gamma_{w,r} / (K_r T_r)$	m
斜距	$R(\eta)$	$\sqrt{R_0^2 + V_r^2 \eta^2}$	m
脉冲响应	$h_{\text{imp}}(\tau, \eta)$	$w_r(\tau - 2R(\eta)/c) w_a(\eta - \eta_c) \exp\{-j 4\pi f_0 R(\eta)/c\} \exp\{j\pi K_r (\tau - 2R(\eta)/c)^2\}$	
方位向波束宽度	θ_{bw}	$0.886 \lambda / L_a$	rad
方位向波束覆盖区	ρ'_a	$0.886 R(\eta_c) \lambda / L_a$	m
合成孔径	L_s	$[0.886 R(\eta_c) \lambda / (L_a)] (V_s / V_g)$	m
合成角	θ_{syn}	$(V_s / V_g) \theta_{\text{bw}}$	rad
$\eta = \eta_c$ 时的多普勒频率	f_{η_c}	$2 V_r \sin \theta_{r,c} / \lambda = 2 V_s \sin \theta_{sq,c} / \lambda$	Hz
波束中心穿越时刻	η_c	$-R_0 \tan \theta_{r,c} / V_r = -R_0 \tan \theta_{sq,c} / V_g$	s
多普勒带宽	Δf_{dop}	$0.886 (2 V_s \cos \theta_{r,c} / L_a)$	Hz
照射时间	T_a	$0.886 R(\eta_c) \lambda / (L_a V_g \cos \theta_{r,c})$	s
方位向调频率	K_a	$2 V_r^2 \cos^2 \theta_{r,c} / [\lambda R(\eta_c)]$	Hz/s
方位向分辨率	ρ_a	$(0.886 V_g \cos \theta_{r,c} / \Delta f_{\text{dop}}) \gamma_{w,a}$	m
方位向分辨率	ρ_a	$(L_a / 2) (V_g / V_s) \gamma_{w,a} \approx L_a / 2$	m
方位向分辨率	ρ_a	$[0.886 \lambda / (2 \theta_{\text{syn}})] \gamma_{w,a}$	m
距离压缩比	$C_{r,r}$	$ K_r T_r^2$	
方位压缩比	$C_{r,a}$	$K_a T_a^2$	
总的压缩比	$C_{r,t}$	$ K_r T_r^2 K_a T_a^2$	

图 7.43: 合成孔径相关公式

表 4.1 机载和星载 SAR 的典型参数

参数	符号	机载	星载	单位
距离向参数				
景中心斜距	$R(\eta_c)$	30	850	km
高度		10	800	km
发射脉冲时宽	T_r	10	40	μs
距离脉冲调频率	K_r	10	0.5	MHz/ μs
信号带宽		100	20	MHz
距离采样率	F_r	120	24	MHz
斜距条带宽度		10	50	km
方位向参数				
雷达有效速度 ⁽¹⁾	V_r	250	7100	m/s
雷达工作频率	f_0	9.4	5.3	GHz
雷达工作波长	λ	0.032	0.057	m
方位调频率	K_a	131	2095	Hz/s
合成孔径长度 ⁽²⁾	L_s	0.85	4.8	km
目标照射时间	T_a	3.4	0.64	s
天线长度	L_a	1	10	m
多普勒带宽	Δf_{dop}	443	1338	Hz
方位采样率 (PRF)	F_a	600	1700	Hz
斜视角 ⁽³⁾	$\theta_{r,c}$	< 8	< 4	度

注：

(1) 在星载情况下， V_s 比 V_r 大 6%，而 V_g 比 V_r 小 6%

(2) 参见 4.7.2 节

(3) $\theta_{r,c} \approx \theta_{sq,c}$

图 7.44: 相关参数说明

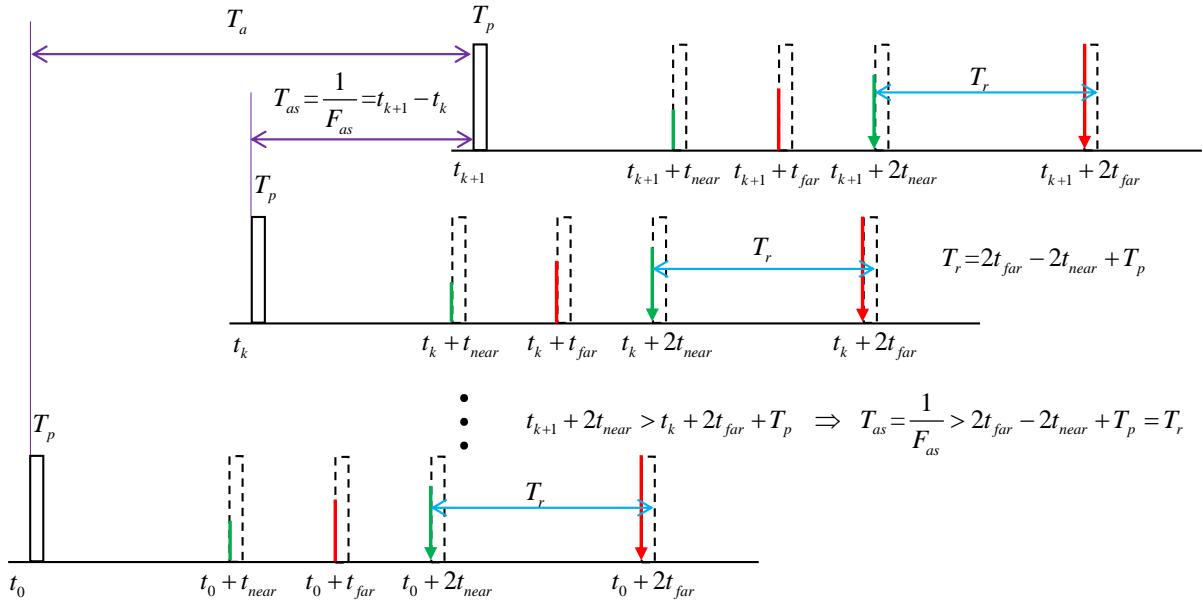


图 7.45: Time sequence of SAR transmitting and receiving pulse.

7.5.2.2.3 几何建模

7.5.2.2.3.1 关键参数计算

7.5.2.2.3.2 卫星速度与波束掠过地面的速度

如图 7.46 所示, 地球半径为 R_e , 卫星在距离地面高度为 H 的高空以速度 V_s 围绕地球运动, 下视角为 θ_d , 波束掠过地面的速度为 V_g . 由图 7.46 中几何关系知 $x + AB\tan\theta_d = R_e + H$, 即

$$x + \sqrt{R_e^2 - x^2} \tan\theta_d = R_e + H$$

化简后得

$$(1 + \tan^2\theta_d)x^2 - 2(R_e + H)x + (R_e + H)^2 - R_e^2\tan^2\theta_d = 0 \quad (7.3)$$

式 7.3 的解为

$$x = \frac{(R_e + H) \pm \tan\theta_d \sqrt{R_e^2\tan^2\theta_d - 2HR_e - H^2}}{1 + \tan^2\theta_d}, \quad (7.4)$$

式 7.4 中, 当 $0 < \theta_d < \pi/2$ 时, 分子取加号; 当 $\pi/2 < \theta_d < \pi$ 时, 分子取减号.

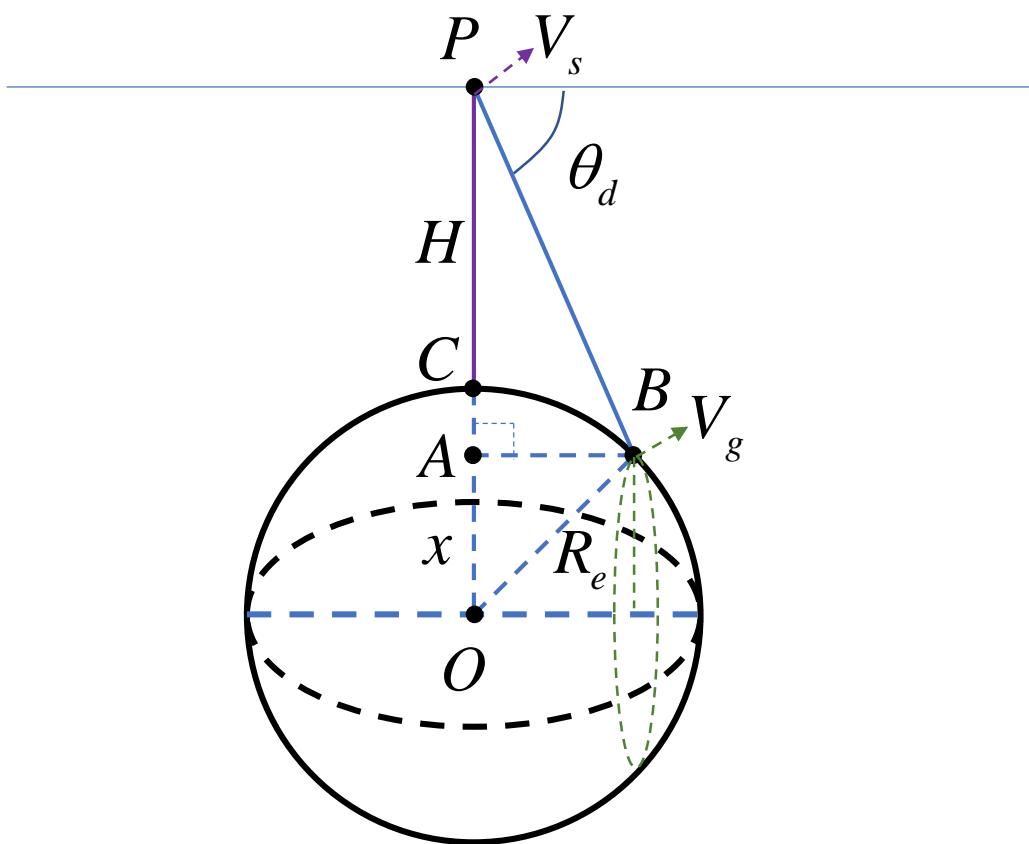


图 7.46: Satellite velocity and beam skimming velocity

7.5.2.2.3.3 距离向波束宽度计算

根据采样点数 N_r , 采样率 F_{rs} , 平台高度 H 以及下视角 θ_d , 可以计算距离向波束宽度 (antenna elevation beamwidth) β_e . 如 图 7.47 所示, 由 图 7.47 中几何关系及式.7.1 得

$$\begin{aligned} T_r - T_p &= \frac{N_r - N_p}{F_{rs}} \\ &\approx 2t_{far} - 2t_{near} \\ &= \frac{2PB}{c} - \frac{2PA}{c} \\ &= \frac{2H}{c \cdot \sin(\theta_d - \beta_e/2)} - \frac{2H}{c \cdot \sin(\theta_d + \beta_e/2)} \end{aligned} \quad (7.5)$$

其中, c 为光速, N_r 为距离向采样点数, N_p 为 T_p 时间内的距离向采样点数. 式.7.5 代入积化和差 $\sin \alpha \sin \beta = \frac{1}{2}[\cos(\alpha - \beta) - \cos(\alpha + \beta)]$ 与和差化积公式 $\sin \alpha - \sin \beta = 2 \cos \frac{\alpha + \beta}{2} \sin \frac{\alpha - \beta}{2}$, 式.7.5 化简后得到

$$a \cdot \sin^2 \frac{\beta_e}{2} + 4H \cos \theta_d \sin \frac{\beta_e}{2} + \frac{a}{2} (\cos 2\theta_d - 1) = 0$$

即

$$\left(\sin \frac{\beta_e}{2} + \frac{2H \cos \theta_d}{a} \right)^2 = a \cdot \sin^2 \theta_d + \frac{4H^2 \cos^2 \theta_d}{a}$$

从而解得

$$\sin \frac{\beta_e}{2} = -\frac{2H \cos \theta_d}{a} \pm \sqrt{\sin^2 \theta_d + \frac{4H^2 \cos^2 \theta_d}{a^2}},$$

其中, $a = \frac{c(N_r - N_p)}{F_{rs}}$, c 为光速. 由于 $\sin \frac{\beta_e}{2} \in [-1, 1]$, 可以排除一个解.

7.5.2.3 线性调频信号

7.5.2.3.1 线性调频信号

线性频率调制信号 (Linear Frequency Modulated, LFM) 简称 线性调频信号, 之所以称之为 线性调频是因为 其频率随时间呈线性变化, 数学表示为:

$$p(t) = \text{rect}\left(\frac{t}{T}\right) \exp\left\{j\pi Kt^2\right\},$$

其中, t 为时间, 单位为 s, K 为线性调频率, 可正可负, 单位为 Hz/s, 脉冲相位 $\phi(t) = \pi Kt^2$, $\text{rect}\left(\frac{t}{T}\right)$ 为矩形窗函数, 只是对信号加了一定大小的窗, 窗大小由 T 决定, 矩形窗函数的表达式如下:

$$\text{rect}(t) = \begin{cases} 0, & |t| > 1/2 \\ 1/2, & |t| = 1/2 \\ 1, & |t| < 1/2 \end{cases}$$

脉冲相位 $\phi(t) = \pi Kt^2$ 对时间取微分得到

$$f = \frac{d(\phi(t)/2\pi)}{dt} = Kt,$$

即频率随时间线性变化. LFM 信号的带宽为 $B = |K|t$.

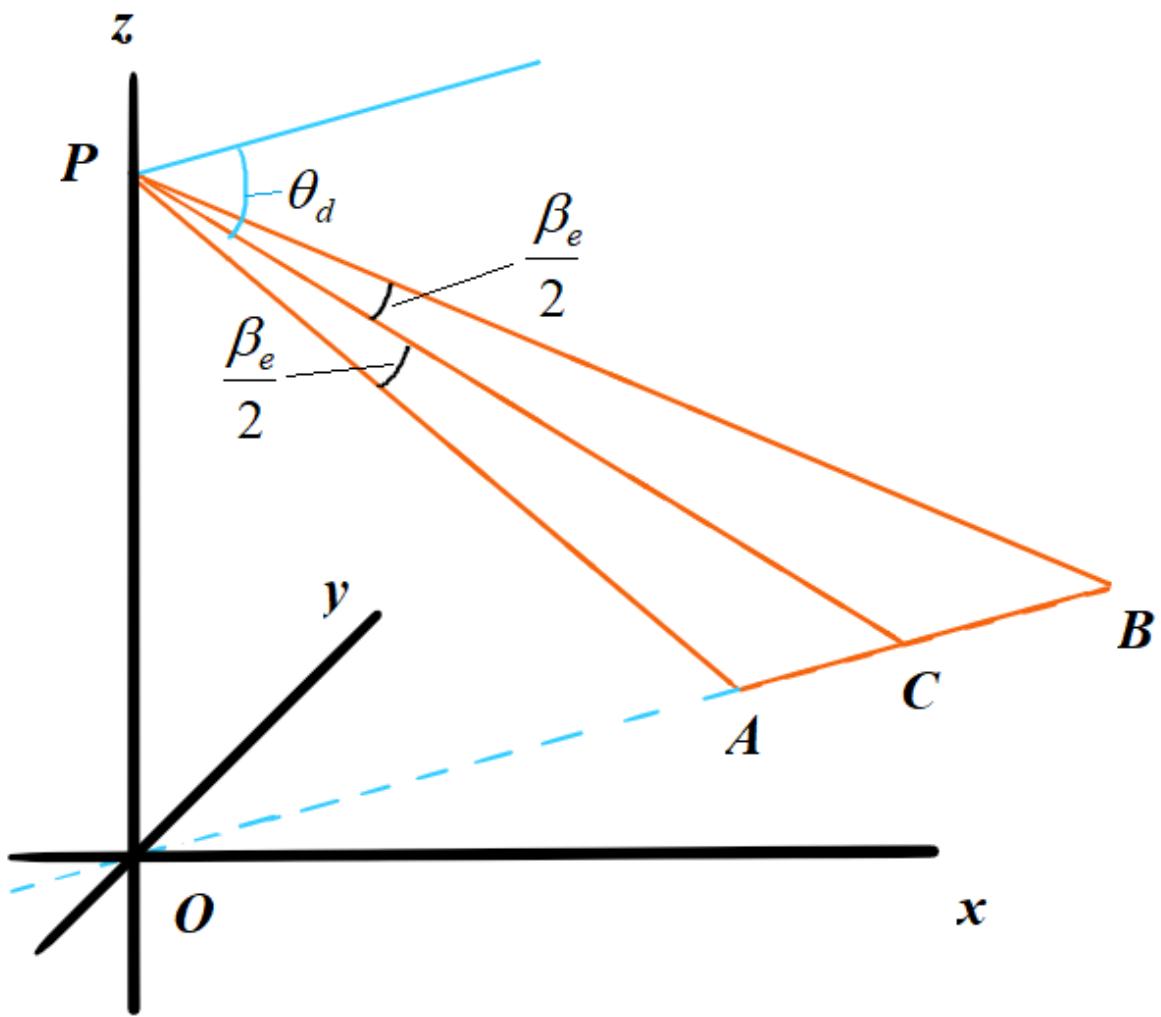


图 7.47: Beam width and footprint geometry in range direction

7.5.2.3.2 解调后的信号

7.5.2.3.3 实验仿真

7.5.2.3.3.1 生成分析 LFM

7.5.2.3.3.2 实验代码

python 代码如下

```

import matplotlib.pyplot as plt
import numpy as np

def rect(x):
    """
    Rectangle function:
    rect(x) = {1, if |x|<= 0.5; 0, otherwise}
    """
    return np.where(np.abs(x) >= 0.5, 0, 1)

def chirp(t, T, Kr):
    """
    Create a chirp signal :
    S_{tx}(t) = rect(t/T) * exp(1j*pi*Kr*t^2)
    """
    return rect(t / T) * np.exp(1j * np.pi * Kr * t ** 2)

Ns = 1000
t = np.linspace(-1, 1, Ns)
yrect = rect(t)

plt.figure()
plt.plot(t, yrect, '-r', linewidth=2)
plt.axis([-1, 1, -0.1, 1.1])
plt.grid()
plt.xlabel('Time/s')
plt.ylabel('Amplitude')
plt.title('rect')
plt.show()

Kr1 = 5
Kr2 = 20
T1 = 2.0
T2 = 4.0

```

(下页继续)

(续上页)

```

t = np.linspace(-3, 3, Ns)
y chirp1 = chirp(t, T1, Kr=Kr1)
y chirp2 = chirp(t, T2, Kr=Kr1)

y chirp3 = chirp(t, T1, Kr=Kr2)
y chirp4 = chirp(t, T2, Kr=Kr2)

plt.figure()
plt.subplot(221)
plt.grid()
plt.plot(t, y chirp1, '-r')
plt.xlabel('Time/s')
plt.ylabel('Amplitude')
plt.title("T=" + str(T1) + "s, K=" + str(Kr1) + "Hz/s")
plt.subplot(222)
plt.grid()
plt.plot(t, y chirp2, '-b')
plt.xlabel('Time/s')
plt.ylabel('Amplitude')
plt.title("T=" + str(T2) + "s, K=" + str(Kr1) + "Hz/s")
plt.subplot(223)
plt.grid()
plt.plot(t, y chirp3, '-r')
plt.xlabel('Time/s')
plt.ylabel('Amplitude')
plt.title("T=" + str(T1) + "s, K=" + str(Kr2) + "Hz/s")
plt.subplot(224)
plt.grid()
plt.plot(t, y chirp4, '-b')
plt.xlabel('Time/s')
plt.ylabel('Amplitude')
plt.title("T=" + str(T2) + "s, K=" + str(Kr2) + "Hz/s")
plt.tight_layout()
plt.show()

# =====compute spectral

y chirp1_fft = np.fft.fft(y chirp1)
y chirp2_fft = np.fft.fft(y chirp2)
y chirp3_fft = np.fft.fft(y chirp3)
y chirp4_fft = np.fft.fft(y chirp4)

y chirp1_fft = np.fft.fftshift(y chirp1_fft)
y chirp2_fft = np.fft.fftshift(y chirp2_fft)
y chirp3_fft = np.fft.fftshift(y chirp3_fft)
y chirp4_fft = np.fft.fftshift(y chirp4_fft)

```

(下页继续)

(续上页)

```
f = np.fft.fftfreq(t.shape[-1])
# print(f)
f = np.fft.fftshift(f)
# print(f)

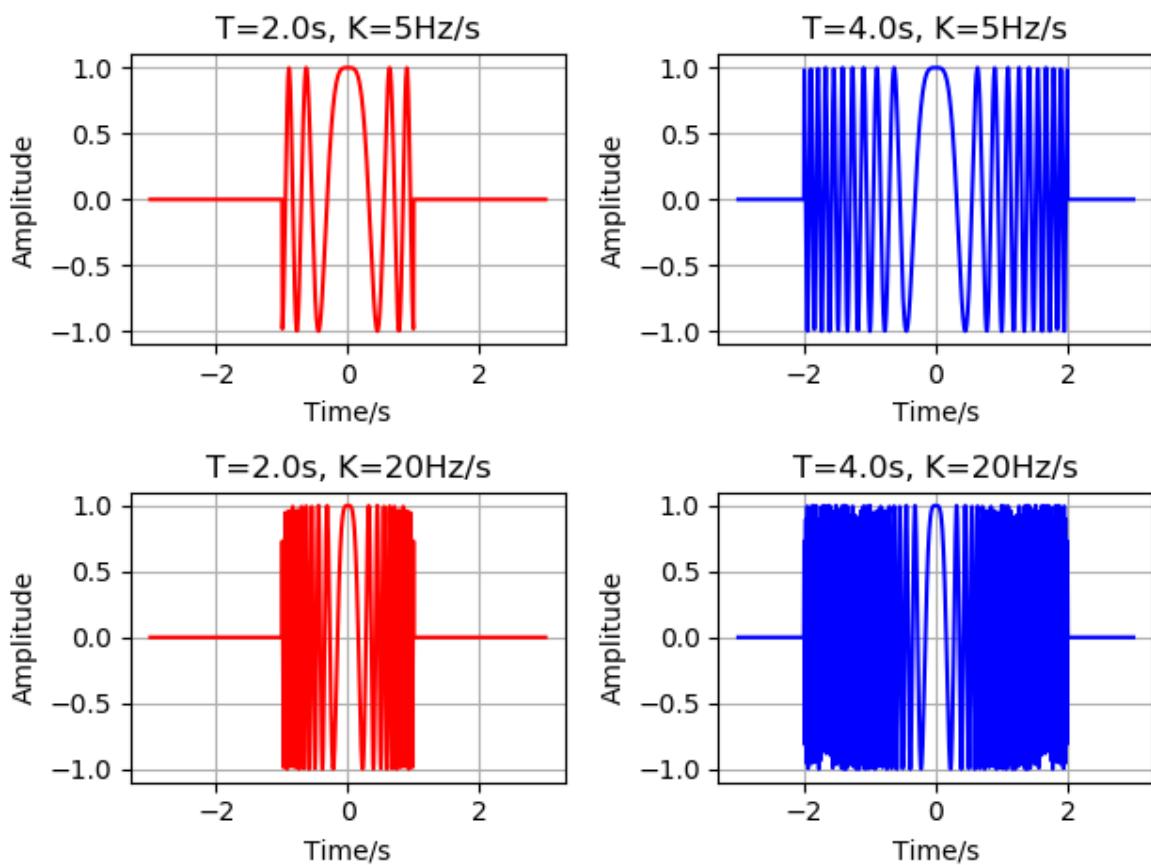
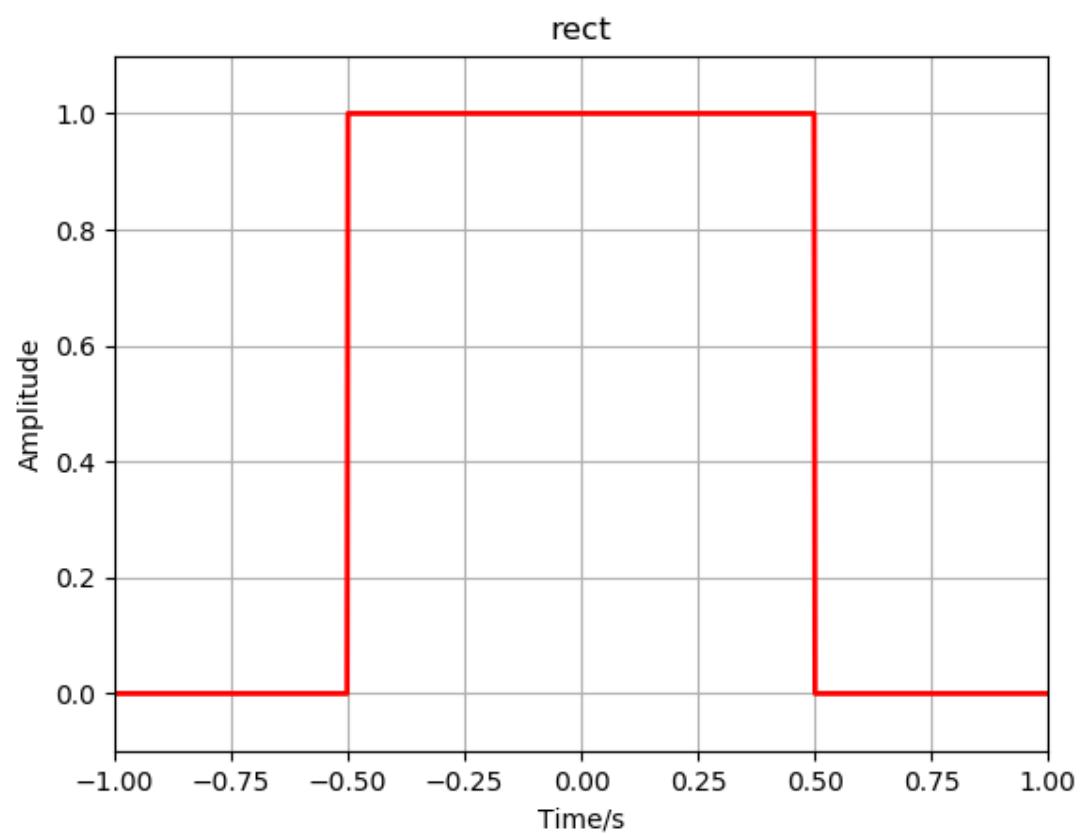
plt.figure()
plt.subplot(221)
plt.grid()
plt.plot(f, np.abs(ychirp1_fft), '-r')
plt.xlabel('Frequency/Hz')
plt.ylabel('Amplitude')
plt.title("T=" + str(T1) + "s, K=" + str(Kr1) + "Hz/s")
plt.subplot(222)
plt.grid()
plt.plot(f, np.abs(ychirp2_fft), '-b')
plt.xlabel('Frequency/Hz')
plt.ylabel('Amplitude')
plt.title("T=" + str(T2) + "s, K=" + str(Kr2) + "Hz/s")
plt.subplot(223)
plt.grid()
plt.plot(f, np.abs(ychirp3_fft), '-r')
plt.xlabel('Frequency/Hz')
plt.ylabel('Amplitude')
plt.title("T=" + str(T1) + "s, K=" + str(Kr1) + "Hz/s")
plt.subplot(224)
plt.grid()
plt.plot(f, np.abs(ychirp4_fft), '-b')
plt.xlabel('Frequency/Hz')
plt.ylabel('Amplitude')
plt.title("T=" + str(T2) + "s, K=" + str(Kr2) + "Hz/s")
plt.tight_layout()
plt.show()
```

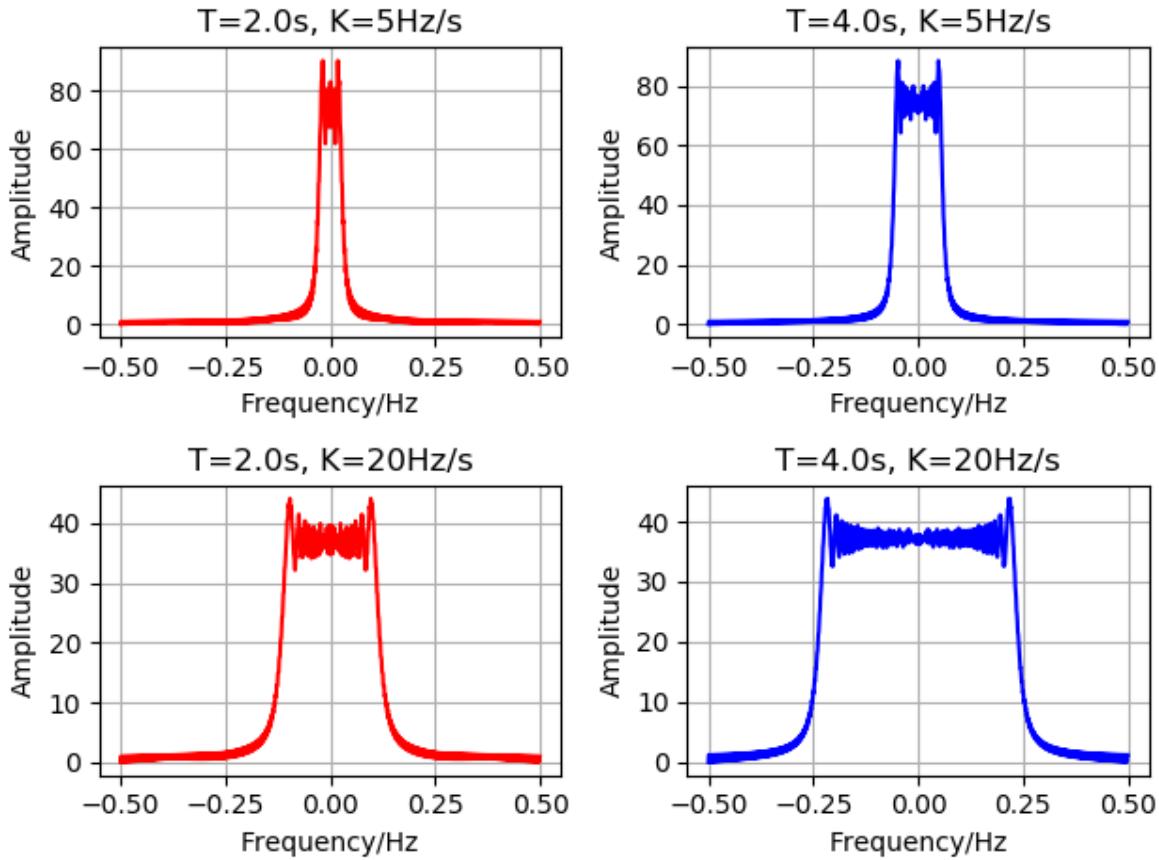
7.5.2.3.3 实验结果

生成的 rect 信号:

生成的 chirp 线性调频信号:

生成的 chirp 线性调频信号的幅度谱:





7.5.2.4 SAR 回波信号及其性质

7.5.2.4.1 SAR 回波信号

假设, 在距离向雷达发射信号为:

$$s_t(\tau) = p(\tau)\exp(j2\pi f_0\tau),$$

其中, f_0 为发射信号中心频率, τ 是距离向时间, 也称快时间. $p(\tau)$ 为线性调频信号, 通常为

$$p(\tau) = w_r(\tau)\exp(j\pi K_r\tau^2),$$

其中, K_r 为距离向调频斜率, T_p 为脉冲持续时间, $w_r(\tau) = \text{rect}\left(\frac{\tau}{T_p}\right)$ 为脉冲包络.

在方位向, 接收信号的强度可以表示为方位时间的函数¹:

$$w_a(\eta) = P_a^2(\theta(\eta)) = \text{sinc}^2\left(\frac{0.886\theta(\eta)}{\beta_{bw}}\right),$$

其中, $\theta(\eta)$ 为方位向 η 时刻斜距平面内测得的 (slant range) 与视线的夹角 (即 η 时刻目标与波束中心线的夹角, 不是斜视角, 显然目标处于波束中心时, 信号强度最大), $\beta_{bw} = 0.886\lambda/L_a$ 为方位向波束宽度, L_a 为天线方位向宽度.

以零多普勒时间 (*Zero Doppler Time*) 为参考, 记 η_c 为波束中心经过目标的时刻, 在斜视角不为零时, η_c 不

¹ 参见《合成孔径雷达成像——算法与实现》p91 页.

为零(前视时 $\eta_c < 0$, 后视时 $\eta_c > 0$)

$$\eta_c = -\frac{R_0 \tan \theta_{r,c}}{V_r} = -\frac{R(\eta_c) \sin \theta_{r,c}}{V_r} = -\frac{R_0 \tan \theta_{s,c}}{V_g} = -\frac{R(\eta_c) \sin \theta_{s,c}}{V_g}.$$

其中, $R(\eta_c)$ 为目标被波束中心线照射时, 雷达与目标的斜距, $\theta_{s,c}$ 为 η_c 时刻 θ_s 的值.

注解: 多普勒中心频率是指 $\eta = \eta_c$ 时刻的多普勒频率

$$\begin{aligned} f_{\eta_c} &= -\frac{2}{\lambda} \frac{dR(\eta)}{d\eta} \Big|_{\eta=\eta_c} \\ &= -\frac{2V_r^2 \eta_c}{\lambda R(\eta_c)} \\ &= \frac{2V_r \sin \theta_{r,c}}{\lambda} \\ &= \frac{2V_s \sin \theta_{s,c}}{\lambda}, \end{aligned}$$

其中, $\theta_{s,c}, \theta_{r,c}$ 分别为 η_c 时刻, 基于轨道几何与直线几何的斜视角(参见《合成孔径雷达成像算法与实现》 p84). 且 θ_s, θ_r 与方位向慢时间的关系为

$$\begin{aligned} \sin \theta_s &= \frac{Y_g}{R(\eta)} = -\frac{V_g \eta}{R(\eta)} \\ \sin \theta_r &= \frac{Y_r}{R(\eta)} = -\frac{V_r \eta}{R(\eta)} \end{aligned}$$

其中, V_g, θ_s, Y_g 为地球弯曲几何中的变量, V_r, θ_r, Y_r 为直线几何中的变量, 且有 $\theta_r = \frac{V_r}{V_g} \theta_s = \frac{V_s}{V_r} \theta_s$. 对于机载 SAR, 两种几何模型之间的差异可以忽略, 等效为直线几何.

低斜视角时, $f_{\eta_c} \approx -K_a \eta_c$.

这种情况下 $\theta(\eta) = \theta_s - \theta_{s,c}$, 且有

$$w_a(\eta - \eta_c) = P_a^2(\theta_s - \theta_{s,c}) \approx P_a^2 \left(\arctan \left(\frac{V_g(\eta - \eta_c)}{R_0} \right) \right).$$

其中, V_g 为波束覆盖面沿着地球表面移动的速度, 设 V_s 为 SAR 平台沿轨道的运行的速度, 即平台实际物理速度; V_r 为等效雷达速度, 即按两点间直线距离计算出的等效的速度, 则

$$V_r \approx \sqrt{V_s V_g}, V_g < V_r < V_s.$$

时刻 η 雷达与目标的斜距可表示为:

$$R^2(\eta) = R_0^2 + V_r^2 \eta^2.$$

接收的回波信号为二维的线性调频信号, 点目标回波表达式为:

$$\begin{aligned} s(\eta, \tau) &= g(\eta, \tau) w_a(\eta - \eta_c) w_r(\tau - 2R(\eta)/c) \\ &\quad \exp \left(2\pi f_0(\tau - 2R(\eta)/c) + \pi K_r (\tau - 2R(\eta)/c)^2 \right) \end{aligned} \tag{7.6}$$

7.5.2.4.2 回波信号解调

载频不含目标信息, 解调便是将天线接收的信号中的载频成分移除. 在雷达接收系统中, 常采用正交解调的方法, 其原理图见 图 7.48 . 天线接收的信号 ① 分别与 $\cos(2f_c t)$, $-\sin(2f_c t)$ 相乘将有用信号与载频信号分离, 得到信号 ② 和 ③; 分离开的信号经低通滤波滤出高频 (载频) 信号, 从而得到有用信号 ④ 和 ⑤; 信号 ④ 和 ⑤ 分别作为实部虚部合成复数信号 ⑥, ⑦ 为 ⑥ 的离散化采样形式.

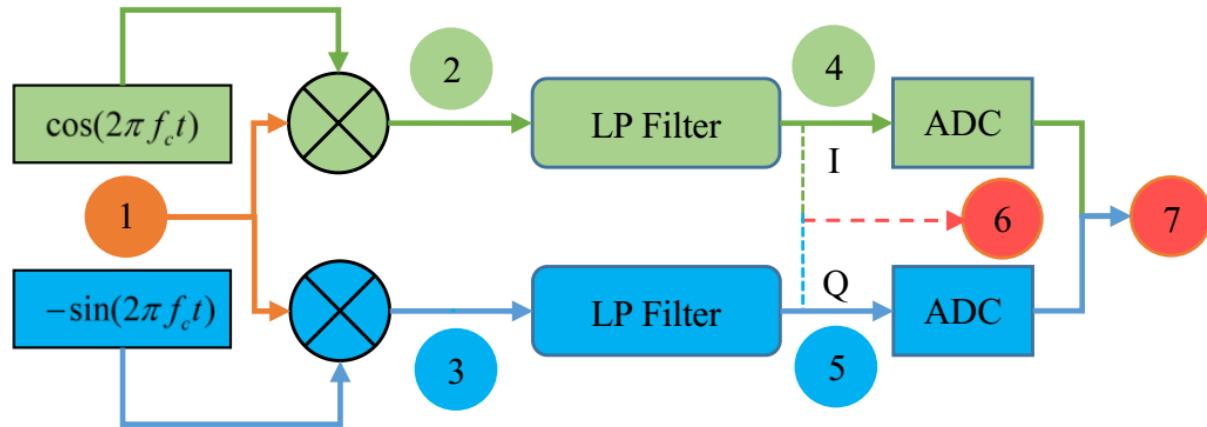


图 7.48: SAR IQ Demodulation

$$\textcircled{1} \cos\left(2\pi f_c \tau - \frac{4\pi f_c R(\eta)}{c} + \pi K_r \left(\tau - \frac{2R(\eta)}{c}\right)^2\right) = \cos(2\pi f_c \tau + \phi(\eta, \tau))$$

$$\textcircled{2} \frac{1}{2} \cos[\phi(\eta, \tau)] + \frac{1}{2} \cos[4\pi f_c \tau + \phi(\eta, \tau)]$$

$$\textcircled{3} \frac{1}{2} \sin[\phi(\eta, \tau)] + \frac{1}{2} \sin[4\pi f_c \tau + \phi(\eta, \tau)]$$

$$\textcircled{4} \frac{1}{2} \cos[\phi(\eta, \tau)]$$

$$\textcircled{5} \frac{1}{2} \sin[\phi(\eta, \tau)]$$

$$\textcircled{6} \frac{1}{2} \exp\{j\phi(\eta, \tau)\} = \frac{1}{2} \exp\left\{-j\frac{4\pi f_c R(\eta)}{c} + j\pi K_r \left(\tau - \frac{2R(\eta)}{c}\right)^2\right\}$$

式.7.6 解调后的 SAR 信号可以表示为:

$$s(\eta, \tau) = g(\eta, \tau) w_a(\eta - \eta_c) w_r (\tau - 2R(\eta)/c) \exp\left\{-j4\pi f_0 R(\eta)/c + j\pi K_r (\tau - 2R(\eta)/c)^2\right\} \quad (7.7)$$

其中系数 $g(\eta, \tau) = G' \exp(j\phi)$ 为复常数, G' 为强度. 然后对距离向信号进行采样, 可以得到 SAR 原始数据. 上式表示的是从场景反射系数 $g(\eta, \tau)$ 的点目标接收到的经解调后的 SAR 基带信号, 也是 SAR 系统进行记录与下传的信号, 称为 **SAR 原始数据** (SAR raw data), **SAR 信号数据** (SAR signal data), **SAR 相位历史数据** (SAR phase history data).

若场景中含 N 个目标, 则接收到的 SAR 数据为各目标回波的叠加

$$s(\eta, \tau) = \sum_{i=1}^N s_i(\eta, \tau)$$

若考虑系统噪声, 则有

$$s(\eta, \tau) = \sum_{i=1}^N s_i(\eta, \tau) + n(\eta, \tau)$$

提示: 方位向多普勒带宽, 调频率及目标照射时间, 参见¹ p93 页.

7.5.2.4.3 SAR 冲击响应

SAR 的回波信号可以看作是地面反射率 $g(\eta, \tau)$ 与雷达系统冲激响应 $h(\eta, \tau)$ 进行二维卷积, 即:

$$\begin{aligned} s(\eta, \tau) &= g(\eta, \tau) \otimes h(\eta, \tau) + n(\eta, \tau) \\ &= \iint g(u, v) \cdot h(\eta - u, \tau - v) du dv + n(\eta, \tau) \\ &= g(\eta, \tau) w_a(\eta - \eta_c) w_r (\tau - 2R(\eta)/c) \\ &\quad \exp \left\{ -j4\pi f_0 R(\eta)/c + j\pi K_r (\tau - 2R(\eta)/c)^2 \right\} + n(\eta, \tau) \end{aligned} \quad (7.8)$$

其中, $n(\tau, \eta)$ 为系统噪声. 雷达二维冲激响应为

$$\begin{aligned} h(\eta, \tau) &= w_a(\eta - \eta_c) w_r (\tau - 2R(\eta)/c) \\ &\quad \exp \left\{ -j4\pi f_0 R(\eta)/c + j\pi K_r (\tau - 2R(\eta)/c)^2 \right\} \end{aligned} \quad (7.9)$$

因此不管是点目标还是面目标都可以通过 SAR 的接收信号的一般模型求得回波表达式, SAR 的成像处理过程, 其实际上也是一个通过解卷积从回波信号中最大程度地、无失真地提取地表的后向散射系数的二维分布.

冲激响应 $h(\eta, \tau)$ 可以表示成距离向上的冲激响应 $h_r(\eta, \tau)$ 和方位向上的冲激响应 $h_a(\eta, \tau)$ 之积, 即

$$h(\eta, \tau) = h_a(\eta, \tau) h_r(\eta, \tau) = h_r(\eta, \tau) h_a(\eta, \tau)$$

则式.7.8 可以表示为地面反射系数与两个方向上的冲激响应的一维级联卷积

$$\begin{aligned} s(\eta, \tau) &= g(\eta, \tau) \otimes h(\eta, \tau) \\ &= g(\eta, \tau) * h_a(\eta, \tau) * h_r(\eta, \tau) \\ &= g(\eta, \tau) * h_r(\eta, \tau) * h_a(\eta, \tau) \end{aligned} \quad (7.10)$$

其中, \otimes 表示二维卷积, $*$ 表示一维卷积.

方位向上的冲激响应为

$$\begin{aligned} h_a(\eta, \tau) &= w_a(\eta - \eta_c) \\ &\quad \exp \left\{ -j4\pi f_0 R(\eta)/c \right\} \end{aligned} \quad (7.11)$$

距离向上的冲激响应为

$$\begin{aligned} h_r(\eta, \tau) &= w_r (\tau - 2R(\eta)/c) \\ &\quad \exp \left\{ j\pi K_r (\tau - 2R(\eta)/c)^2 \right\} \end{aligned} \quad (7.12)$$

若用 S, G, H 分别表示 SAR 回波的频域信号, 地面目标后向散射系数的频域信号, 二维冲激响应的频域信号, 则

$$S = GH. \quad (7.13)$$

式.7.13 说明, 可以通过频域相乘来简化和加速 SAR 回波仿真过程. 其过程可以简要叙述为

注解: 频域 SAR 信号模拟生成

输入: 时域场景反射系数矩阵 $\mathbf{G}_{H \times W}$, 时域二维冲激响应矩阵 \mathbf{H}

输出: 时域 SAR 回波信号矩阵 $\mathbf{S}_{N_a \times N_r}$

Step 1: 对时域场景反射系数矩阵 $\mathbf{G}_{H \times W}$ 做二维傅里叶变换, 得到其频域二维形式 $G = \text{FFT}(\mathbf{G})$

Step 2: 对时域二维冲激响应矩阵 \mathbf{H} 做二维傅里叶变换, 得到其频域二维形式 $H = \text{FFT}(\mathbf{H})$ Step3: 通过频域相乘得到频域 SAR 回波信号 $S = GH$

Step4: 通过逆傅里叶变换得到时域 SAR 回波信号矩阵 $\mathbf{S} = \text{IFFT}(S)$

7.5.2.4.4 SAR 回波信号的频谱

7.5.2.4.4.1 距离维频谱

7.5.2.4.4.2 方位维频谱

7.5.2.4.4.3 距离多普勒域频谱

7.5.2.4.4.4 二维频谱

7.5.2.5 脉冲压缩

7.5.3 SAR 仿真技术

7.5.3.1 回波信号模型

7.5.3.1.1 SAR 回波信号

7.5.3.1.1.1 目标回波分析

由图中几何关系知: $R(\eta)^2 = R_0^2 + (V\eta)^2$, 在低斜视角下, $R(\eta)$ 可由菲涅尔近似为: $R(\eta) = R_0 + \frac{(V\eta)^2}{2R_0}$
接收的回波信号为二维的线性调频信号, 点目标回波表达式为:

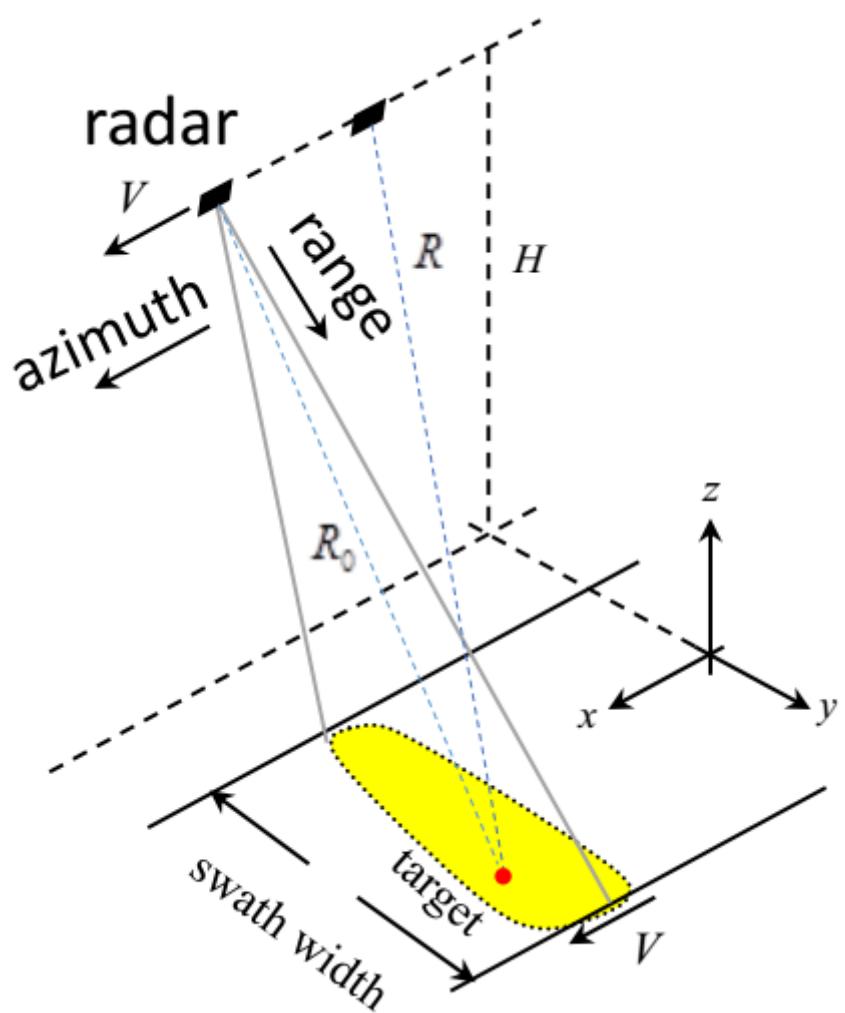
$$s_r(\tau, \eta) = A_0 w_r(\tau - 2R(\eta)/C) w_a(\eta - \eta_c) \\ \exp \left(2\pi f_0 (\tau - 2R(\eta)/C) + \pi K_r (\tau - 2R(\eta)/C)^2 \right)$$

去除载频后的基带信号为:

$$s_r(\tau, \eta) = A_0 w_r(\tau - 2R(\eta)/C) w_a(\eta - \eta_c) \\ \exp \left(-j4\pi f_0 R(\eta)/C + j\pi K_r (\tau - 2R(\eta)/C)^2 \right)$$

其中:

- A_0 为复数, 代表点目标的后向散射引起的一个相位和幅度变化, $A_0 = A'_0 \exp(j\phi)$, A'_0 为后向散射系数, ϕ 为地表散射引起的雷达信号相位的改变



- τ 为距离向时间
- η 为方位向时间, η_c 为零多普勒时间
- $w_a(\cdot)$ 为方位向天线增益, $w_r(\cdot)$ 为距离向天线增益
- f_0 为载频

通常取全向天线增益为 1, 则回波变为:

$$s_r(\tau, \eta) = A_0 \exp \left(-j4\pi f_0 R(\eta)/C + j\pi K_r (\tau - 2R(\eta)/C)^2 \right)$$

二维场景离散化:

$$s(t_{a,n_a}, t_{r,n_r}) = \sum_{i=1}^H \sum_{j=1}^W g(i, j) \cdot p(t_{r,n_r} - 2R(t_{a,n_a}, i, j)/c) \exp(-j4\pi f_c R(t_{a,n_a}, i, j)/c),$$

7.5.3.1.1.2 目标回波基础理论

SAR 的回波数据可以看作是地面反射率 $\gamma(\tau, \eta)$ 与雷达系统冲激响应 $h(\tau, \eta)$ 进行二维卷积, 即:

$$S(\tau, \eta) = \gamma(\tau, \eta) \otimes h(\tau, \eta) + n(\tau, \eta)$$

其中, $n(\tau, \eta)$ 为系统噪声. 因此不管是点目标还是面目标都可以通过 SAR 的接收信号的一般模型求得回波表达式, SAR 的成像处理过程, 其实际上也是一个通过解卷积从回波信号中最大程度地、无失真地提取地表的后向散射系数的二维分布

7.5.3.1.2 时域回波仿真

在得到点目标回波信号后, 传统的时域回波模拟方法也叫距离时域脉冲相干算法是将一个目标场景图像细化成一个点阵, 直接模拟雷达的工作过程, 在雷达平台方位向的每一个位置, 先计算出天线波束照射范围内即距离向每个点目标的回波信号, 然后将这些信号叠加得到最终的回波信号.

7.5.3.1.2.1 点目标

单个点目标回波的表达式即为上述点目标回波表达式, 多个点目标可以看成每个点目标回波的叠加:

$$s(\tau, \eta) = \sum_{i=1}^n s_i(\tau, \eta)$$

其中, n 为点目标的个数, $s_i(\tau, \eta)$ 为点目标回波.

7.5.3.1.2.2 面目标

对于面目标就是把目标场景分割成一个均匀分布的点阵。这个是相对精确地面目标时域回波仿真方法，但是在面目标相对较大时，那么它被分割成的点阵数量就多，相应的数据量就大，计算量也高。

7.5.3.2 SAR 回波频谱分析

7.5.3.2.1 仿真数据

7.5.3.2.2 真实数据

7.5.4 SAR 成像理论

7.5.4.1 SAR 成像简介

注解： 目前压缩感知重构算法分为两大类：

1. 贪婪算法：通过迭代选择合适的原子使得重构误差最小，包括匹配追踪，正交匹配追踪，补空间匹配追踪等
2. 凸优化：通过放宽正则约束（如 $\ell_0 \rightarrow \ell_1, \ell_2$ ）转化为凸优化问题，包括梯度投影法，基追踪，最小角回归

凸优化方法比贪婪算法所求解更为精确，但计算复杂度高。

提示： 压缩感知与正则化成像：

1. 压缩感知强调压缩观测，在信号不稀疏时先进行稀疏表示，再求解
 2. 正则化方法强调通过加入先验，使得问题有极小范数解
 3. 优化方法可以一致
-

7.5.4.1.1 不同成像方法结果对比

- 仿真场景大小: 32×32
- 回波矩阵大小: 32×32
- 数据生成
 - 通过模拟 SAR 成像过程生成 SAR 原始数据 s
 - 通过 $s = Ag$ 生成 SAR 原始数据 s
- 成像方法

- 广义逆: $\hat{\mathbf{g}} = \mathbf{A}^+ \mathbf{s}$
- 共轭转置: $\hat{\mathbf{g}} = \mathbf{A}^H \mathbf{s}$
- 距离多普勒方法: 匹配滤波, 距离徙动校正等.

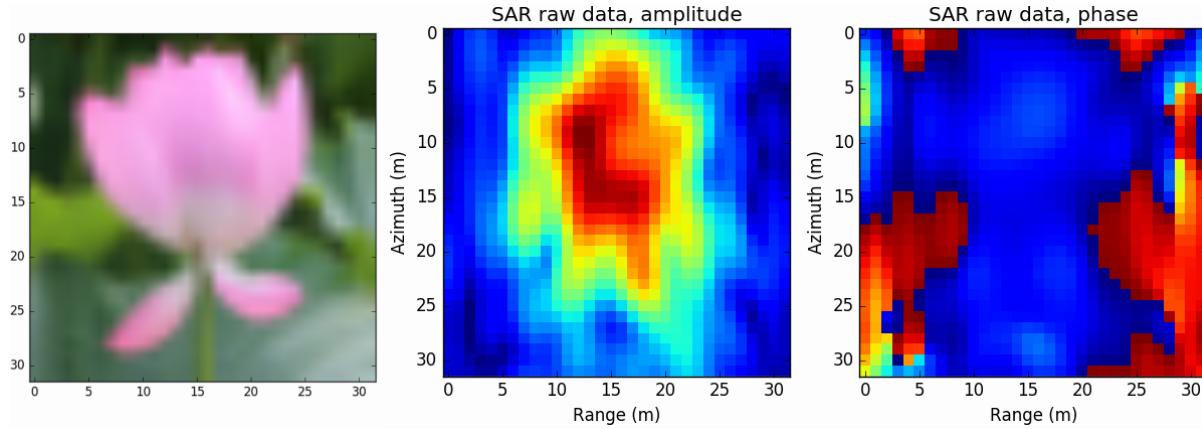


图 7.49: 原始彩色荷花图 (左) 与仿真得到的 SAR 原始数据幅度 (中) 与相位 (右).

7.5.4.2 距离多普勒成像

7.5.4.2.1 距离多普勒方法

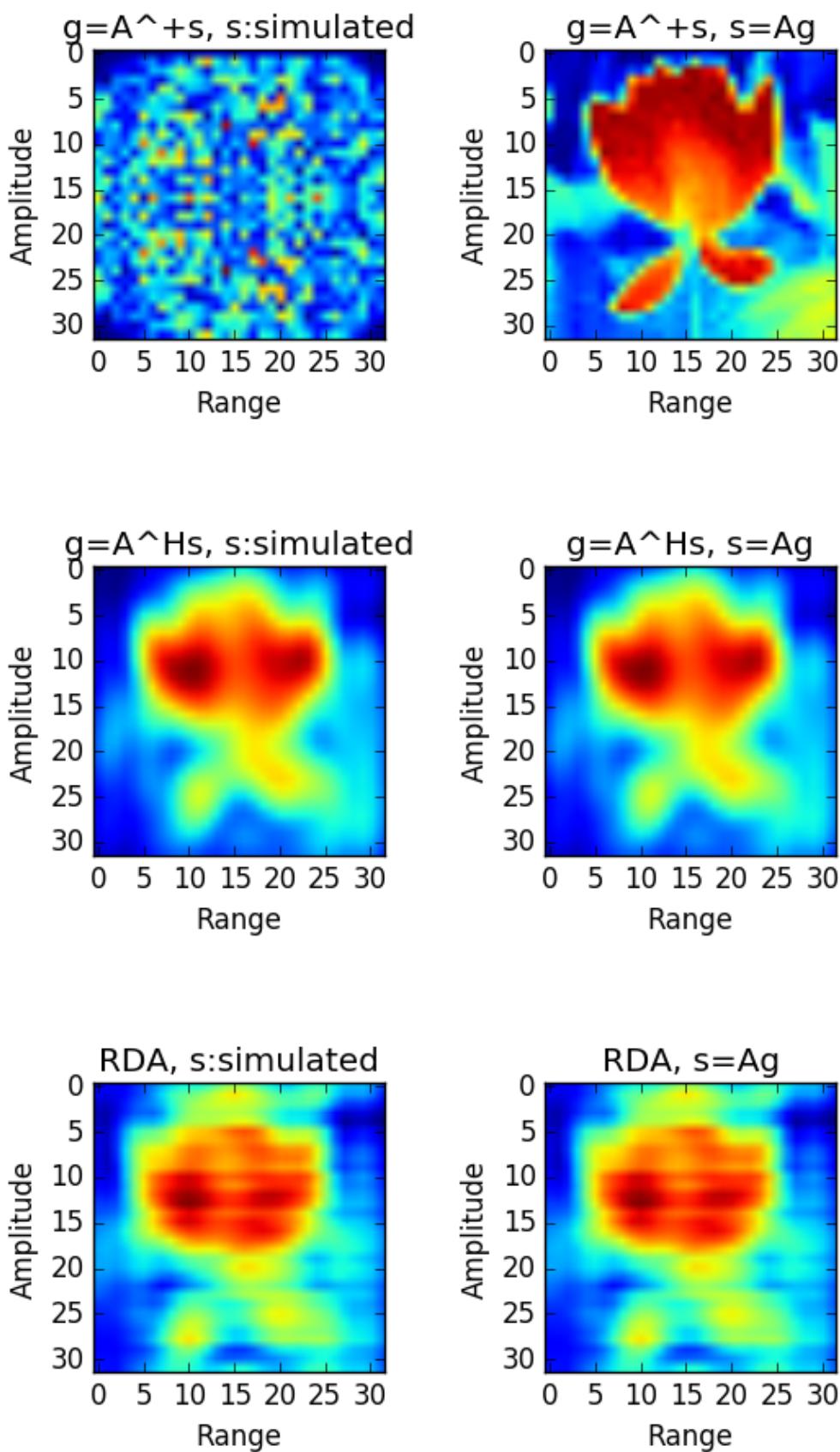
提示: 对于大斜视 SAR, 距离压缩中包含二次距离压缩 (Second Range Compression, SRC), 距离徙动校正也不一样.

7.5.4.2.1.1 小斜视下的 RDA

7.5.4.2.1.2 SAR 原始数据

SAR 原始数据是指 SAR 系统接收到的数据, 数据先被解调至基带, 距离频率中心被置零, 解调后的基带信号为 (参见: [SAR 回波信号及其性质](#) (页 450) 小结, 式.7.7)

$$\begin{aligned} s_i(\eta, \tau) &= G_i h(\tau, \eta) \\ &= G_i w_r (\tau - 2R(\eta)/c) w_a(\eta - \eta_c) \\ &\exp \left\{ -j4\pi f_0 R(\eta)/c + j\pi K_r (\tau - 2R(\eta)/c)^2 \right\}. \end{aligned}$$



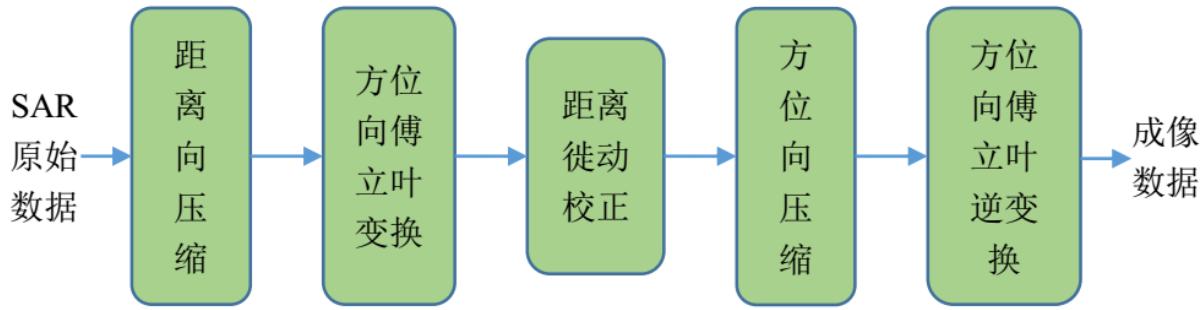


图 7.51: Framework of Range Doppler Algorithm

7.5.4.2.1.3 距离压缩

- 将 SAR 基带信号的距离维 FFT 变换 $S_r(\eta, f_\tau)$ 与距离向匹配滤波器 $H_r(f_\tau)$ 相乘;
- 进行距离维逆 FFT 完成距离维压缩.

匹配滤波器的生成与实现方式不同, 距离压缩的方式也不同. 如采用复制脉冲尾部补零经 FFT 后的复共轭作为滤波器.

SAR 回波信号 $s(\eta, \tau)$ 的距离维傅里叶变换为

$$\begin{aligned} S_r(\eta, f_\tau) &= \text{FFT}_\tau \{s(\eta, \tau)\} \\ &= GW_r(f_\tau)w_a(\eta - \eta_c) \\ &\exp \left\{ -j \frac{4\pi(f_0 + f_\tau)R(\eta)}{c} \right\} \exp \left\{ -j \frac{\pi f_\tau^2}{K_r} \right\}, \end{aligned}$$

其中, G 为常数, $W_r(f_\tau) = w_r(f_\tau/K_r)$ 是距离频谱的包络.

匹配滤波的目的在于消除上式中的第二个指数项, 故取距离向匹配滤波器为

$$H_r(f_\tau) = \exp \left\{ j \frac{\pi f_\tau^2}{K_r} \right\}, \quad (7.14)$$

对滤波器输出进行距离向 IFFT, 得到距离向压缩输出为

$$\begin{aligned} s_{rc}(\eta, \tau) &= \text{IFFT}_\tau \{S_r(\eta, f_\tau)H_r(f_\tau)\} \\ &= Gp_r \left(\tau - \frac{2R(\eta)}{c} \right) w_a(\eta - \eta_c) \\ &\exp \left(-j \frac{4\pi f_0 R(\eta)}{c} \right), \end{aligned} \quad (7.15)$$

其中, 压缩脉冲包络 $p_r(\tau)$ 为窗函数 $W_r(f_\tau)$ 的傅里叶逆变换: 对于矩形窗, $p_r(\tau)$ 为 sinc 函数, 对于锐化窗 (kasier), $p_r(\tau)$ 为旁瓣较低的 sinc 函数. G 为包括散射系数在内的总增益, 常假定为 1.

7.5.4.2.1.4 方位向傅里叶变换

小斜视下, 波束指向接近零多普勒方向, 且 $R_0 \gg V_r \eta$, 可将距离近似为抛物线

$$R(\eta) = \sqrt{R_0^2 + V_r^2 \eta^2} \approx R_0 + \frac{V_r^2 \eta^2}{2R_0},$$

代入距离向压缩输出表达式 7.15 得

$$\begin{aligned} s_{rc}(\eta, \tau) &\approx G p_r \left(\tau - \frac{2R(\eta)}{c} \right) w_a(\eta - \eta_c) \\ &\exp \left(-j \frac{4\pi f_0 R_0}{c} \right) \exp \left(-j \frac{\pi 2V_r^2 \eta^2}{\lambda R_0} \right). \end{aligned}$$

方位向的时频关系为 $f_\eta = -K_a \eta$, 其中记方位向调频率 $K_a \approx \frac{2V_r^2}{\lambda R_0}$, 代入上式, 方位向 FFT 后的信号变为

$$\begin{aligned} S_a(f_\eta, \tau) &= \text{FFT}_\eta \{ s_{rc}(\eta, \tau) \} \\ &= G p_r \left(\tau - \frac{2R_{rd}(f_\eta)}{c} \right) W_a(f_\eta - f_{\eta_c}) \\ &\exp \left(-j \frac{4\pi f_0 R_0}{c} \right) \exp \left(j \frac{\pi f_\eta^2}{K_a} \right), \end{aligned}$$

其中, $W_a(f_\eta - f_{\eta_c})$ 为方位向天线方向图 $w_a(\eta - \eta_c)$ 的频域形式. 第一个指数项含有目标固有的相位信息 (干涉极化) 与图像强度无关. 第二项指数项为具有线性调频特性的频域方位调制. 距离徙动补偿项为

$$R_{rd}(f_\eta) \approx R_0 + \frac{V_r^2}{2R_0} \left(\frac{f_\eta}{K_a} \right)^2 = R_0 + \frac{\lambda^2 R_0 f_\eta^2}{8V_r^2}.$$

7.5.4.2.1.5 距离徙动校正

距离徙动校正 (RCMC) 有两种实现方式.

一种是在距离多普勒域进行距离插值, 可以基于 sinc 函数进行插值处理. 需要校正的 RCM 为方位频率 f_η 的函数, 也是 R_0 的函数:

$$\Delta R(f_\eta) = \frac{\lambda^2 R_0 f_\eta^2}{8V_r^2}.$$

距离徙动校正因子:

$$D(f_\eta, V_r) = \sqrt{1 - \frac{c^2 f_\eta^2}{4V_r^2 f_0^2}} = \sqrt{1 - \frac{\lambda^2 f_\eta^2}{4V_r^2}} \quad (7.16)$$

另一种是基于 RCM 在有限区域内不随距离改变, 从而可以通过 FFT --> 线性相位相乘 --> IFFT 实现, 相位乘法器为

$$Q_{rcmc}(f_\tau) = \exp \left(j \frac{4\pi f_\tau \Delta R(f_\eta)}{c} \right),$$

但这种需要对数据进行分块, 复杂度高, 一般不采用.

基于 sinc 插值进行 RCMC 后的信号变为

$$\begin{aligned} S_{rcmc}(f_\eta, \tau) &= \text{FFT}_\eta \{ s_{rc}(\eta, \tau) \} \\ &= G p_r \left(\tau - \frac{2R_0}{c} \right) W_a(f_\eta - f_{\eta_c}) \\ &\exp \left(-j \frac{4\pi f_0 R_0}{c} \right) \exp \left(j \frac{\pi f_\eta^2}{K_a} \right). \end{aligned}$$

有关实验请参考 [距离徙动校正 \(页 521\)](#) 小节.

7.5.4.2.1.6 方位压缩

- 将方位向 FFT 后的 $S_a(f_\eta, \tau)$ (或 RCMC 后的 $S_{rcmc}(f_\eta, \tau)$) 与方位向匹配滤波器 $H_a(f_\eta)$ 相乘;
- 进行方位维逆 FFT 完成方位维压缩.

方位向压缩在于消除方位向 FFT(或 RCMC) 后的信号的第二个指数项成分, 因而匹配滤波器为其复共轭:

$$H_a(f_\eta) = \exp \left\{ -j \frac{\pi f_\eta^2}{K_a} \right\}.$$

实现方式如下

匹配滤波后的输出变为

$$\begin{aligned} S_{af}(f_\eta, \tau) &= S_{rcmc}(f_\eta, \tau) H_a(f_\eta) \\ &= G p_r \left(\tau - \frac{2R_0}{c} \right) W_a(f_\eta - f_{\eta_c}) \\ &\quad \exp \left(-j \frac{4\pi f_0 R_0}{c} \right) \end{aligned} \tag{7.17}$$

经过方位向 IFFT 后的压缩信号为

$$\begin{aligned} s_{ac}(\eta, \tau) &= \text{IFFT}_\eta \{ S_{af}(f_\eta, \tau) \} \\ &= G p_r \left(\tau - \frac{2R_0}{c} \right) p_a(\eta) \\ &\quad \exp \left(-j \frac{4\pi f_0 R_0}{c} \right) \exp(j2\pi f_{\eta_c} \eta), \end{aligned}$$

其中, p_a 为方位向冲击响应的幅度, 与 p_r 一样为 sinc 函数. 至此, 目标已经被校正到 $\eta = 0, \tau = 2R_0/c$ 处.

提示: 在非零斜视角下, 使用抛物线近似距离使得相位精度不能被保证, 可以使用双曲相位形式的匹配滤波器, 即不进行近似.

7.5.4.2.1.7 多视处理

多视处理可以减少相干斑点噪声, 其主要思路是在方位向做平均滤波, 当然也可以在距离向上做滤波.

7.5.4.2.1.8 大斜视下的 RDA

在低斜视时, 距离等式近似为时间的抛物线方程式, 抛物线模型相当于时域中的线性调频信号, 变换到频域后的信号也具有线性调频形式 [1]. 在大斜视角时, 距离等式应该采用更为精确的双曲线模型, 此时时频间呈非线性关系, 用于距离徙动补偿和方位匹配滤波器的距离应采用新的距离方程, 大斜视引入较强的距离和方位耦合(散焦), [1] 可以通过二次距离压缩来校正.

提示: 下述中, 将徙动因子 $D(f_\eta, V_r)$ 展开并忽略 f_η 二阶及以上项, 则退化为低斜视角下的表达式.

7.5.4.2.1.9 二次距离压缩

在含有交叉耦合的信号的距离多普勒域信号中的调频率由 K_r 变为了 K_m [1] p170. 其中,

$$K_m = \frac{K_r}{1 - K_r/K_{src}}$$

二次压缩滤波器中的调频率为

$$K_{src}(R_0, f_\eta) = \frac{2V_r^2 f_0^3 D^3(f_\eta, V_r)}{c R_0 f_\eta^2}. \quad (7.18)$$

首先使用调频率为 K_r 的滤波器进行匹配滤波实现初级压缩, 再使用调频率为 K_{src} 的滤波器进行次级的匹配滤波实现二次压缩, 因而叫二次距离压缩. 二次距离压缩的滤波器为

$$H_{src}(f_\tau) = \exp \left\{ -j\pi \frac{f_\tau^2}{K_{src}(R_0, f_\eta)} \right\} \quad (7.19)$$

该滤波器与距离 R_0 和方位向频率 f_η 有关. 在距离频域中, SRC 滤波器可以并入距离压缩滤波器中, 合并的滤波器为

$$\begin{aligned} H_m(f_\tau) &= \exp \left\{ j\pi \frac{f_\tau^2}{K_r} \right\} \exp \left\{ -j\pi \frac{f_\tau^2}{K_{src}(R_0, f_\eta)} \right\} \\ &= \exp \left\{ j\pi f_\tau^2 \left(\frac{1}{K_r} - \frac{1}{K_{src}(R_0, f_\eta)} \right) \right\} \\ &= \exp \left\{ j\pi \frac{f_\tau^2}{K_m(R_0, f_\eta)} \right\} \end{aligned}$$

提示: 二次距离压缩的实现方式

二次距离压缩的实现方式有三种:

1. 方式一: 在距离多普勒域中, 随 RCMC 插值一同进行
2. 方式二: 通过二维频域中的相位相乘实现
3. 方式三: 在距离频率 - 方位时域中进行

7.5.4.2.1.10 多普勒相位补偿

对于斜视 SAR, 还需要做多普勒相位补偿 (Doppler Phase Compensation, DPC), 补偿滤波器为

$$H(f_\eta) = \exp \{ 1j2f_\eta Y_c/V_s \}$$

其中, f_η 为方位向频率, Y_c 为场景中心在方位向坐标轴上的投影, V_s 为 SAR 平台速度.

7.5.4.2.1.11 距离徙动的改进

RDA 算法的距离徙动补偿在距离多普勒域执行, 徙动因子如式 7.16 所示. 在大斜视时, 在距离多普勒域的距离徙动量调整为

$$\Delta R(f_\eta) = R_{rd}(f_\eta) - R_0 = R_0 \left[\frac{1 - D(f_\eta, V_r)}{D(f_\eta, V_r)} \right].$$

7.5.4.2.1.12 方位匹配滤波器的改进

对式.7.17 所示方位匹配滤波器调整为

$$H_a(f_\eta) = \exp \left\{ j \frac{4\pi R_0 D(f_\eta, V_r) f_0}{c} \right\} \quad (7.20)$$

7.5.4.2.2 实验与分析

7.5.4.2.2.1 仿真数据实验

7.5.4.2.2.2 实验说明

实验参数及代码参见 SectionBasicExperimentsExperimentsSARRadar 小节, 设置斜视角 $\theta_s = 8.5^\circ$.

7.5.4.2.2.3 实验结果

7.5.4.2.2.4 真实数据实验

7.5.4.2.2.5 实验数据

实验所采用数据为 RADARSAT1 卫星上的合成孔径雷达获取的温哥华地区的图像. 具体介绍参见 RADARSAT 产品介绍 (页 546) 小节.

7.5.4.2.2.6 实验说明

分析 RDA 算法中二次距离压缩与距离徙动补偿的影响.

7.5.4.2.2.7 实验代码

Python 实现代码, 参见文件 [demo_RADARSAT1.py](https://github.com/antsfamily/iprs3.0/tree/master/examples/Products/demo_RADARSAT1.py) (https://github.com/antsfamily/iprs3.0/tree/master/examples/Products/demo_RADARSAT1.py)

7.5.4.2.2.8 实验结果

RADARSAT1 获取的史丹利公园的 SAR 数据幅度与相位

RDA 成像结果 (以下结果均含多普勒相位补偿操作)

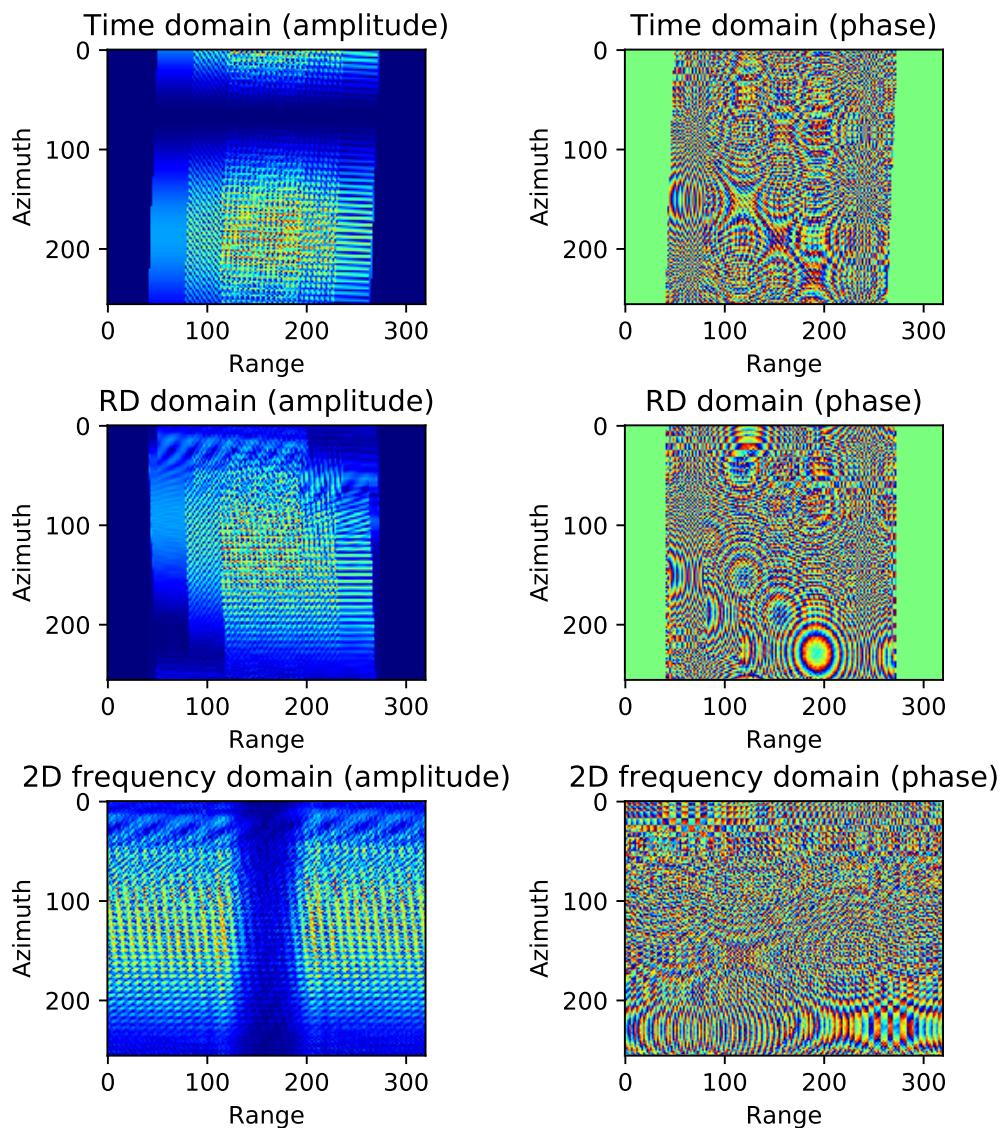


图 7.52: 时域, 距离多普勒域及二维频域幅度相位谱

时域, 距离多普勒域及二维频域幅度相位谱.

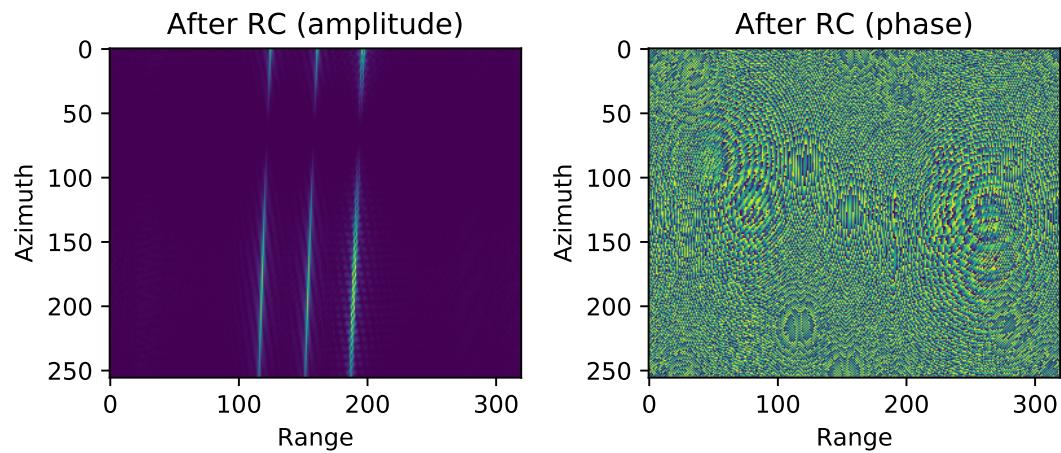


图 7.53: 距离压缩后结果
距离压缩后结果

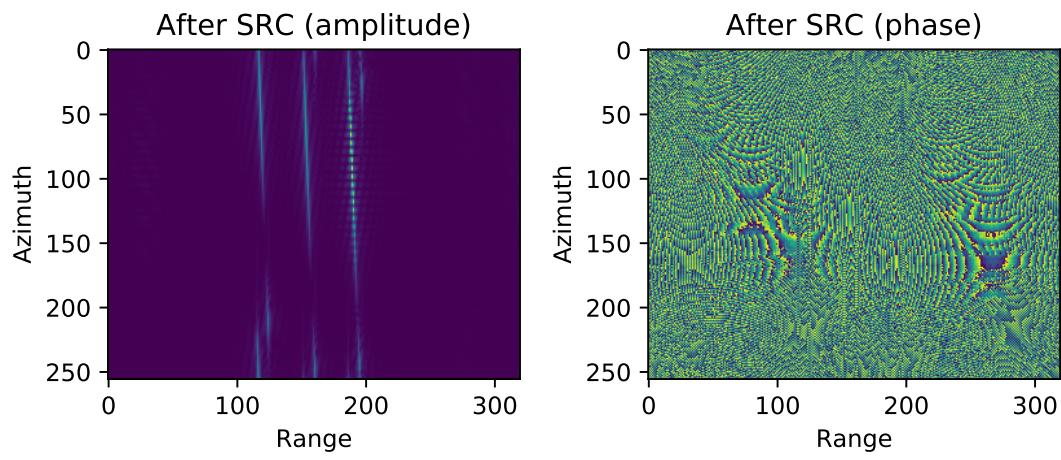


图 7.54: 二次距离压缩后结果
二次距离压缩后结果

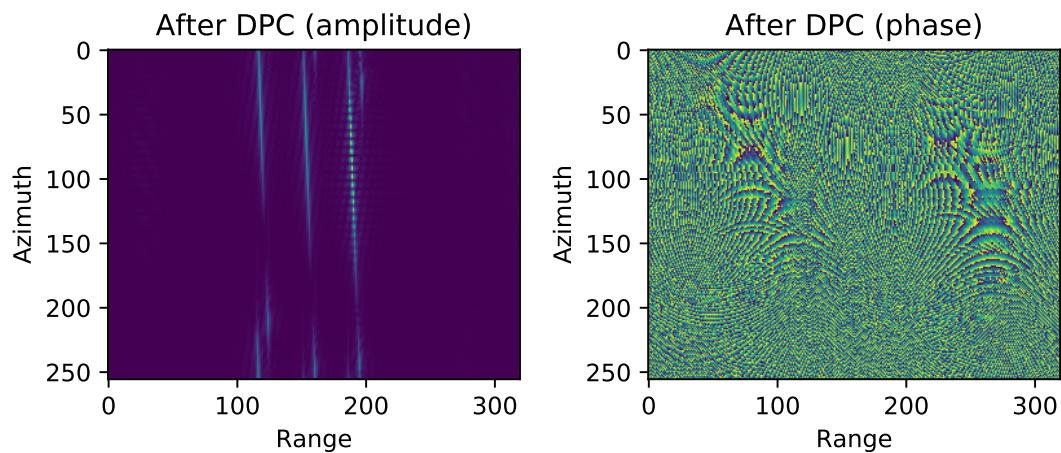


图 7.55: 多普勒相位补偿结果
多普勒相位补偿结果

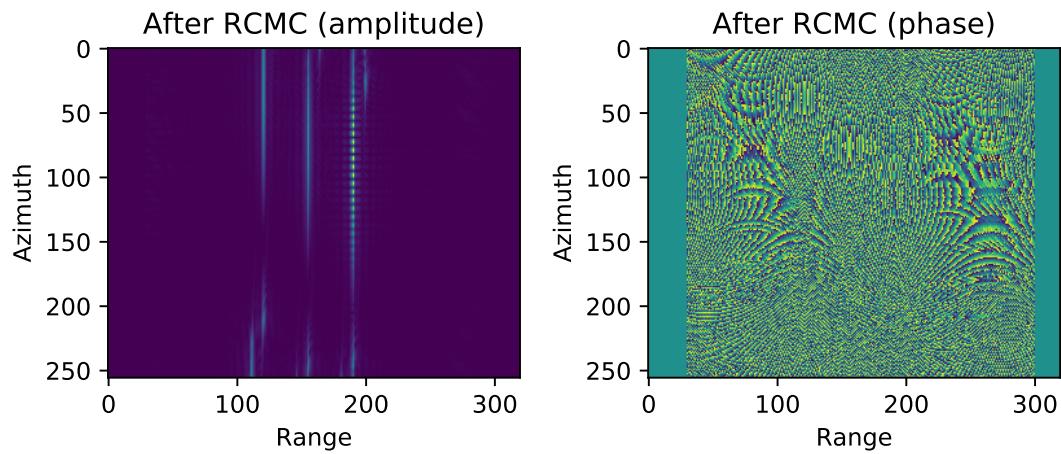


图 7.56: 距离徙动校正结果
距离徙动校正结果

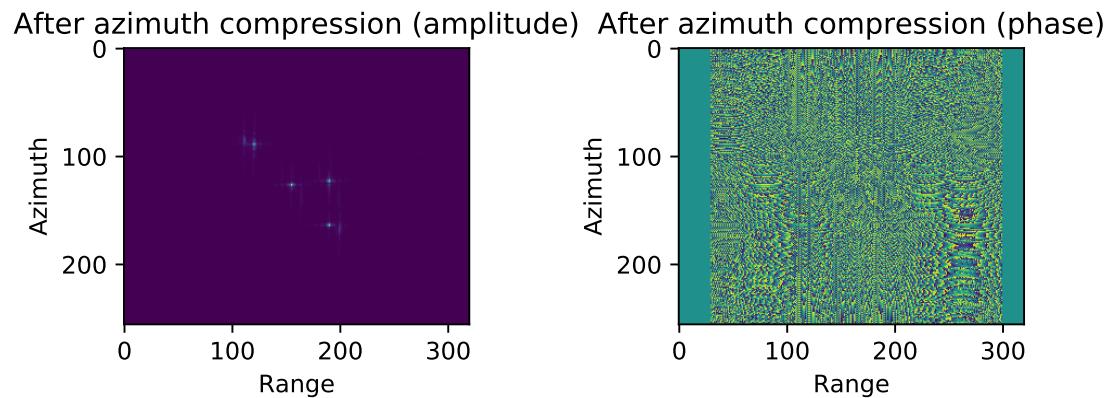


图 7.57: 方位向压缩结果
方位向压缩结果

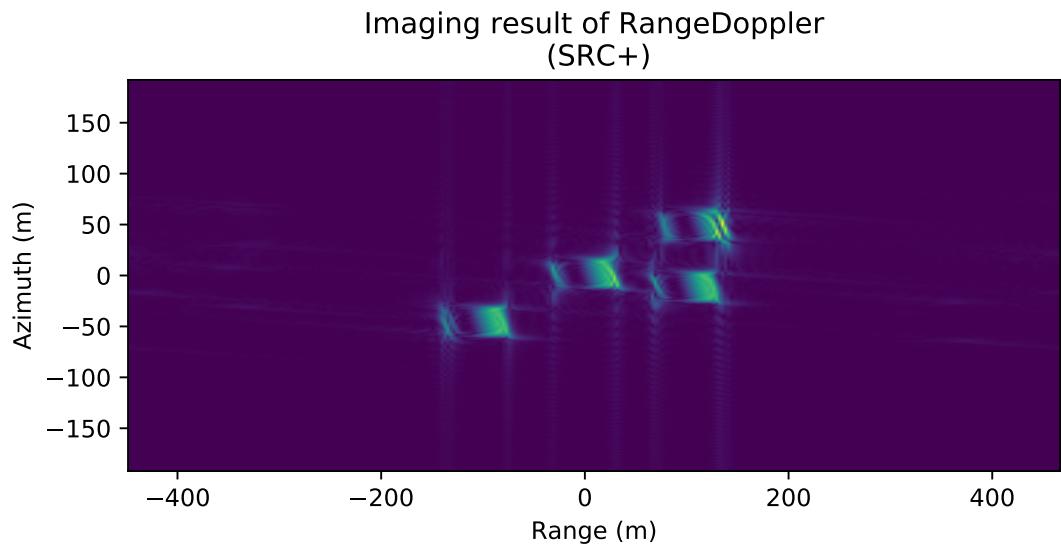


图 7.58: RDA 算法最终成像结果 (不含距离徙动校正)
RDA 算法最终成像结果 (不含距离徙动校正)

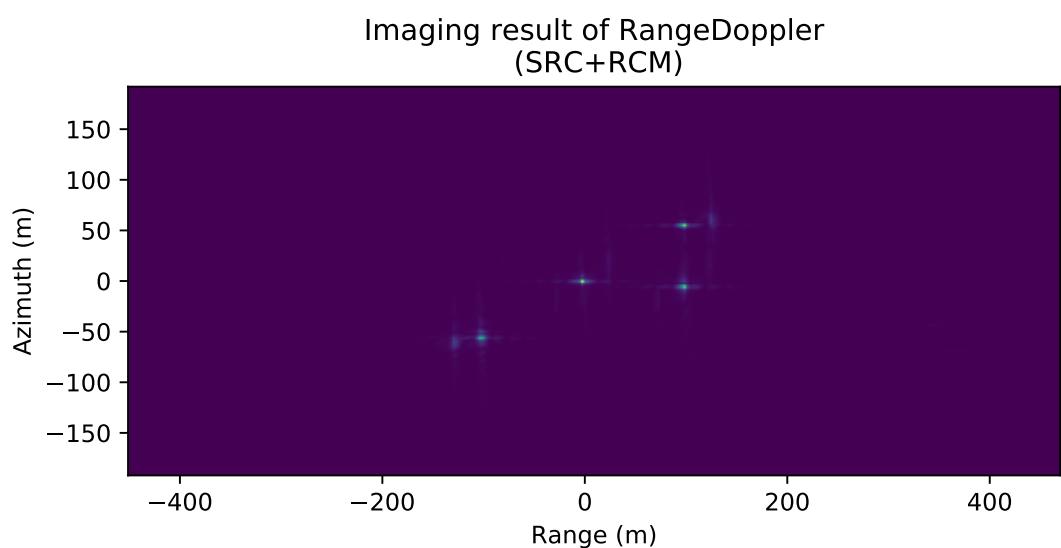


图 7.59: RDA 算法最终成像结果 (含距离徙动校正)
RDA 算法最终成像结果 (含距离徙动校正)

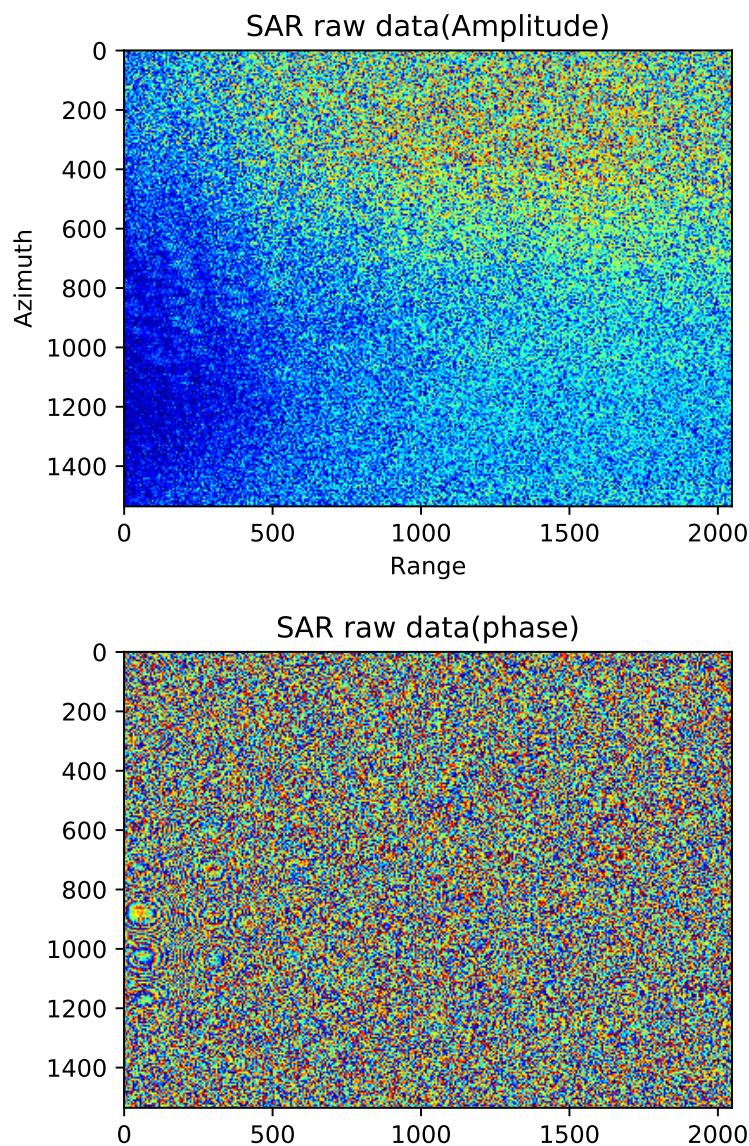


图 7.60: SAR raw data amplitude and phase of Stanley Park.

SAR raw data amplitude and phase of Stanley Park.

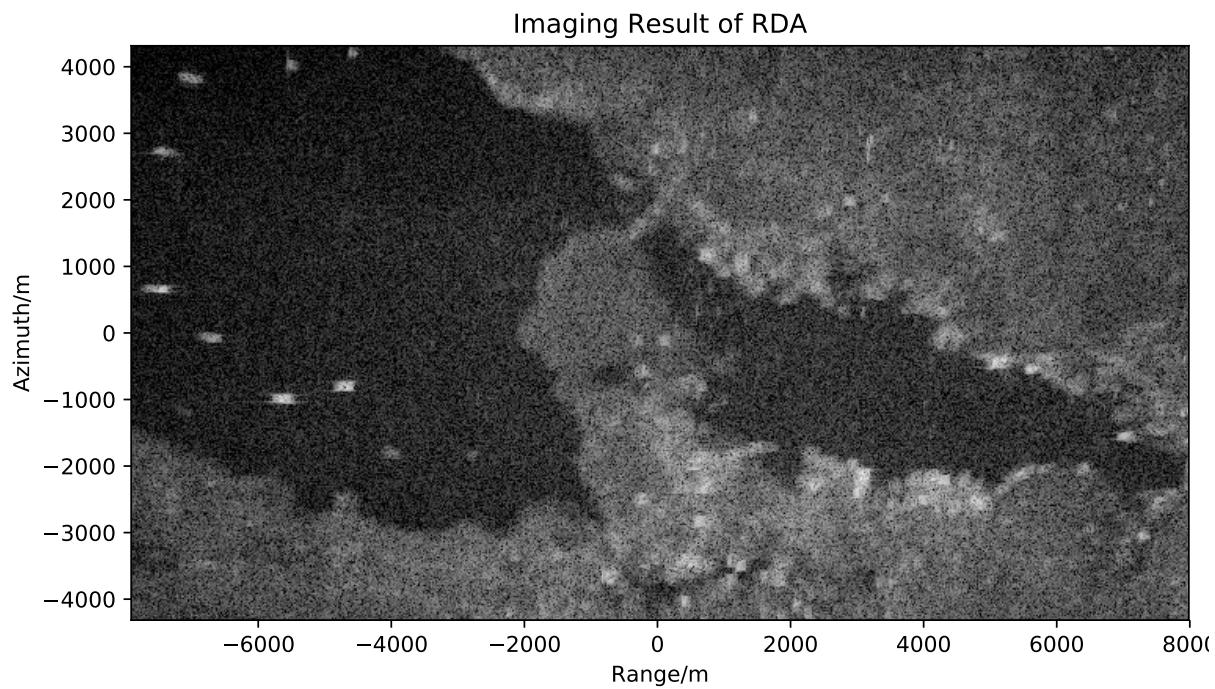


图 7.61: Imaging result of RDA (without SRC, without RCMC)

Imaging result of RDA (without SRC, without RCMC)

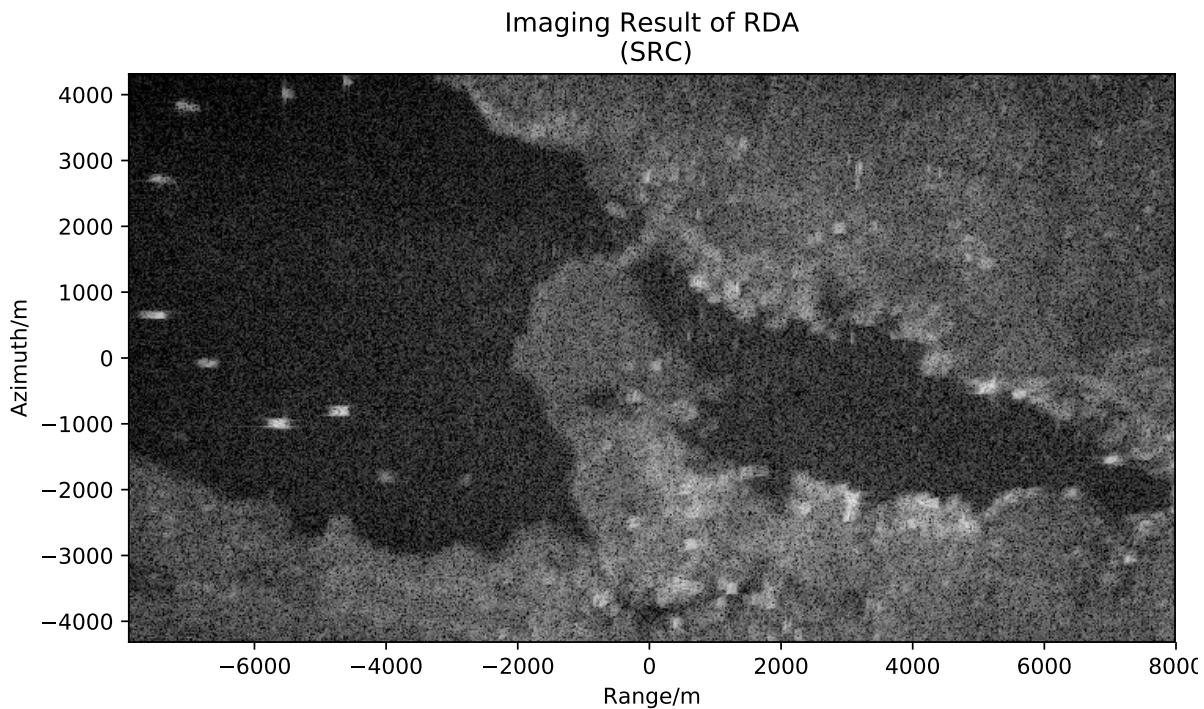


图 7.62: Imaging result of RDA (with SRC, without RCMC)

Imaging result of RDA (with SRC, without RCMC)

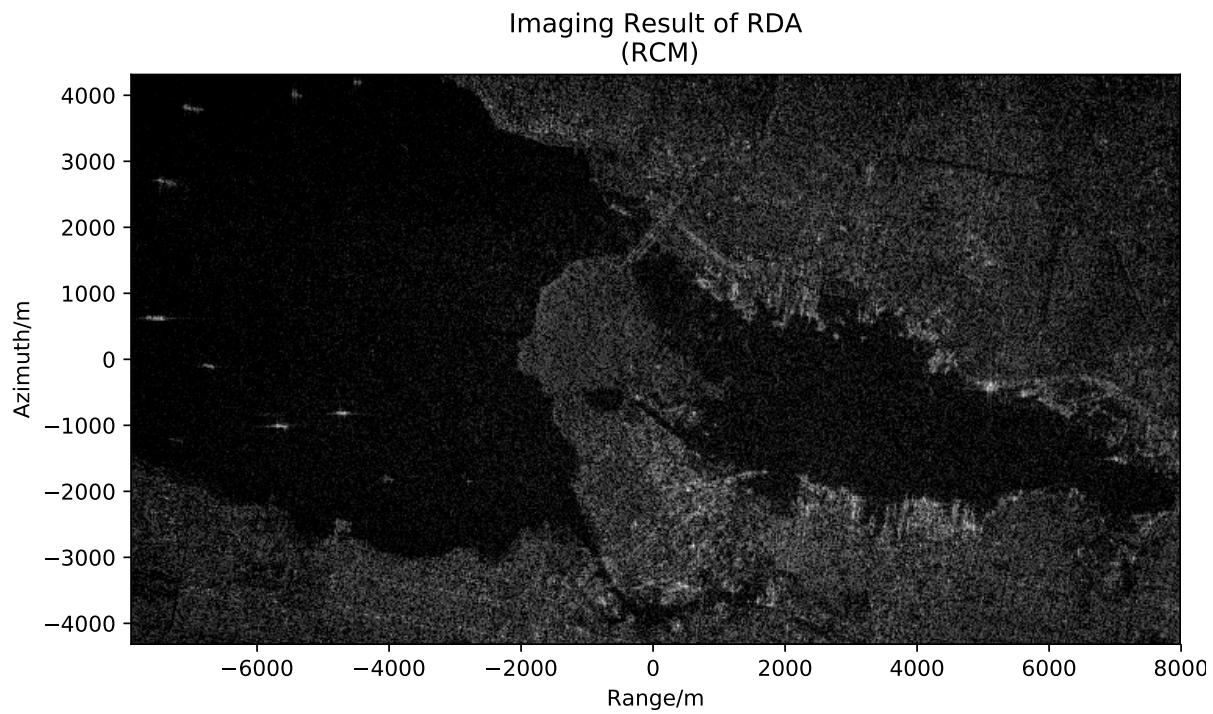


图 7.63: Imaging result of RDA (without SRC, with RCMC(sinc 32))

Imaging result of RDA (without SRC, with RCMC(sinc 32))

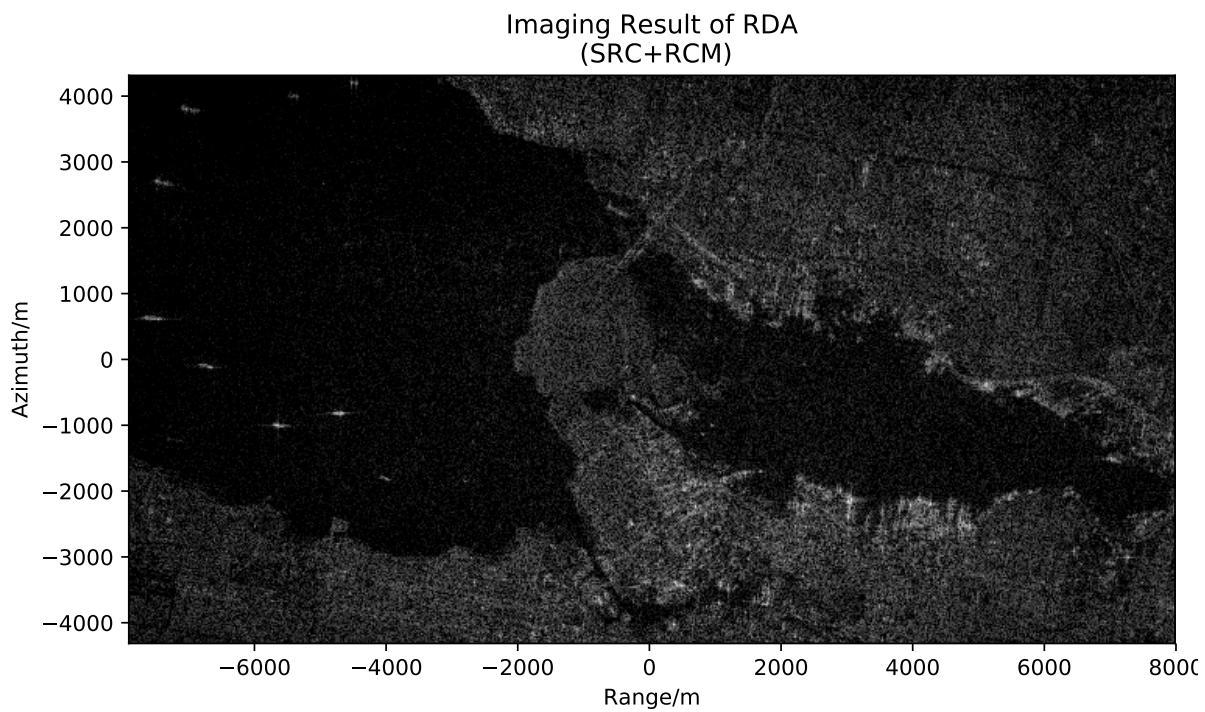


图 7.64: Imaging result of RDA (with SRC, with RCMC(sinc 32))

Imaging result of RDA (with SRC, with RCMC(sinc 32))

7.5.4.3 调频变标成像

7.5.4.3.1 调频变标原理

Chirp Scaling

7.5.4.3.2 调频变标成像

7.5.4.3.2.1 CSA 方法概览

注解: Chirp Scaling 算法

输入: SAR 原始回波数据矩阵 S , SAR 平台参数

输出: SAR 复数图像

Step1. 对时域回波数据 S 作方位 FFT, 变换到距离多普勒域 $\text{FFT}(S)$

Step2. 通过相位相乘实现调频变标操作, 变标方程为式 7.21

Step3. 对 Step2 中结果做 FFT 变换到二维频域

Step4. 完成距离压缩, 二次距离压缩和一致 RCMC

Step5. 做距离向 IFFT 将数据变换回距离多普勒域

Step6. 与随距离变化的匹配滤波器(含相位校正)相乘, 实现方位压缩

Step7. 通过方位向 IFFT 将数据变换回二维时域, 即复数 SAR 图像.

7.5.4.3.2.2 调频变标

线性调频变标方程为

$$s_{sc}(\tau', f_\eta) = \exp \left\{ j\pi K_m \left[\frac{D(f_{\eta_{ref}}, V_{r_{ref}})}{D(f_\eta, V_{r_{ref}})} - 1 \right] (\tau')^2 \right\} \quad (7.21)$$

其中, $\tau' = \tau - \frac{2R_{ref}}{cD(f_\eta, V_{r_{ref}})}$, $K_m = \frac{K_r}{1 - K_r \frac{cR_0 f_\eta^2}{2V_r^2 f_0^3 D^3(f_\eta, V_r)}}$, $D(f_\eta, V_r) = \sqrt{1 - \frac{c^2 f_\eta^2}{4V_r^2 f_0^2}}$, 方位频率 f_η 处的时间平移是距离时间的函数

$$\tau = \frac{2}{c} \left\{ \frac{R_0}{D(f_{\eta_{ref}}, V_r)} + \left[\frac{R_{ref}}{D(f_\eta, V_{r_{ref}})} - \frac{R_{ref}}{D(f_{f_{\eta_{ref}}}, V_{r_{ref}})} \right] \right\}$$

非线性调频变标方程 [1] p209

7.5.4.3.2.3 距离压缩与距离徙动校正

距离匹配滤波器仍为式.7.14 所示的匹配滤波器

$$H_r(f_\tau) = \text{rect}\left(\frac{f_\tau}{|K_r|T_p}\right) \exp\left(j\frac{\pi f_\tau^2}{K_r}\right), \quad (7.22)$$

有关二次距离压缩的内容参见 [距离多普勒成像](#) (页 458) 中的大斜视下的 RDA (页 462).

整体 RCM 为

$$\text{RCM}_{total}(R_0, f_\eta) = \frac{R_0}{D(f_\eta, V_r)} - \frac{R_0}{D(f_{\eta_{ref}}, V_r)} \quad (7.23)$$

一致 RCM 为

$$\text{RCM}_{bulk}(f_\eta) = \frac{R_{ref}}{D(f_\eta, V_{r_{ref}})} - \frac{R_{ref}}{D(f_{\eta_{ref}}, V_{r_{ref}})} \quad (7.24)$$

补余 RCM 为式.7.23 减去式.7.24

7.5.4.3.2.4 方位向处理

[1] p209

方位向包含三部分处理: 方位向匹配滤波, 附加相位校正, 方位向 IFFT.

通过距离向 IFFT 完成距离向的处理, 得到距离多普勒域信号

$$\begin{aligned} S_4(\tau, f_\eta) = & A_2 p_r \left(\tau - \frac{2R_0}{c D(f_{\eta_{ref}}, V_{r_{ref}})} \right) W_a(f_\eta - f_{\eta_c}) \\ & \times \exp \left\{ -j \frac{4\pi R_0 f_0 D(f_\eta, V_r)}{c} \right\} \\ & \times \exp \left\{ j \frac{4\pi K_m}{c^2} \left[1 - \frac{D(f_\eta, V_{r_{ref}})}{D(f_{\eta_{ref}}, V_{r_{ref}})} \right] \times \left[\frac{R_0}{D(f_\eta, V_r)} - \frac{R_{ref}}{D(f_{\eta_{ref}}, V_r)} \right]^2 \right\} \end{aligned} \quad (7.25)$$

- 方位向匹配滤波器为式.7.25 中的 第一个指数项的复共轭
- 对于线性调频变标, 附加相位校正乘法器为式.7.25 中的 第二个指数项的复共轭

7.5.4.3.3 实验与分析

7.5.4.3.3.1 仿真数据实验

7.5.4.3.3.2 实验说明

实验参数及代码参见 [基础分析实验](#) (页 512) 小节, 设置斜视角 $\theta_s = 8.5^\circ$.

7.5.4.3.3.3 实验结果

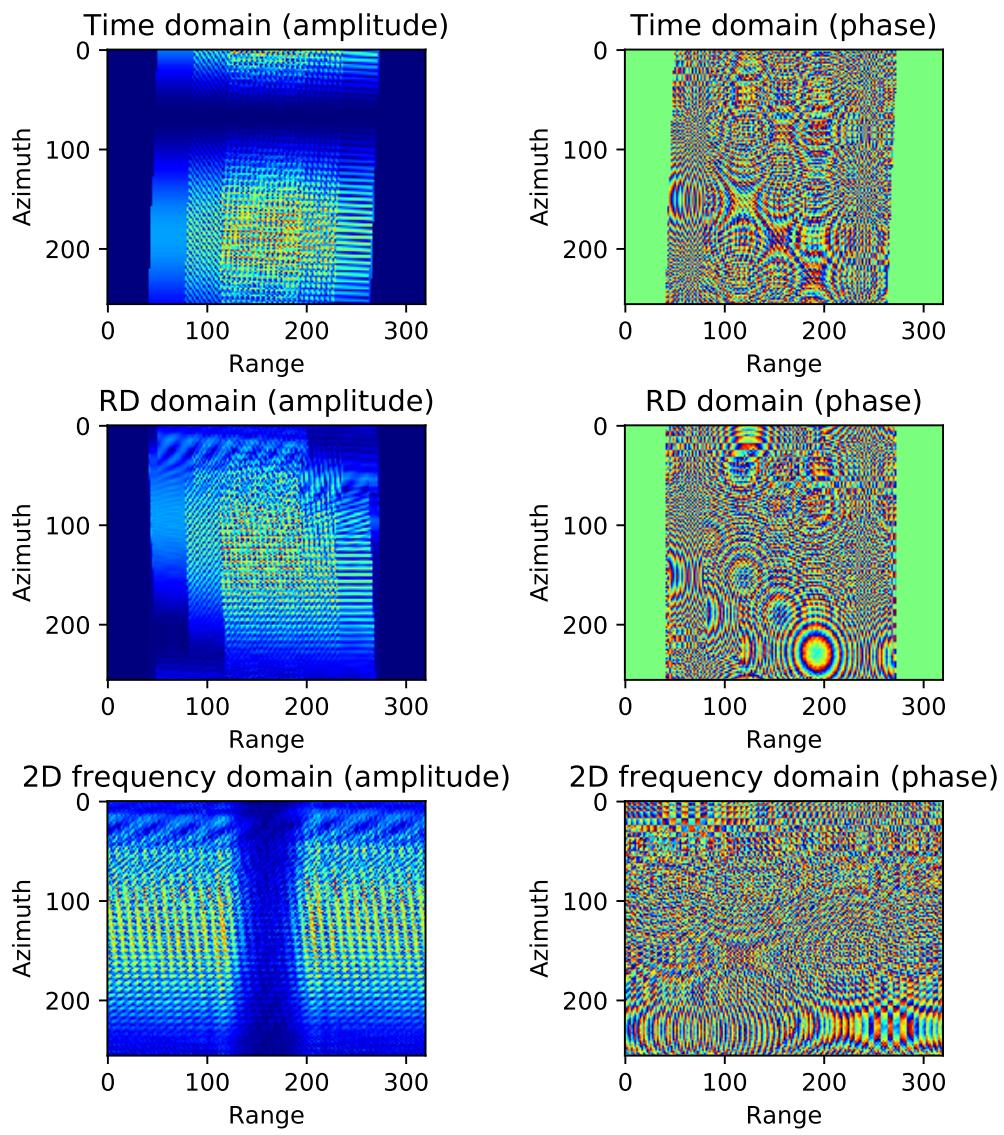


图 7.65: 时域, 距离多普勒域及二维频域幅度相位谱
时域, 距离多普勒域及二维频域幅度相位谱.

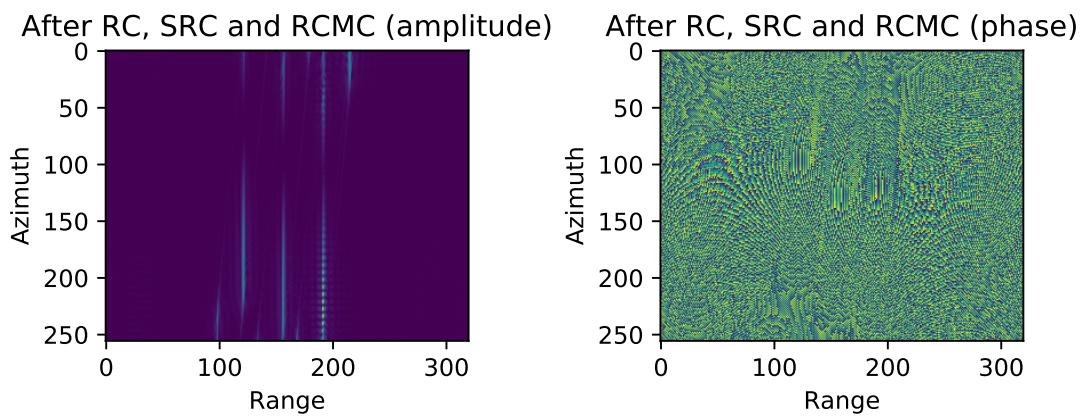


图 7.66: 距离压缩, 二次距离压缩及距离徙动校正结果

距离压缩, 二次距离压缩及距离徙动校正结果

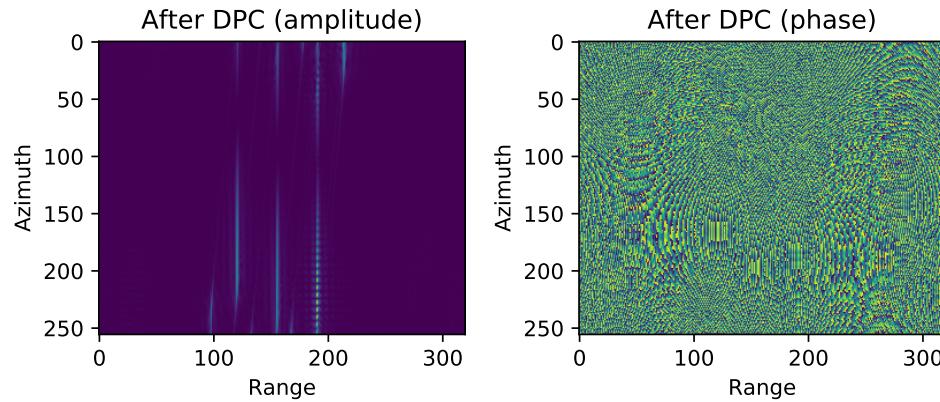


图 7.67: 多普勒相位补偿结果

多普勒相位补偿结果

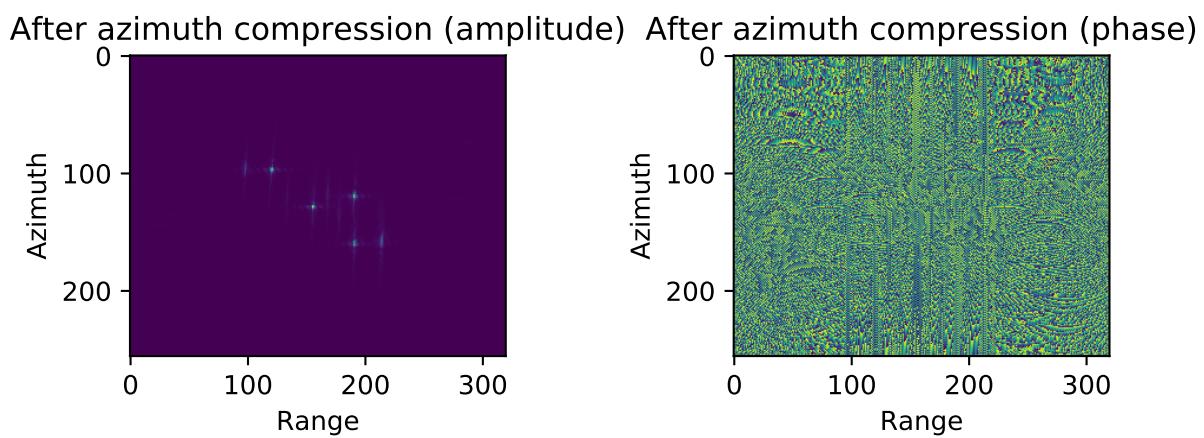


图 7.68: 方位向压缩结果 (含距离徙动校正)

方位向压缩结果 (含距离徙动校正)

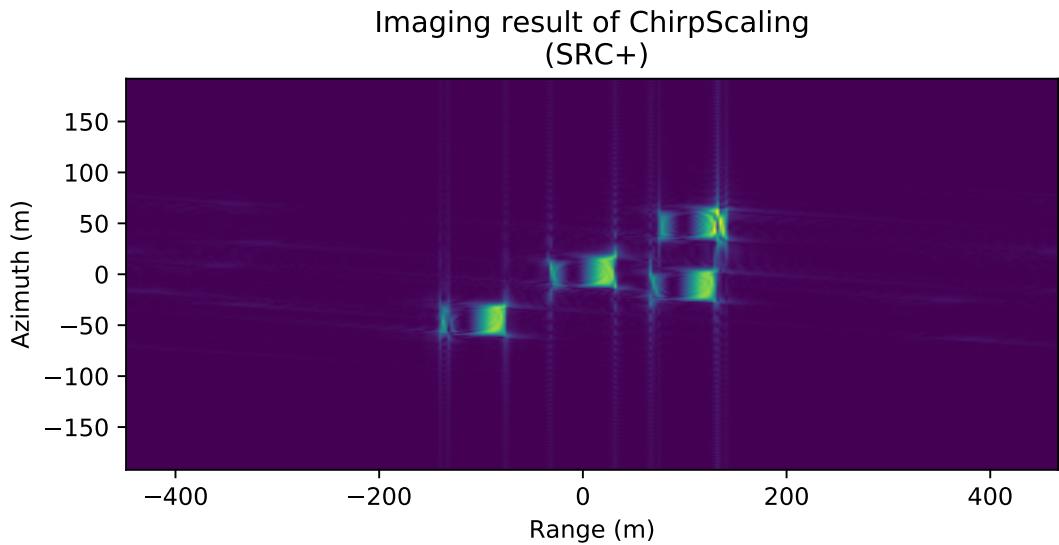


图 7.69: CSA 算法最终成像结果 (不含距离徙动校正, 含多普勒相位补偿)
CSA 算法最终成像结果 (不含距离徙动校正, 含多普勒相位补偿)

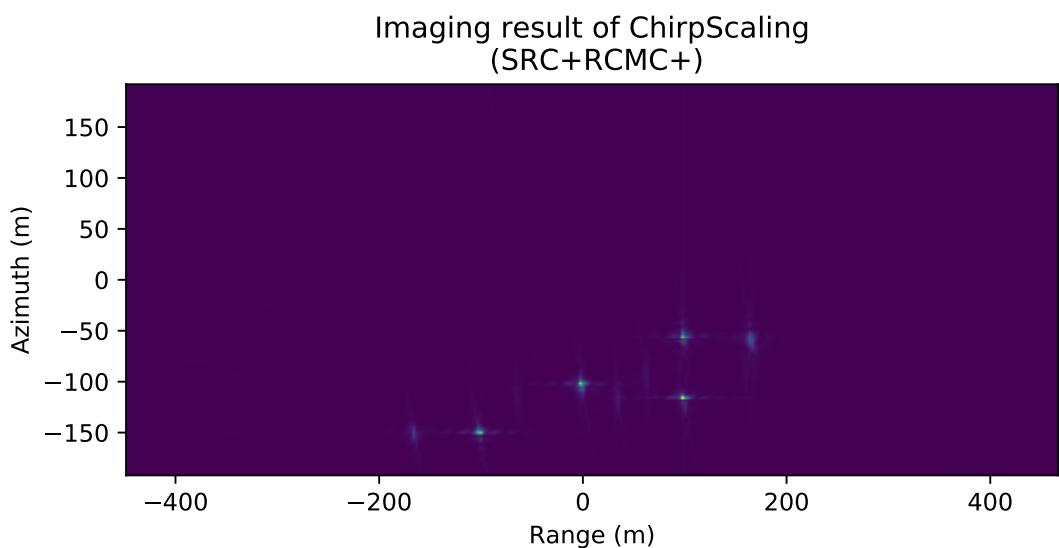


图 7.70: CSA 算法最终成像结果 (含距离徙动校正, 不含多普勒相位补偿)
CSA 算法最终成像结果 (含距离徙动校正, 不含多普勒相位补偿)

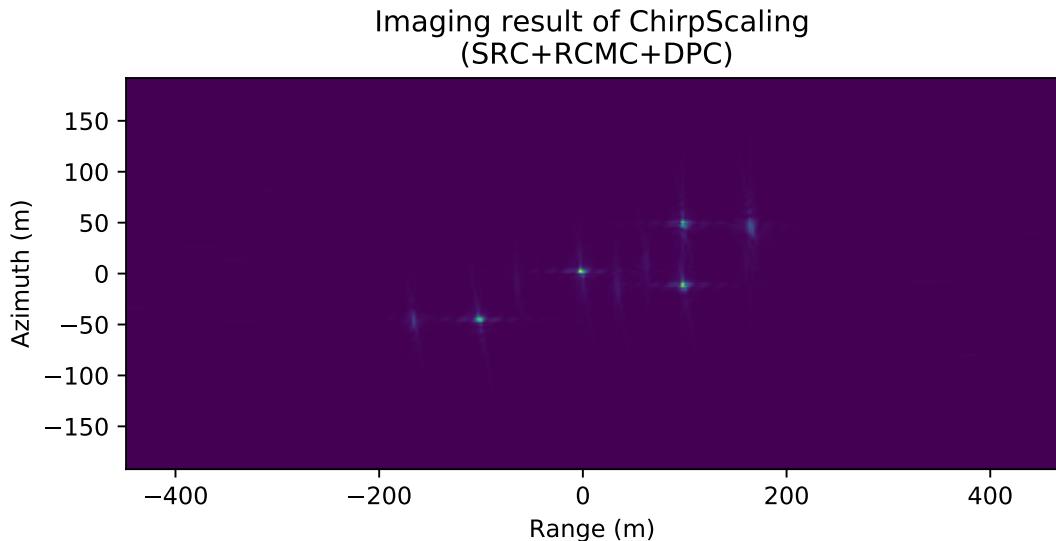


图 7.71: CSA 算法最终成像结果 (含距离徙动校正, 含多普勒相位补偿)

CSA 算法最终成像结果 (含距离徙动校正, 含多普勒相位补偿)

7.5.4.3.3.4 真实数据实验

7.5.4.3.3.5 实验数据

实验所采用数据为 RADARSAT1 卫星上的合成孔径雷达获取的温哥华地区的图像。具体介绍参见 *RADARSAT 产品介绍* (页 546) 小节。

7.5.4.3.3.6 实验说明

分析 RDA 算法中二次距离压缩与距离徙动补偿的影响。

7.5.4.3.3.7 实验代码

Python 实现代码, 参见文件 `demo_RADARSAT1.py` (https://github.com/antsfamily/iprs3.0/tree/master/examples/Products/demo_RADARSAT1.py)。

7.5.4.3.3.8 实验结果

RADARSAT1 获取的史丹利公园的 SAR 数据幅度与相位

CSA 成像结果

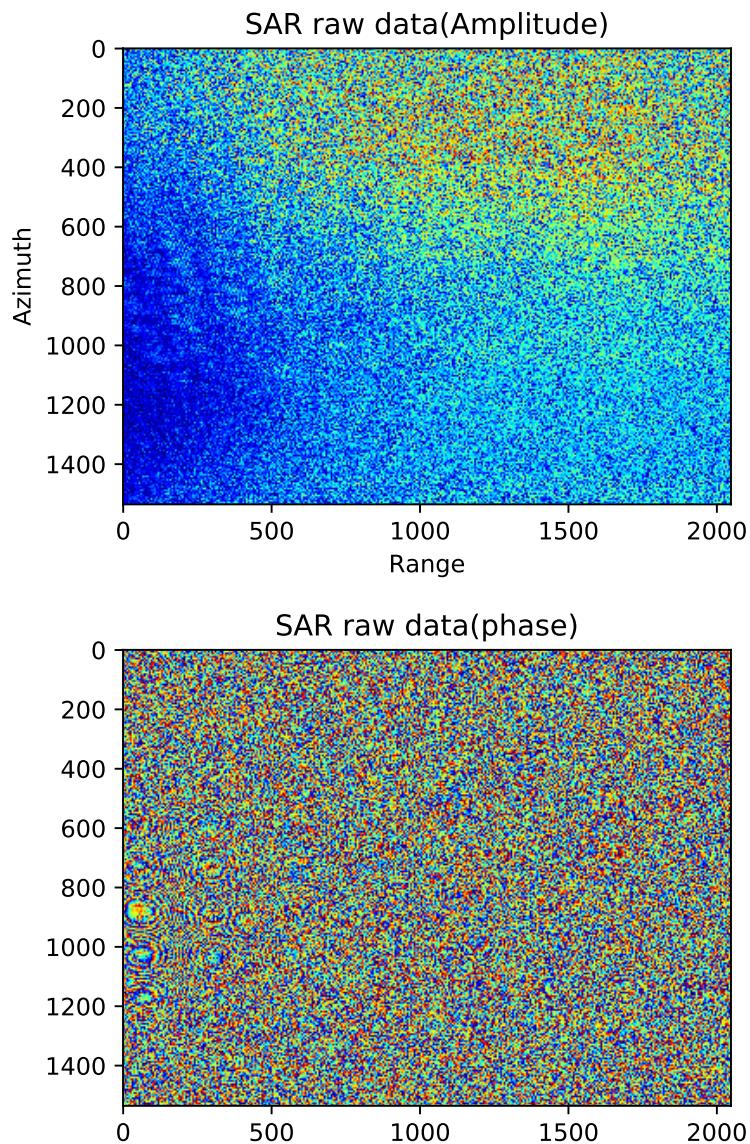


图 7.72: SAR raw data amplitude and phase of Stanley Park.

SAR raw data amplitude and phase of Stanley Park.

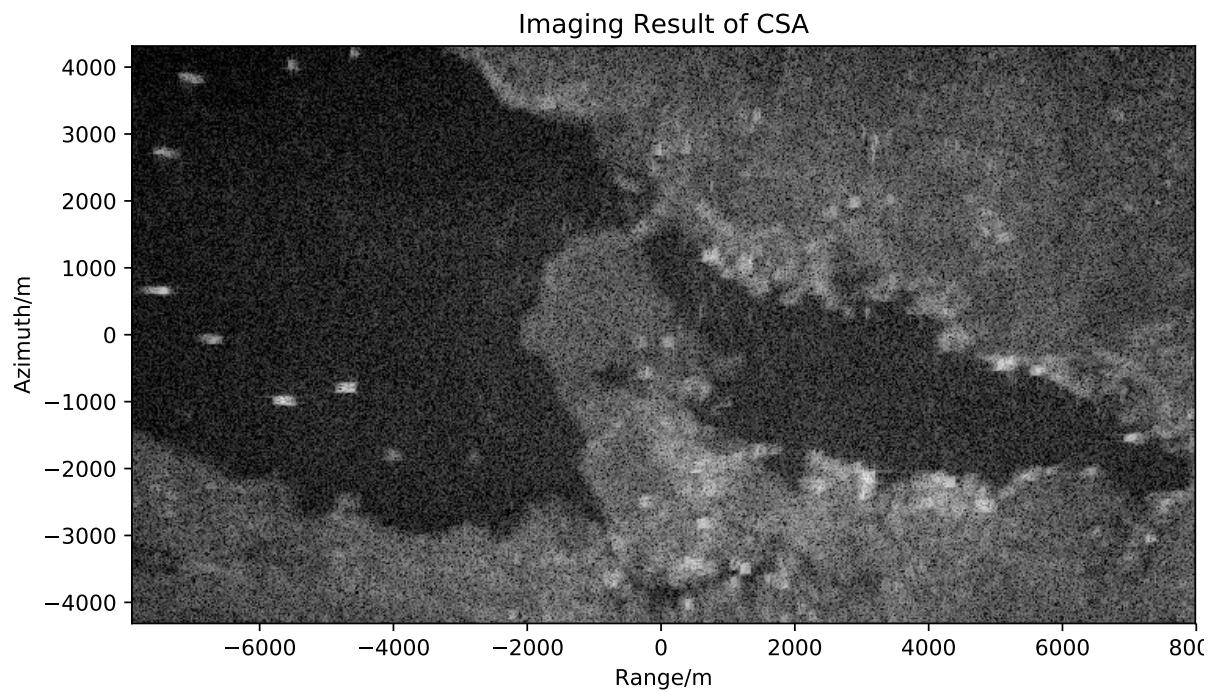


图 7.73: Imaging result of CSA (without SRC, without RCMC)

Imaging result of CSA (without SRC, without RCMC)

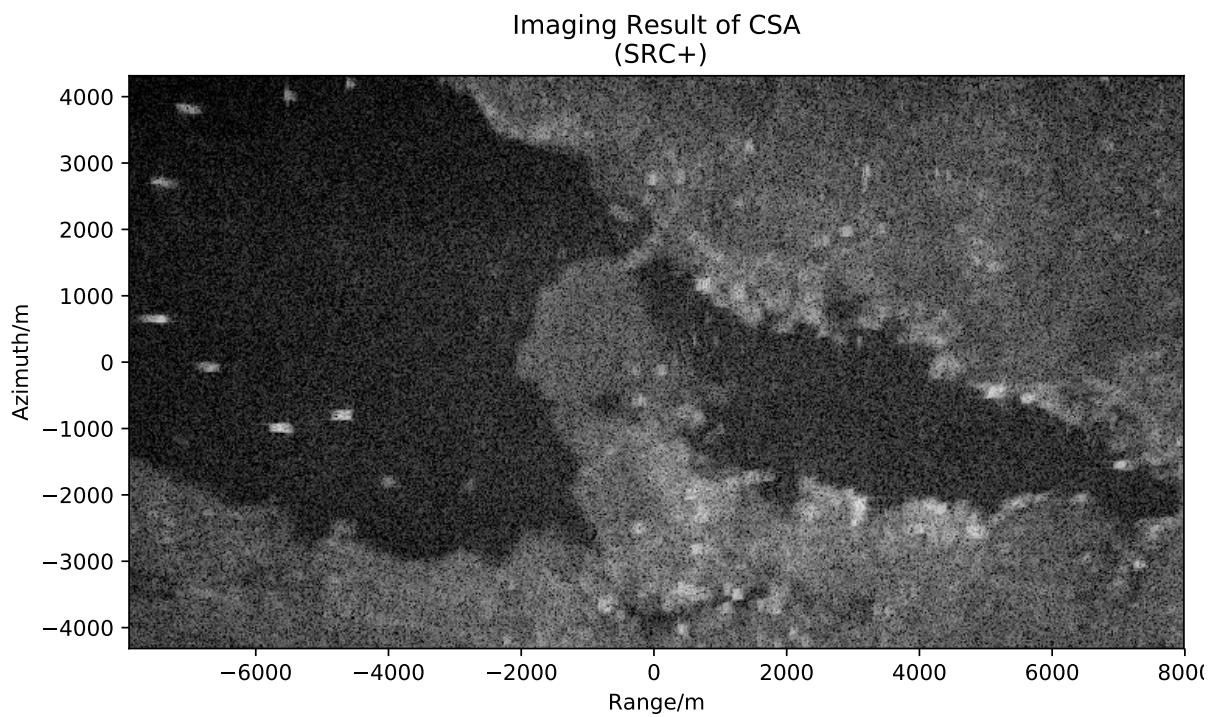


图 7.74: Imaging result of CSA (with SRC, without RCMC)

Imaging result of CSA (with SRC, without RCMC)

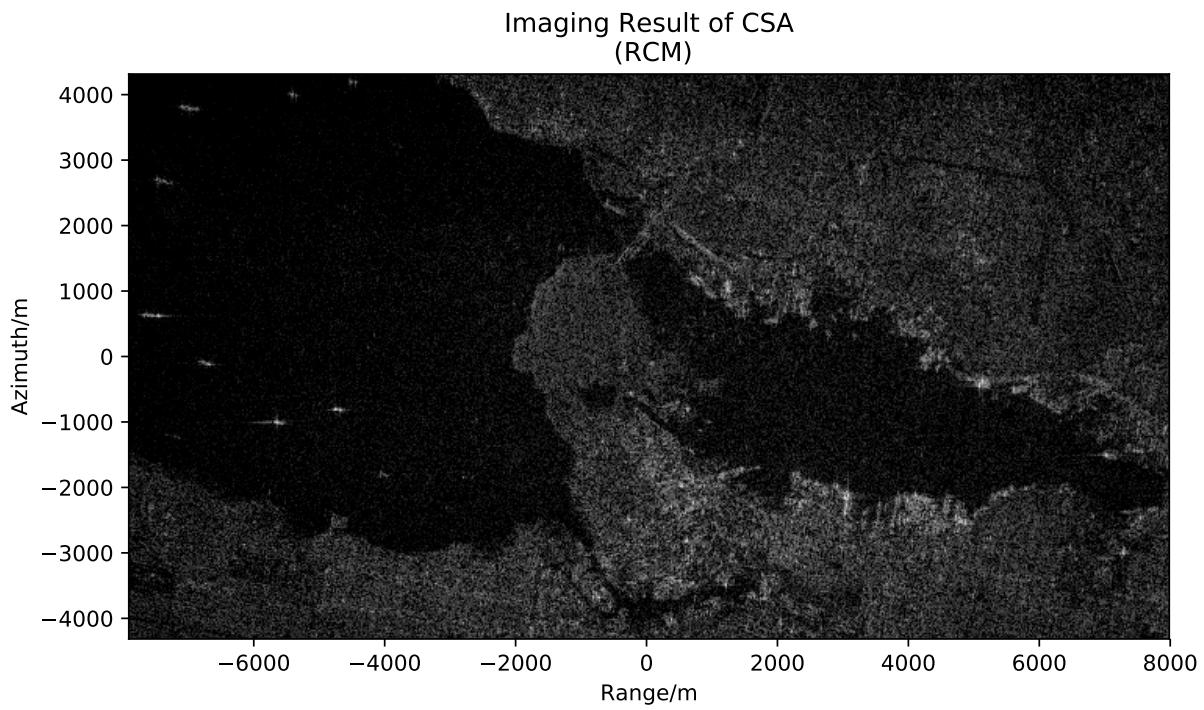


图 7.75: Imaging result of CSA (without SRC, with RCMC)

Imaging result of CSA (without SRC, with RCMC)

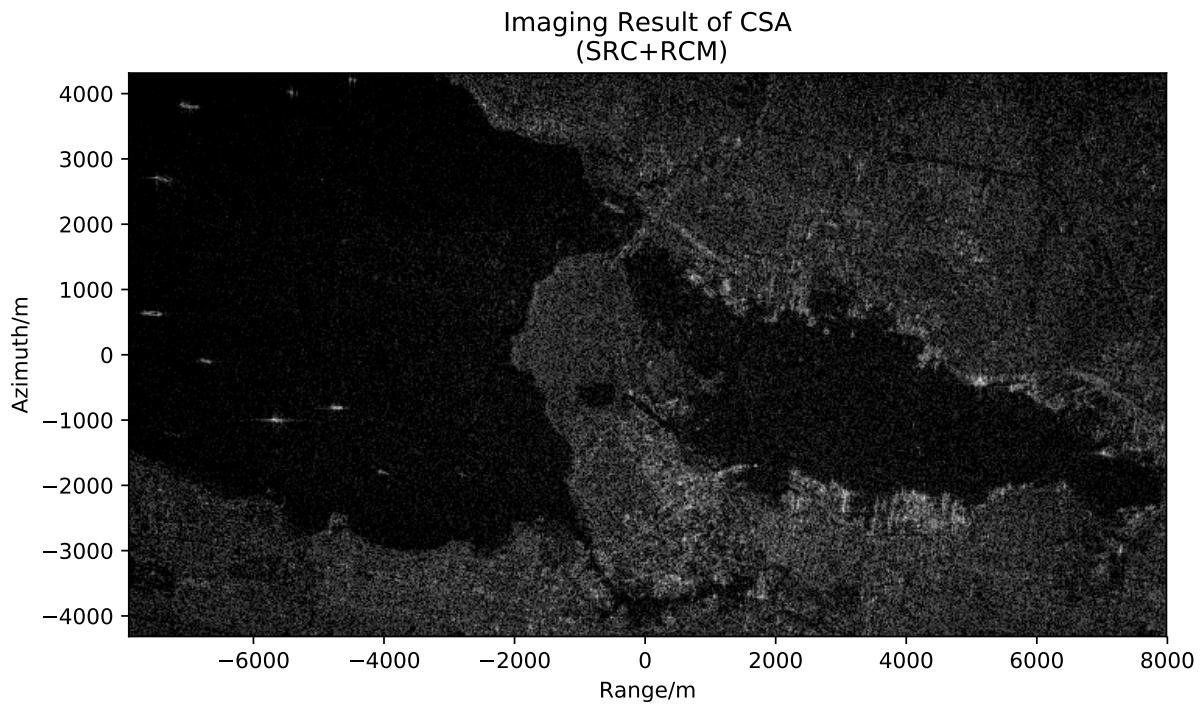


图 7.76: Imaging result of CSA (with SRC, with RCMC)

Imaging result of CSA (with SRC, with RCMC)

7.5.4.4 正则化成像方法

7.5.4.4.1 正则化成像

SAR imaging process can be formulated as

$$\mathbf{s} = \mathbf{A}\mathbf{g},$$

where, \mathbf{s} is the $MN \times 1$ received SAR raw data vector in phase history domain, \mathbf{g} is the $HW \times 1$ reflection vector of scene, and \mathbf{A} represents the mapping from scene to SAR raw data.

Given \mathbf{s} , \mathbf{A} , regularization methods try to reconstruct \mathbf{g} by minimizing

$$\min_{\mathbf{g}} \|\mathbf{A}\mathbf{g} - \mathbf{s}\|_2 + \lambda |\mathbf{g}|_p,$$

where, λ is the balance factor, and $|\cdot|_p$ is the ℓ_p norm.

Note that, if $\mathbf{s}, \mathbf{A}, \mathbf{g} \in \mathbb{C}$, the problem changes to

$$\text{Re}(\mathbf{s}) + j\text{Im}(\mathbf{s}) = \text{Re}(\mathbf{A}\mathbf{g}) + j\text{Im}(\mathbf{A}\mathbf{g})$$

so we have:

$$\begin{bmatrix} \text{Re}(\mathbf{s}) \\ \text{Im}(\mathbf{s}) \end{bmatrix} = \begin{bmatrix} \text{Re}(\mathbf{A}) & -\text{Im}(\mathbf{A}) \\ \text{Im}(\mathbf{A}) & \text{Re}(\mathbf{A}) \end{bmatrix} \begin{bmatrix} \text{Re}(\mathbf{g}) \\ \text{Im}(\mathbf{g}) \end{bmatrix}$$

7.5.4.4.2 实验与分析

7.5.4.4.2.1 实验说明

- 仿真场景大小: 32×32
- 回波矩阵大小: 32×32
- 稀疏表示字典: 无, DCT, DWT
- 优化方法: Lasso, OMP

仿真点目标场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

7.5.4.4.2.2 实验代码

[iprs2.0](https://github.com/antsfamily/iprs2.0) (<https://github.com/antsfamily/iprs2.0>) `demo_regular_sar.py`

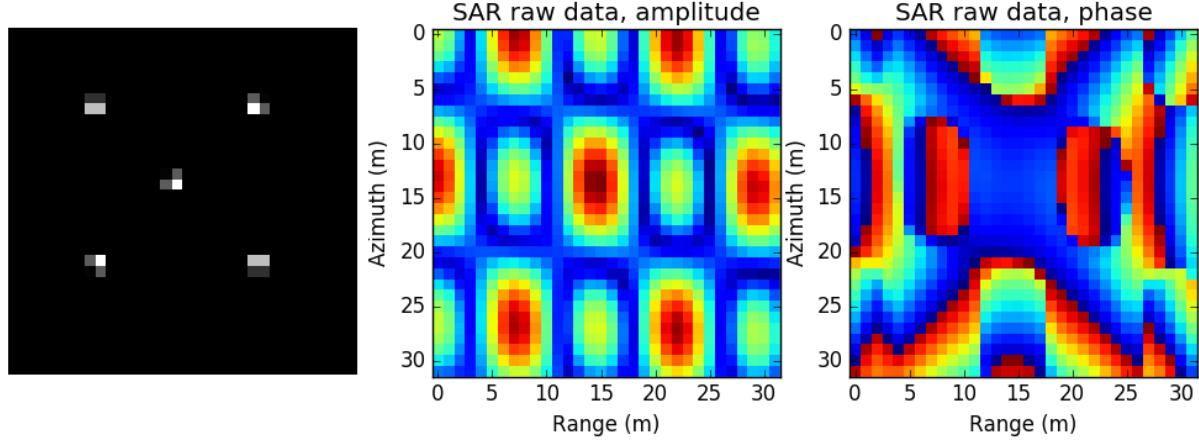


图 7.77: 仿真点目标场景图, 仿真 SAR 原始数据幅度相位图

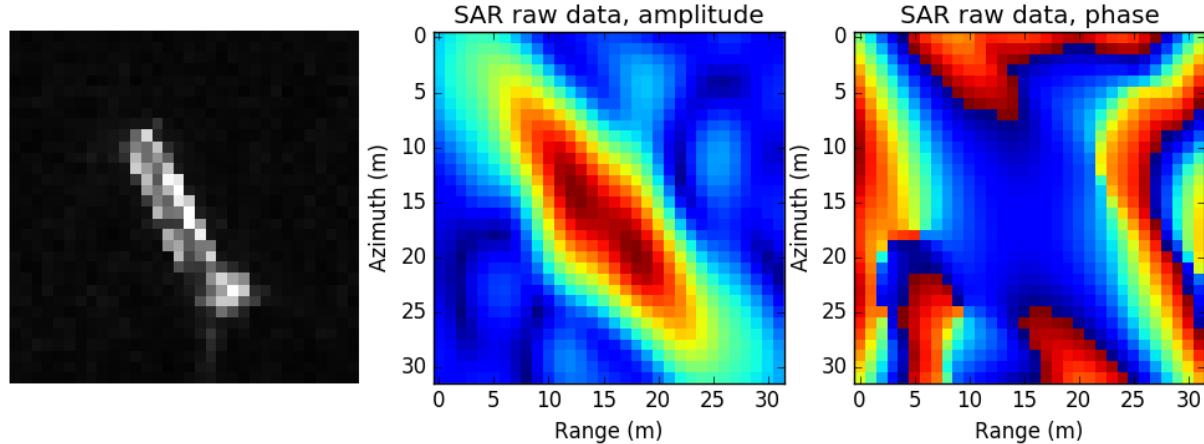


图 7.78: 仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下

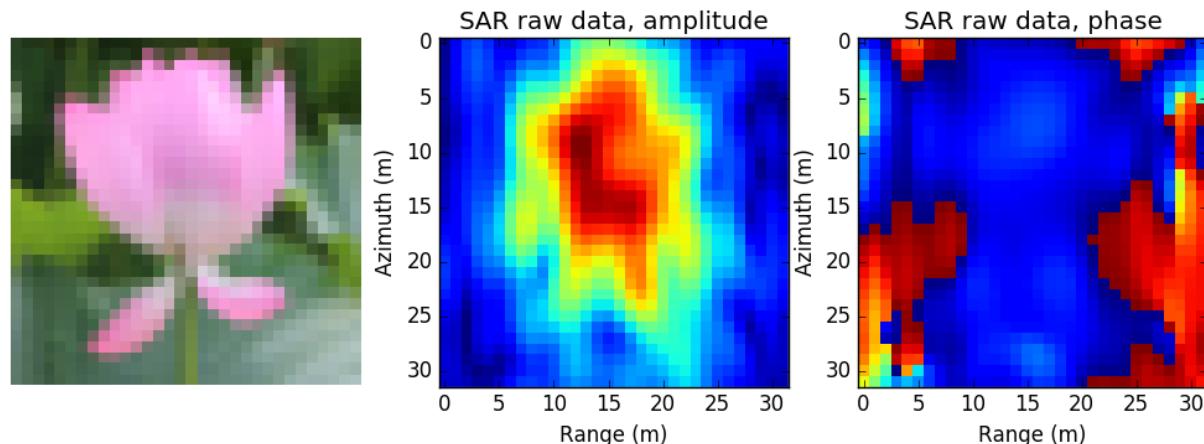


图 7.79: 仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下

7.5.4.4.2.3 实验结果

点目标结果

- 运行时间:
- 重构误差:

7.5.4.5 压缩感知成像

7.5.4.5.1 压缩感知 SAR 成像

越来越多的基于压缩感知的 SAR 成像方法被提出 [2][3][4].

匹配追踪用于量子模拟 [1]

SAR imaging process can be formulated as

$$\mathbf{s} = \mathbf{A}\mathbf{g} + \mathbf{n},$$

where, \mathbf{s} is the $m = MN \times 1$ received SAR raw data vector in phase history domain, \mathbf{g} is the $n = HW \times 1$ reflection vector of scene. \mathbf{A} represents the mapping from scene to SAR raw data. \mathbf{n} is the noise vector.

If \mathbf{g} is not sparse enough, assume that exist a basis $\mathbf{D} = (\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n)$ that satisfies $\mathbf{g} = \mathbf{D}\mathbf{x}$, where, \mathbf{x} is a K sparse $n \times 1$ vector, and \mathbf{D} is the so called dictionary matrix of size $n \times n$.

Our goal is minimize

$$\min_{\mathbf{x}} \|\mathbf{x}\|_p, \quad s.t. \quad \mathbf{s} = \mathbf{A}\mathbf{D}\mathbf{x} + \mathbf{n},$$

i.e.

$$\min_{\mathbf{x}} = \|\mathbf{s} - \mathbf{A}\mathbf{D}\mathbf{x}\|_2 + \lambda \|\mathbf{x}\|_p,$$

where, λ is the balance factor, and $|\cdot|_p$, ($0 < p < 1$) is the ℓ_p norm.

Let $\Phi = \mathbf{AD}$, then we have

$$\min_{\mathbf{x}} = \|\mathbf{s} - \Phi\mathbf{x}\|_2 + \lambda \|\mathbf{x}\|_p.$$

Note that, if $\mathbf{s}, \Phi, \mathbf{x} \in \mathbb{C}$, the problem changes to

$$\text{Re}(\mathbf{s}) + j\text{Im}(\mathbf{s}) = \text{Re}(\Phi\mathbf{x}) + j\text{Im}(\Phi\mathbf{x})$$

so we have:

$$\begin{bmatrix} \text{Re}(\mathbf{s}) \\ \text{Im}(\mathbf{s}) \end{bmatrix} = \begin{bmatrix} \text{Re}(\Phi) & -\text{Im}(\Phi) \\ \text{Im}(\Phi) & \text{Re}(\Phi) \end{bmatrix} \begin{bmatrix} \text{Re}(\mathbf{x}) \\ \text{Im}(\mathbf{x}) \end{bmatrix}$$

提示: 对于非稀疏场景, 无需先对场景进行稀疏表示, 再进行观测; 然而在重构信号时, 由于信号非稀疏, 需要假设其在某一字典下稀疏.

ABC

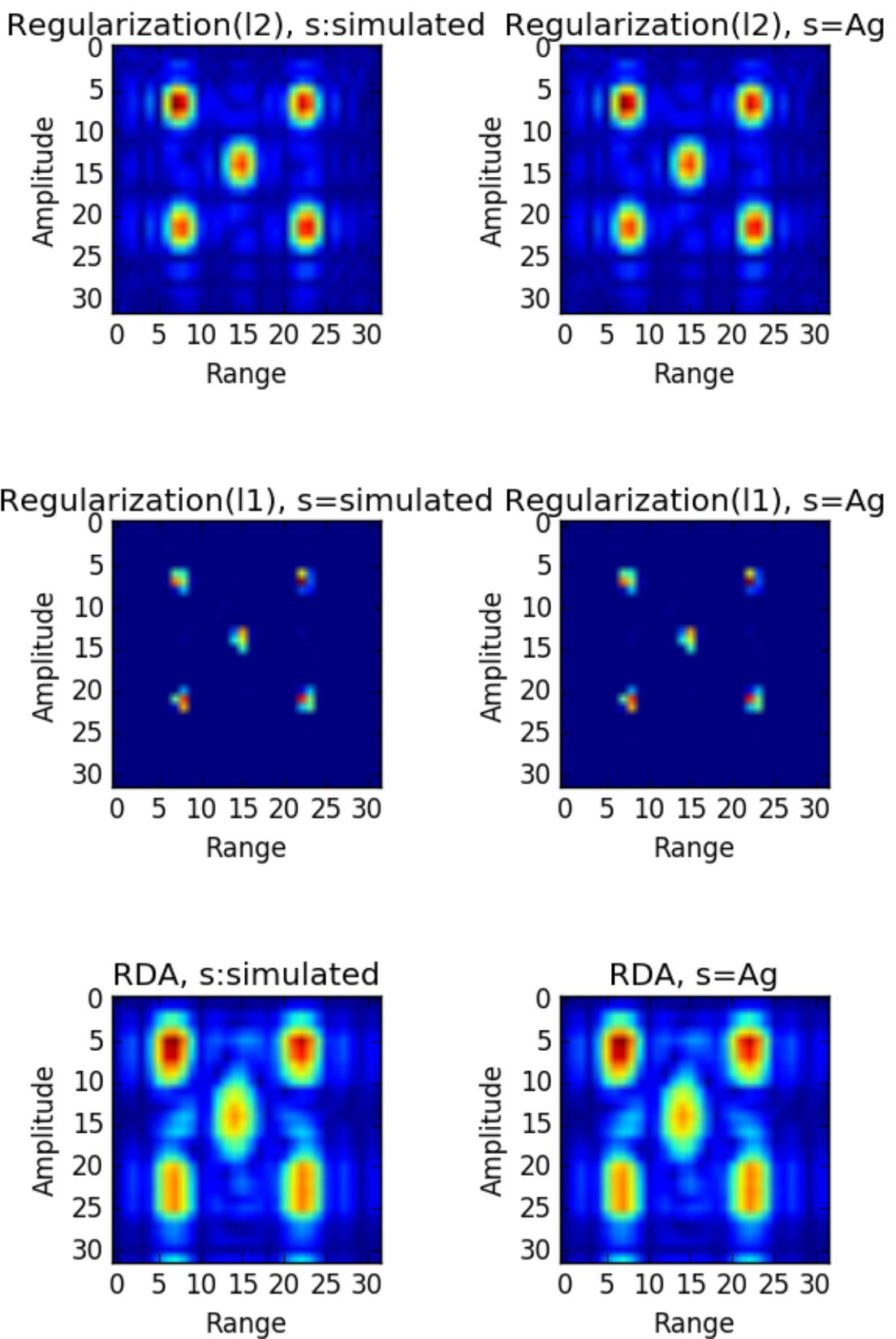
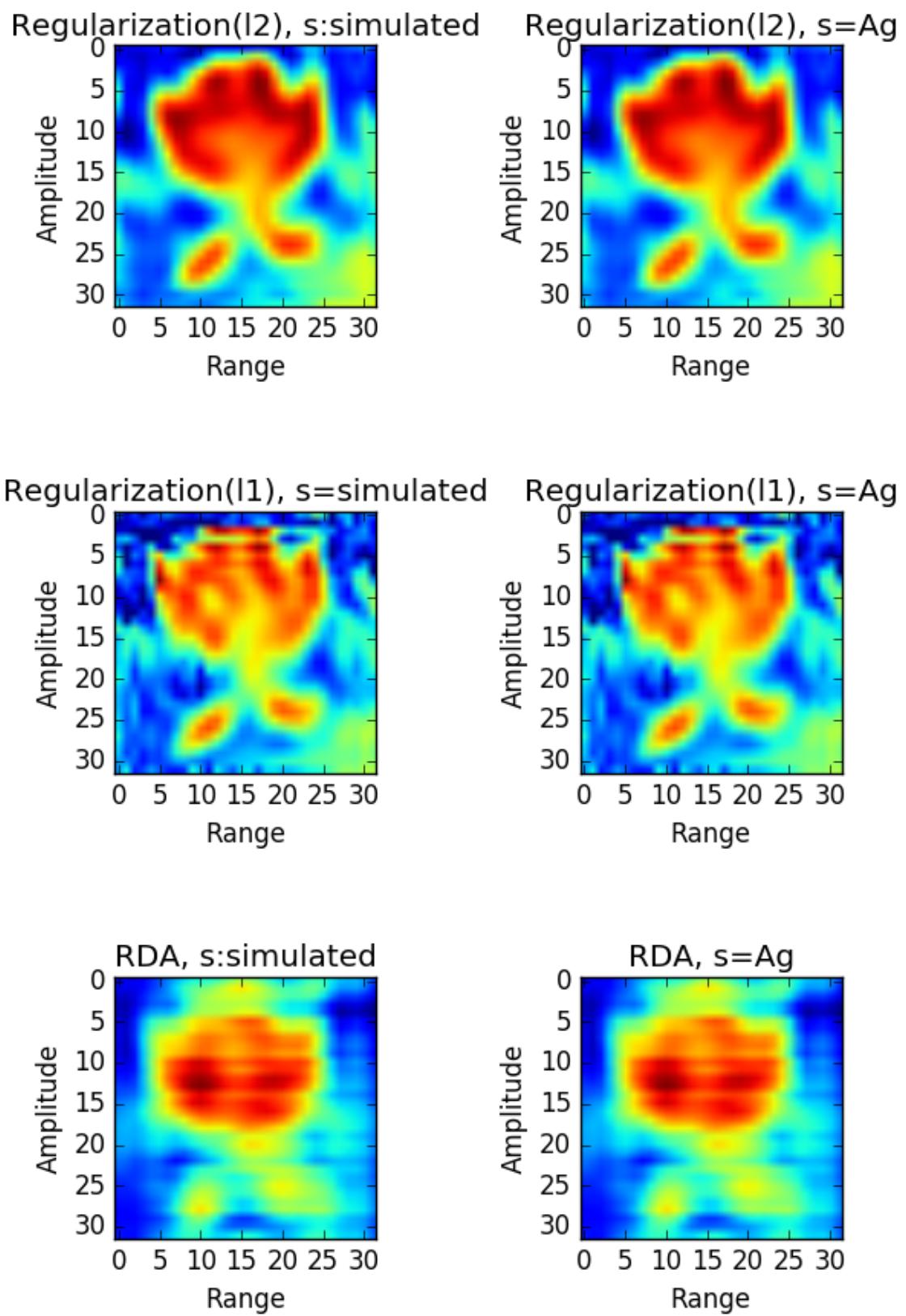


图 7.80: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

图 7.81: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

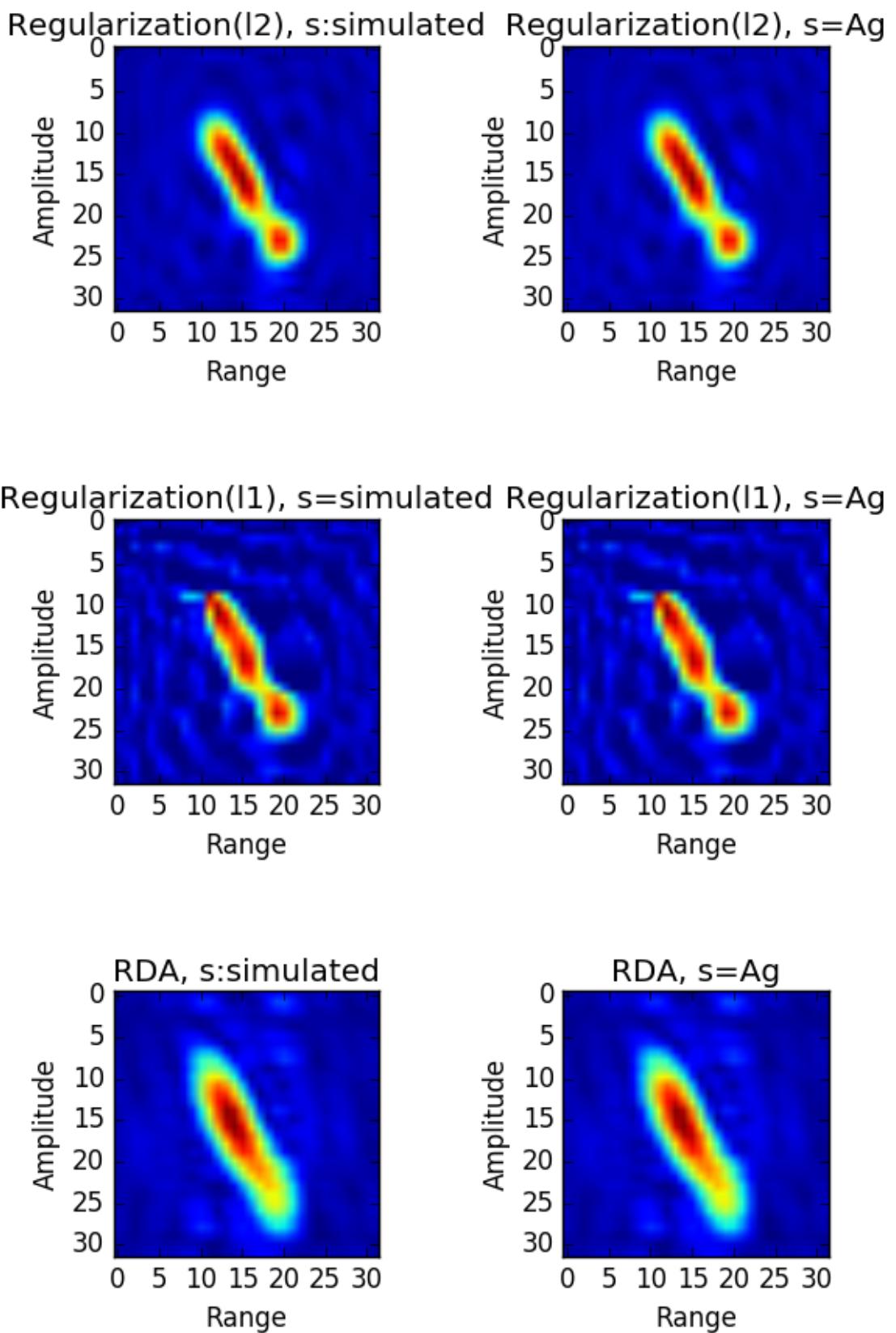


图 7.82: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

7.5.4.5.2 实验与分析

7.5.4.5.2.1 仿真数据

7.5.4.5.2.2 实验说明

- 仿真场景大小: 32×32
- 回波矩阵大小: 32×32
- 稀疏表示字典: 无, DCT , DWT
- 优化方法: Lasso , OMP

仿真点目标场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

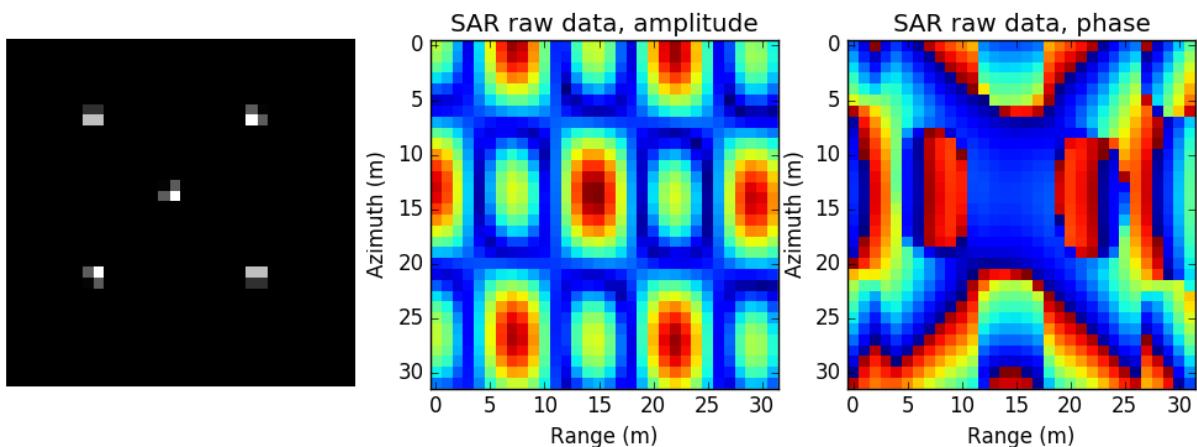


图 7.83: 仿真点目标场景图, 仿真生成点 SAR 原始数据幅度相位图

仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

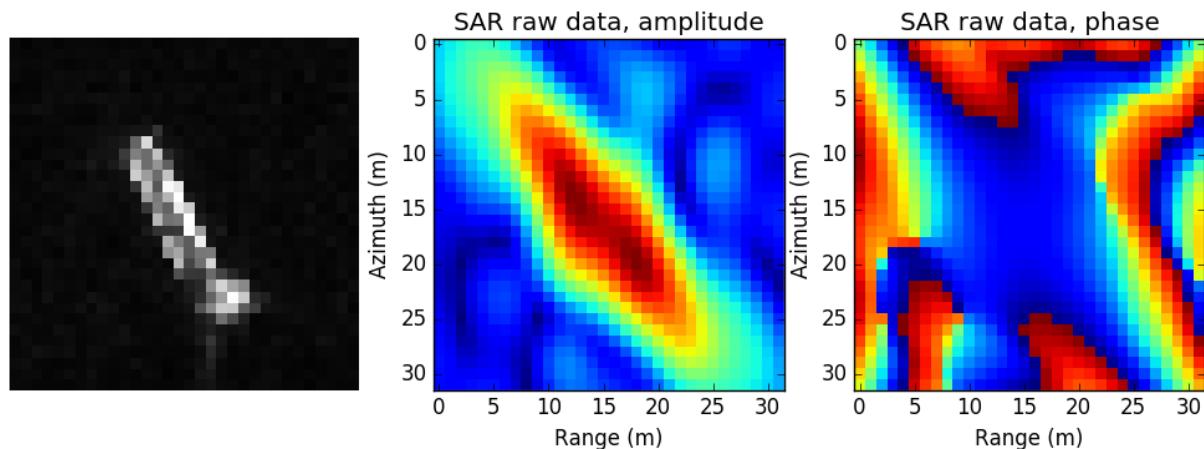


图 7.84: 仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下

仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

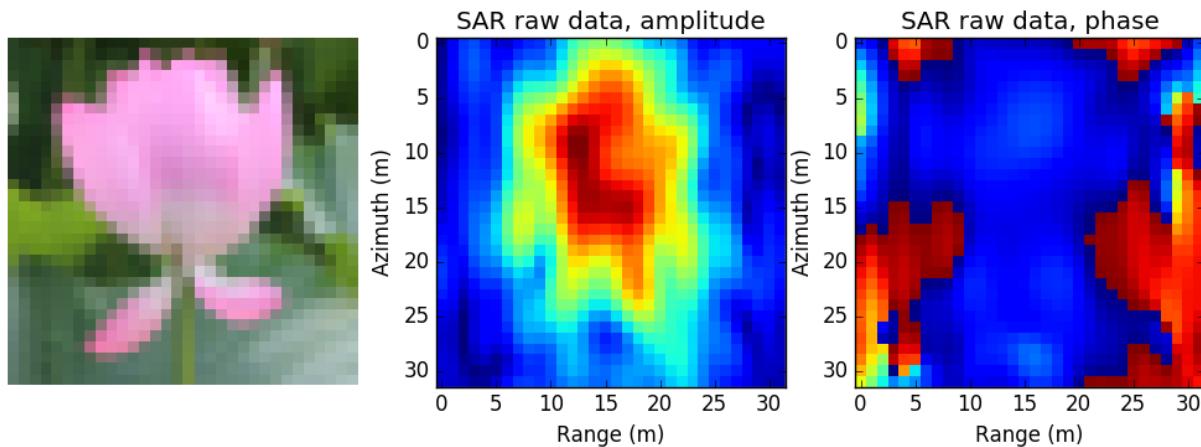


图 7.85: 仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下

7.5.4.5.2.3 实验代码

7.5.4.5.2.4 实验结果

1. OMP 优化, 不采用字典进行稀疏表示, 和采用 DCT 字典进行稀疏表示的结果如下:

点目标结果

船只结果:

荷花结果:

2. Lasso 优化, 不采用字典进行稀疏表示, 和采用 DCT 字典进行稀疏表示的结果如下:

点目标结果

船只结果:

荷花结果:

注解: 由实验结果可知:

- 场景的稀疏性决定了重构的性能, 越稀疏重构越精确
- 字典的选择很重要

7.5.4.5.2.5 真实数据

7.5.4.5.2.6 实验说明

由于压缩感知方法占用内存空间大, 该数据为 RADARSAT1 一景数据中的一小块区域, 区域大小为 128×128 .

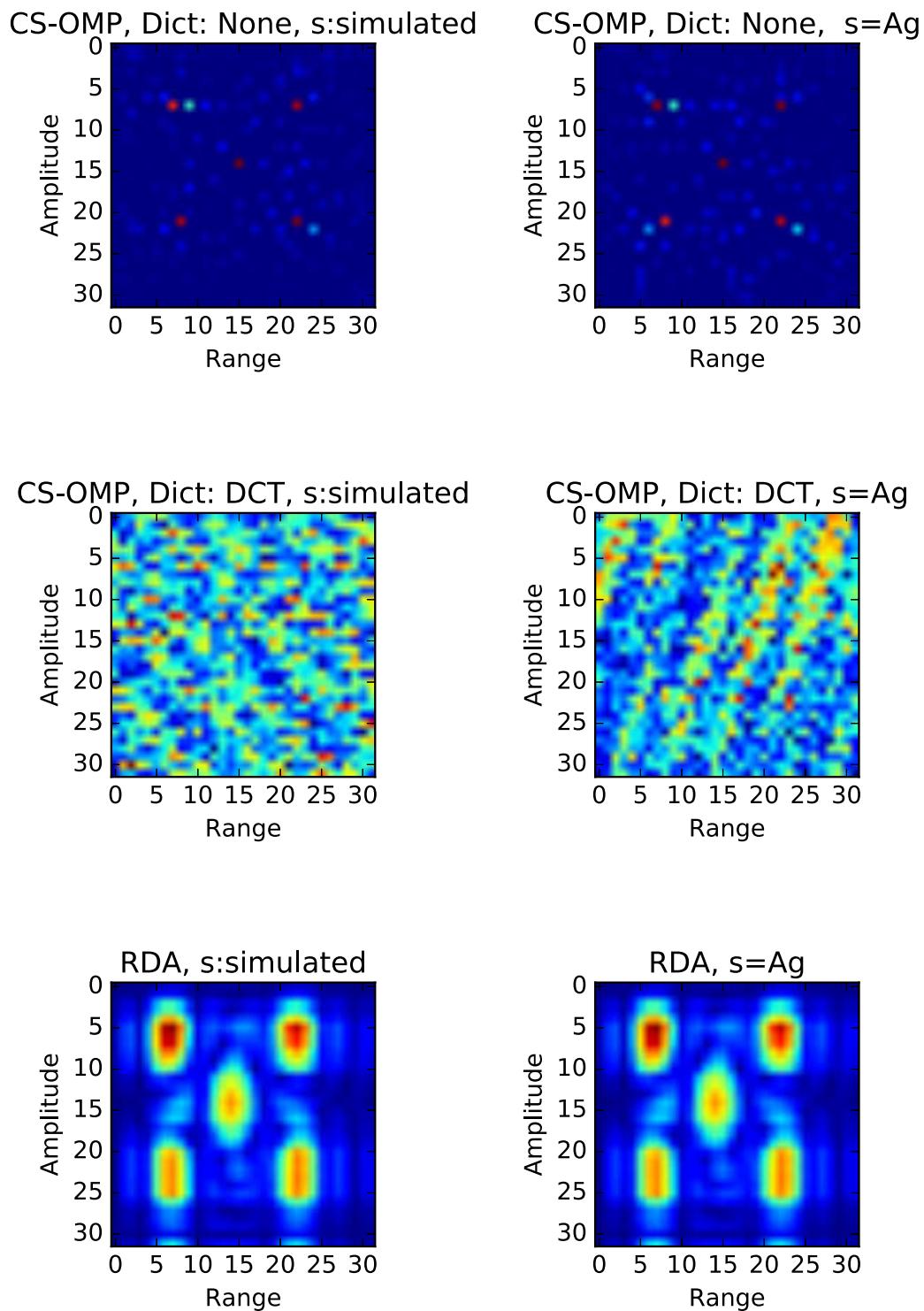


图 7.86: Compressive Sensing based and RDA Imaging Results of points.

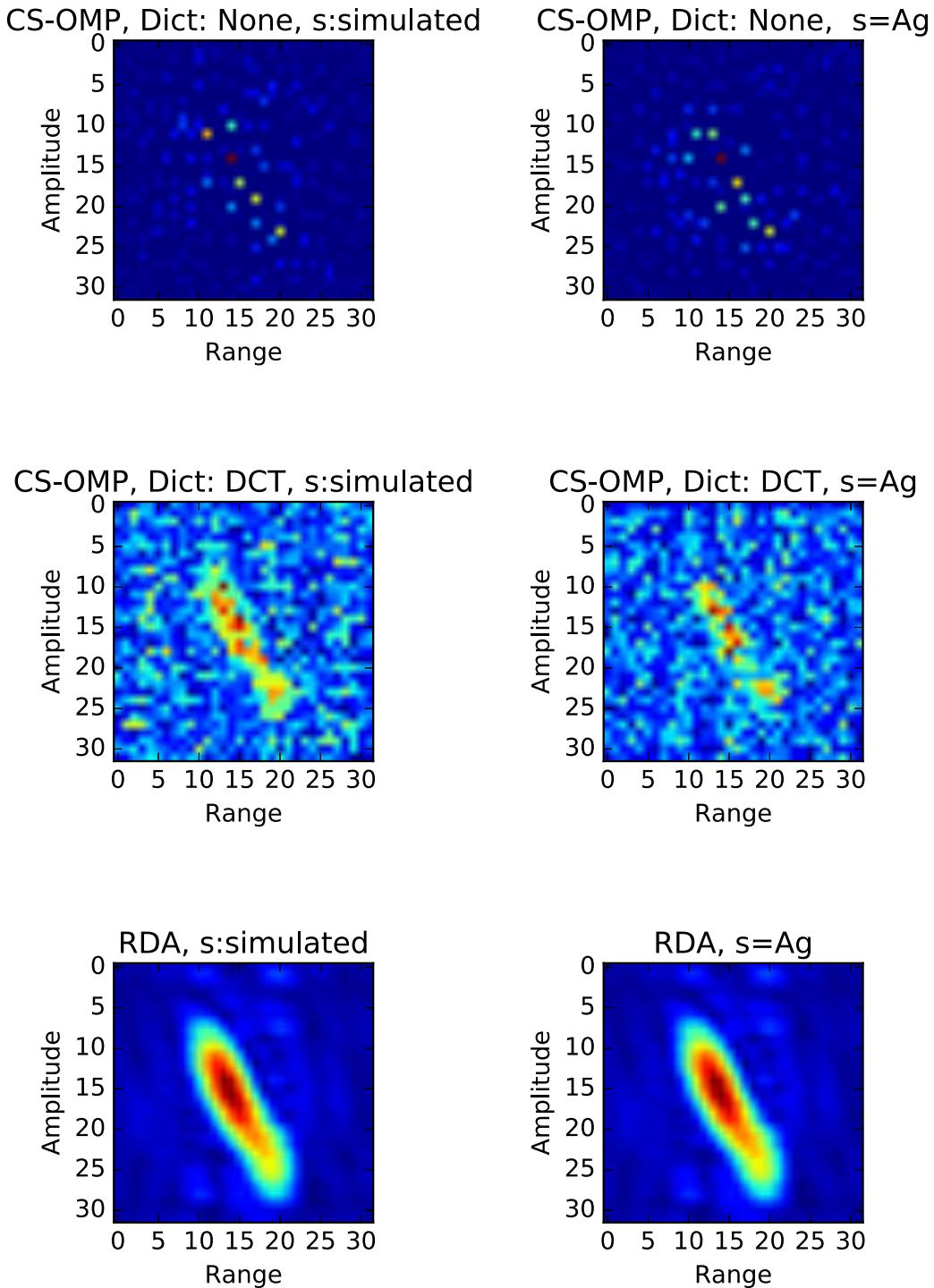


图 7.87: Compressive Sensing based and RDA Imaging Results of ship.

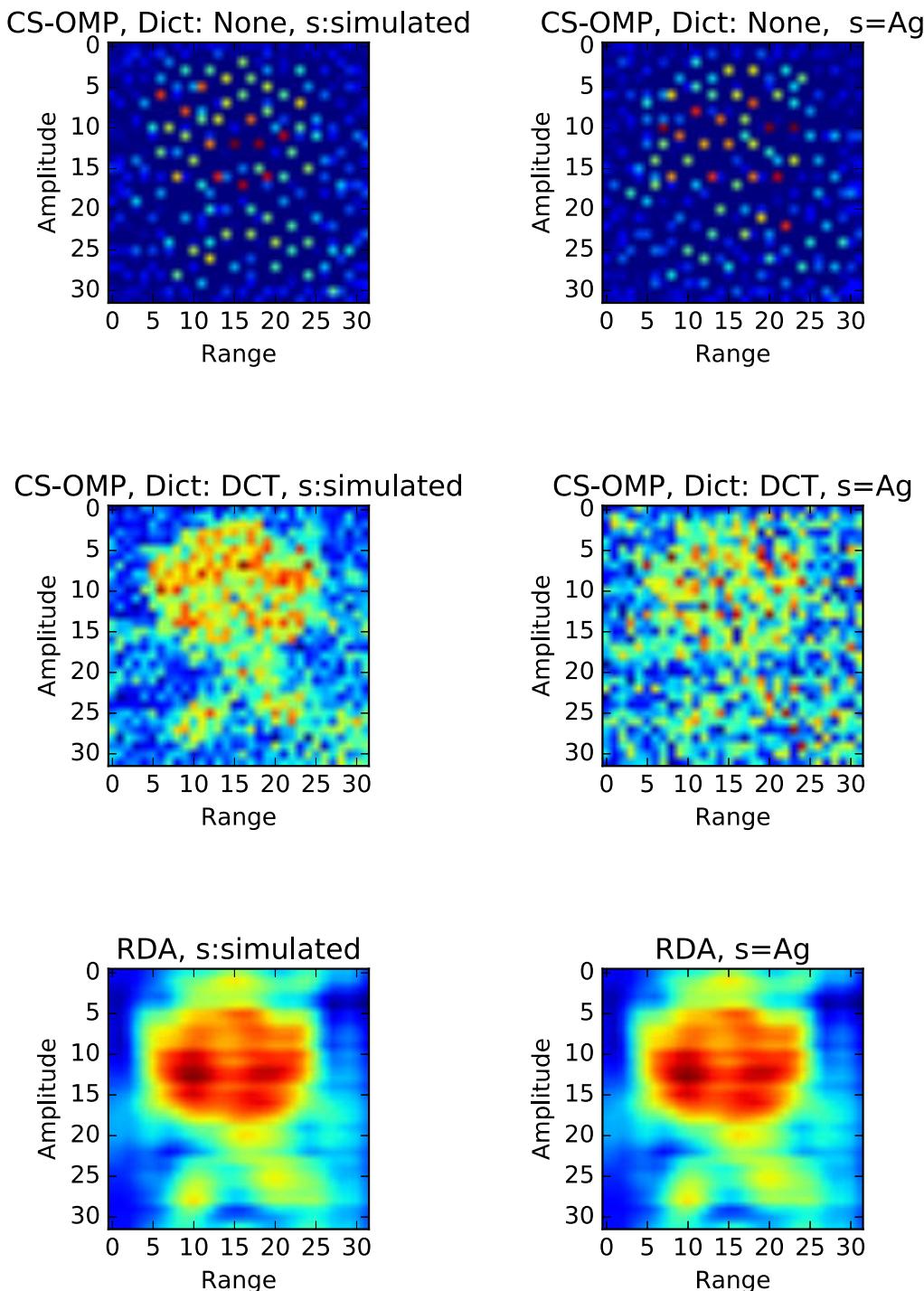


图 7.88: Compressive Sensing based and RDA Imaging Results of lotus.

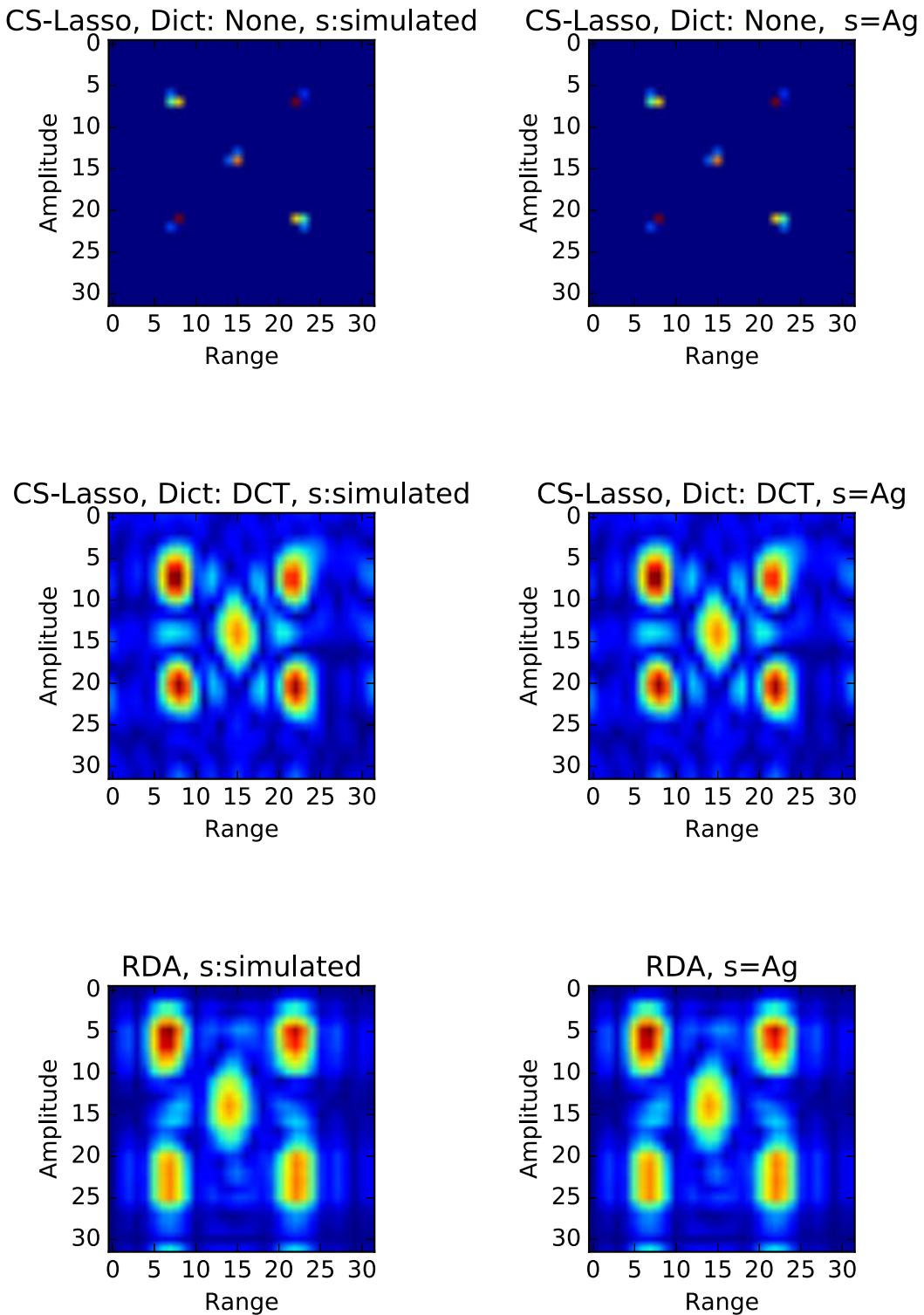


图 7.89: Compressive Sensing based and RDA Imaging Results of points.

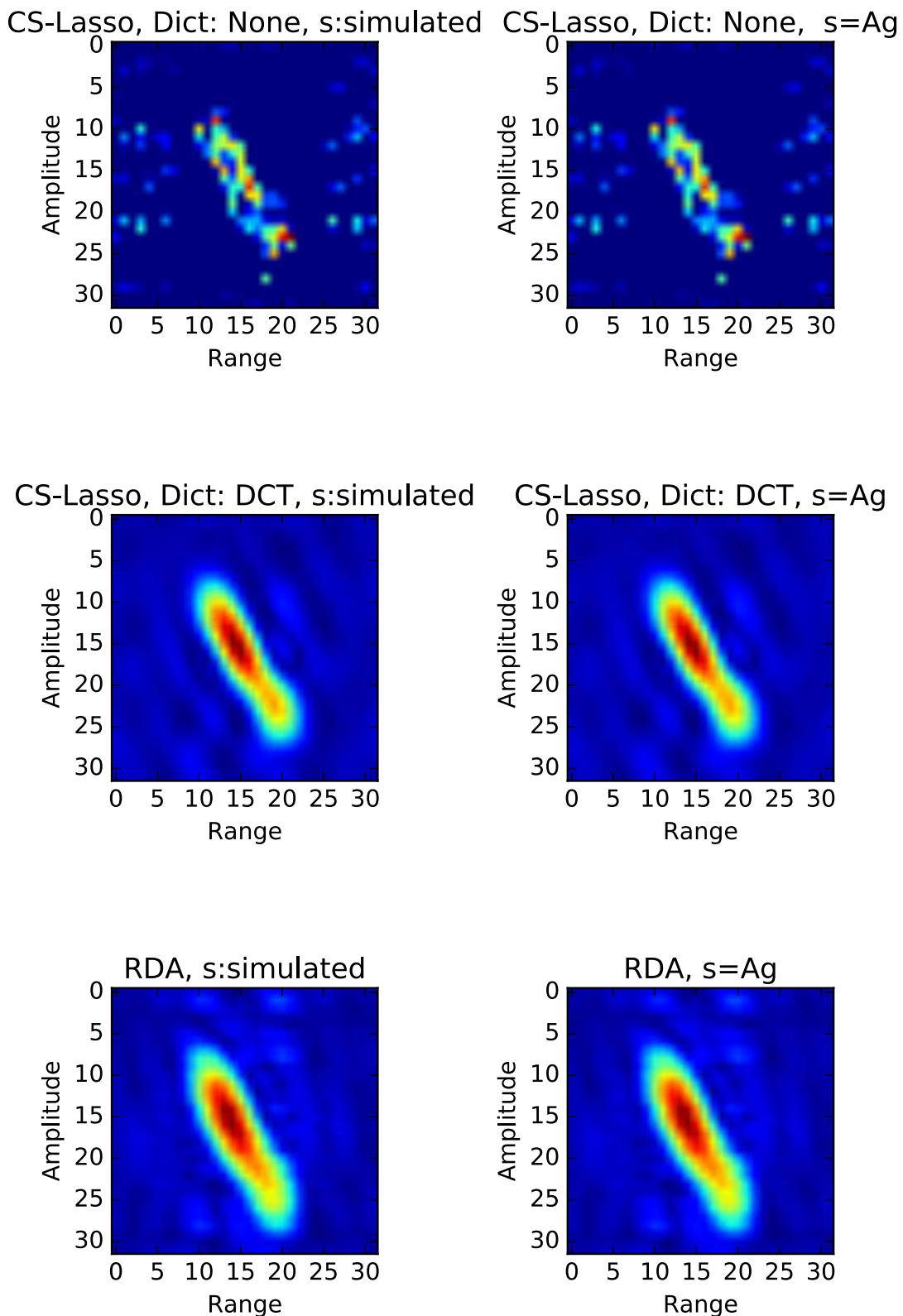


图 7.90: Compressive Sensing based and RDA Imaging Results of ship.

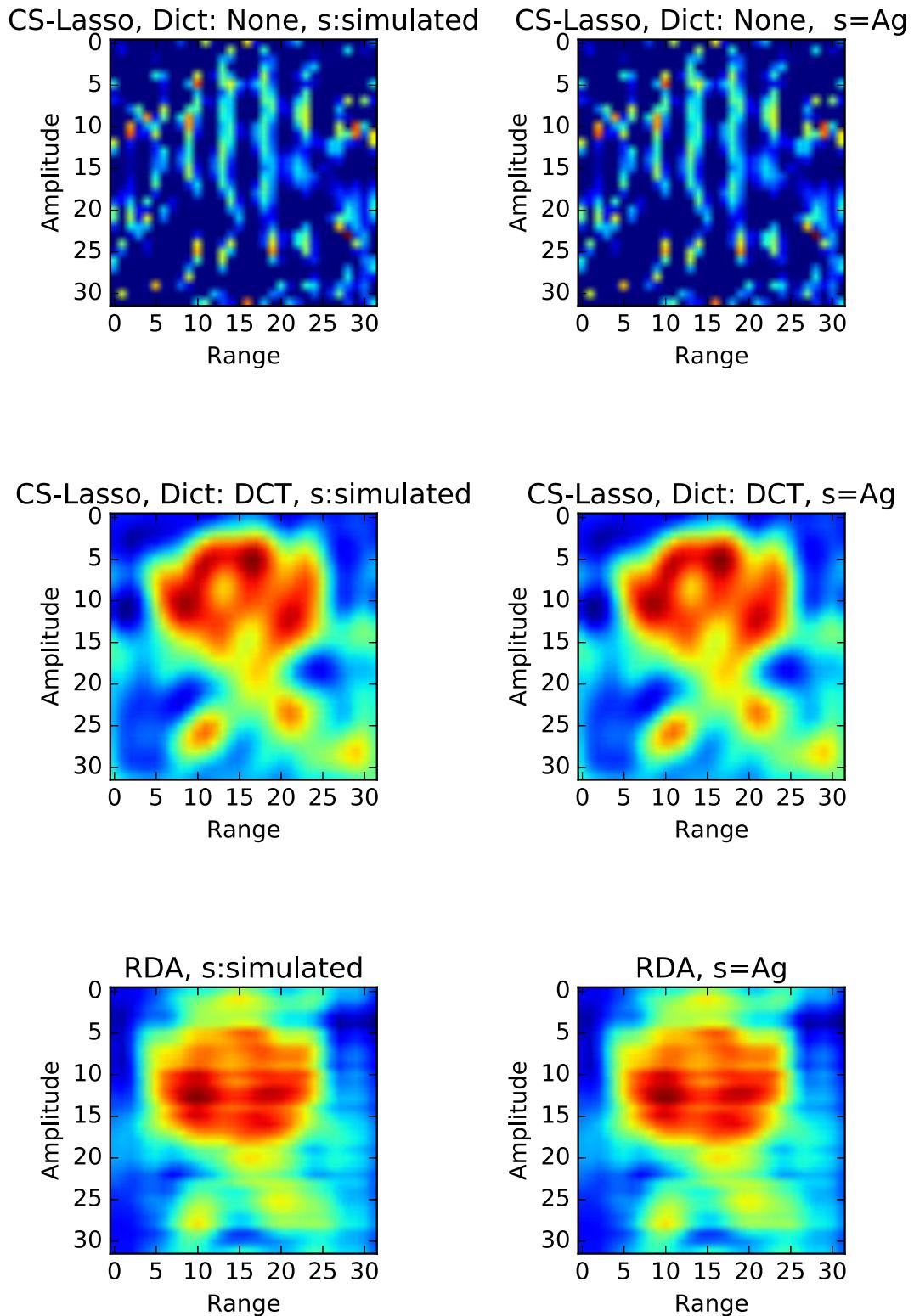


图 7.91: Compressive Sensing based and RDA Imaging Results of lotus.

7.5.4.5.2.7 实验代码

7.5.4.5.2.8 实验结果

7.5.5 SAR 超分辨成像

7.5.5.1 SAR 超分辨成像简介

7.5.5.1.1 不同成像方法结果对比

- 仿真场景大小: 128×128
- 回波矩阵大小: 32×32
- 数据生成
 - 通过模拟 SAR 成像过程生成 SAR 原始数据 s
 - 通过 $s = \mathbf{A}\mathbf{g}$ 生成 SAR 原始数据 s
- 成像方法
 - 广义逆: $\hat{\mathbf{g}} = \mathbf{A}^+ s$
 - 共轭转置: $\hat{\mathbf{g}} = \mathbf{A}^H s$
 - 距离多普勒方法: 匹配滤波, 距离徙动校正等.

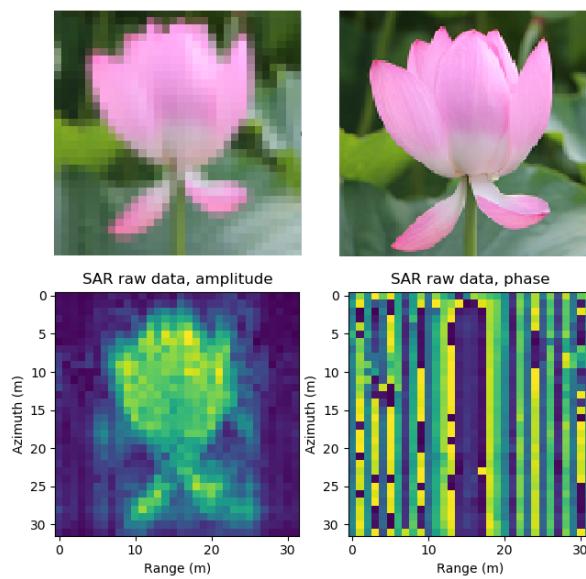


图 7.92: 原始彩色荷花图 (左) 与仿真得到的 SAR 原始数据幅度 (中) 与相位 (右).

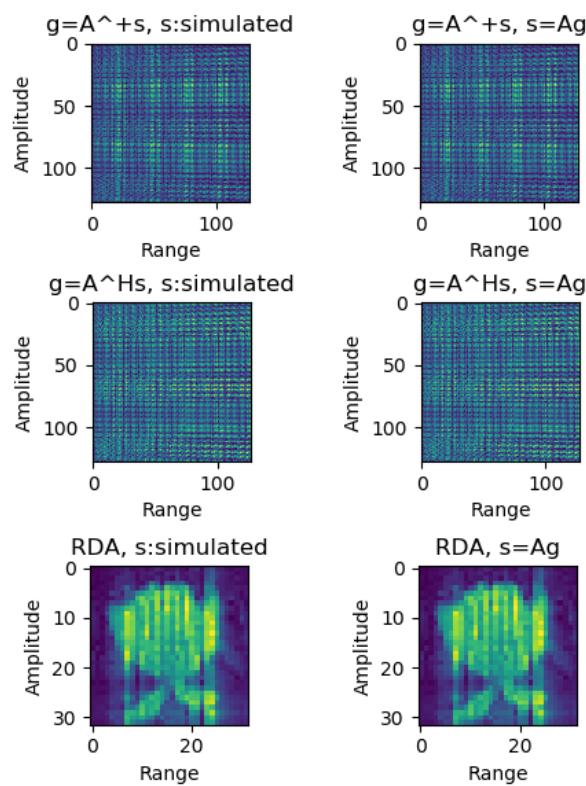


图 7.93: 不同方法成像结果

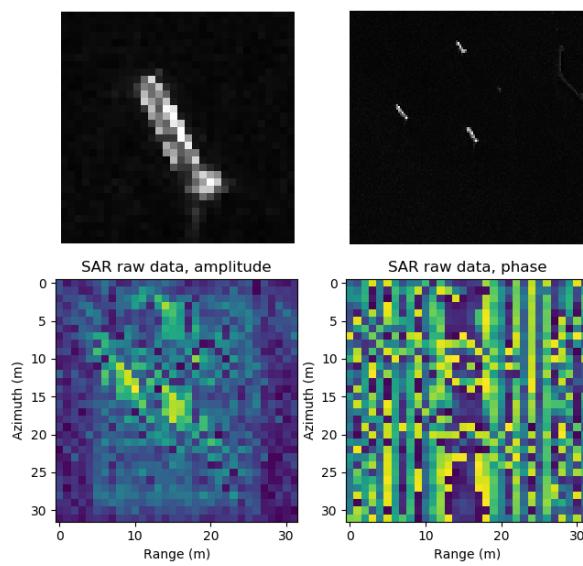


图 7.94: 原始船只图 (左) 与仿真得到的 SAR 原始数据幅度 (中) 与相位 (右).

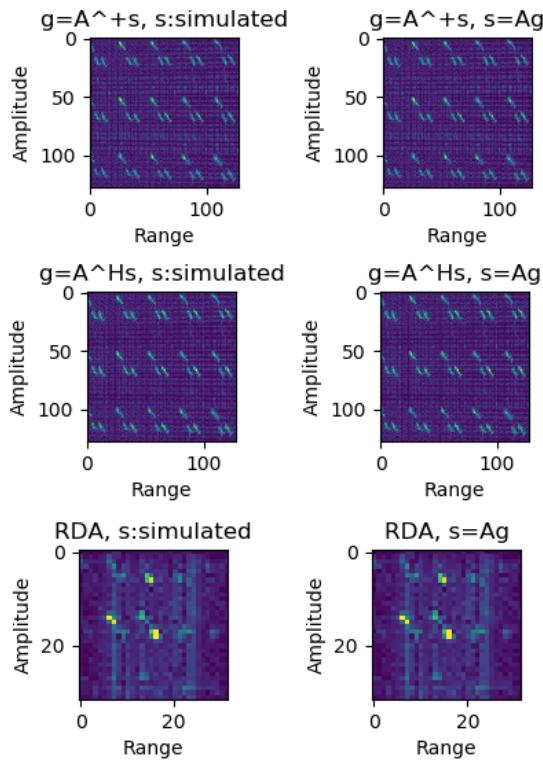


图 7.95: 不同方法成像结果

7.5.5.2 带宽外推超分辨 SAR 成像

7.5.5.2.1 带宽外推

7.5.5.3 经典谱估计超分辨 SAR 成像

7.5.5.3.1 经典谱估计

经典谱估计 (*Classical Spectral Estimation*) 又称 *无参化谱估计* (*Nonparametric Spectral Estimation*)

7.5.5.4 现代谱估计超分辨 SAR 成像

7.5.5.4.1 现代谱估计

现代谱估计 (*Modern Spectral Estimation*) 又称 *参数化谱估计* (*Parametric Spectral Estimation*) .

7.5.5.5 正则化超分辨成像方法

参见正则化成像方法 (页 481) 小节

7.5.5.5.1 实验与分析

7.5.5.5.1.1 实验说明

- 仿真场景大小: 128×128
- 回波矩阵大小: 32×32
- 稀疏表示字典: 无, DCT , DWT
- 优化方法: Lasso , OMP

仿真点目标场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

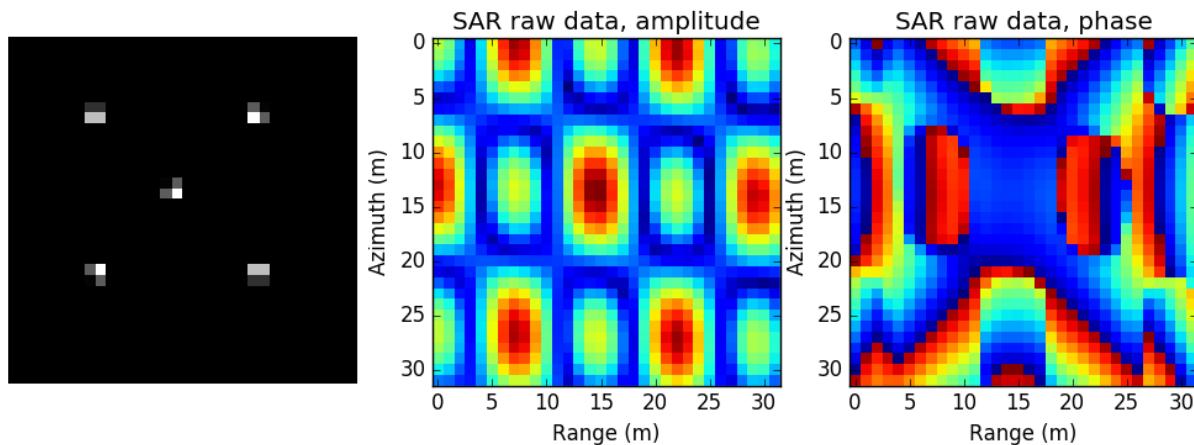


图 7.96: 仿真点目标场景图, 仿真生成点 SAR 原始数据幅度相位图

仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

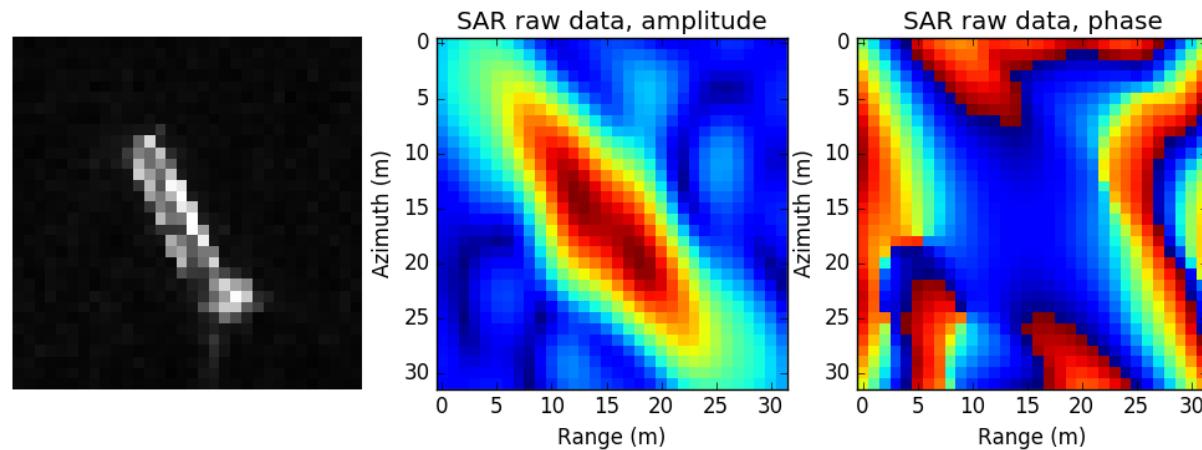


图 7.97: 仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下

仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

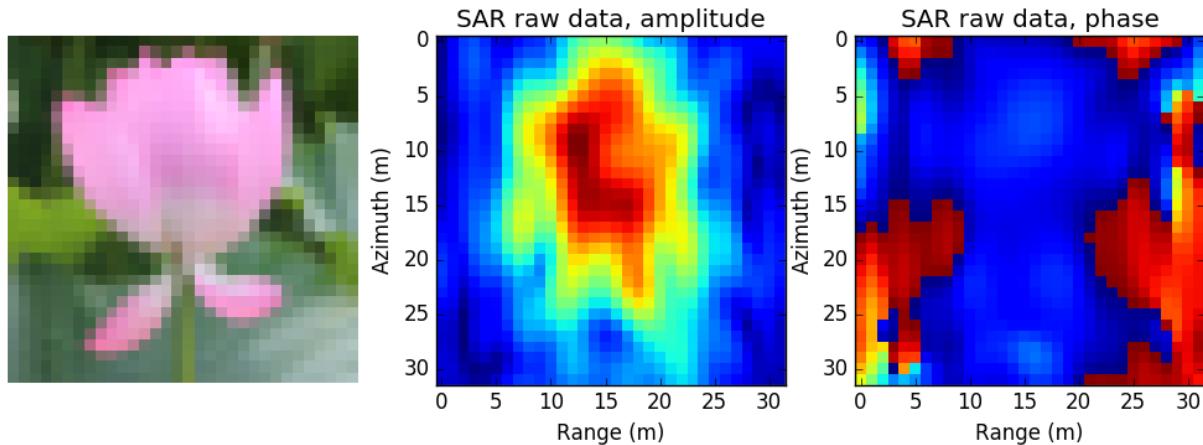


图 7.98: 仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下

7.5.5.5.1.2 实验代码

[iprs2.0](https://github.com/antsfamily/iprs2.0) (<https://github.com/antsfamily/iprs2.0>) demo_regular_sar.py

7.5.5.5.1.3 实验结果

- 运行时间: 平均约 100s
- 重构误差:

7.5.5.6 压缩感知超分辨 SAR 成像

7.5.5.6.1 压缩感知超分辨 SAR 成像

在压缩感知成像 (页 483) 小节已经介绍了压缩感知 SAR 成像原理.

压缩成像 [3],

[2]

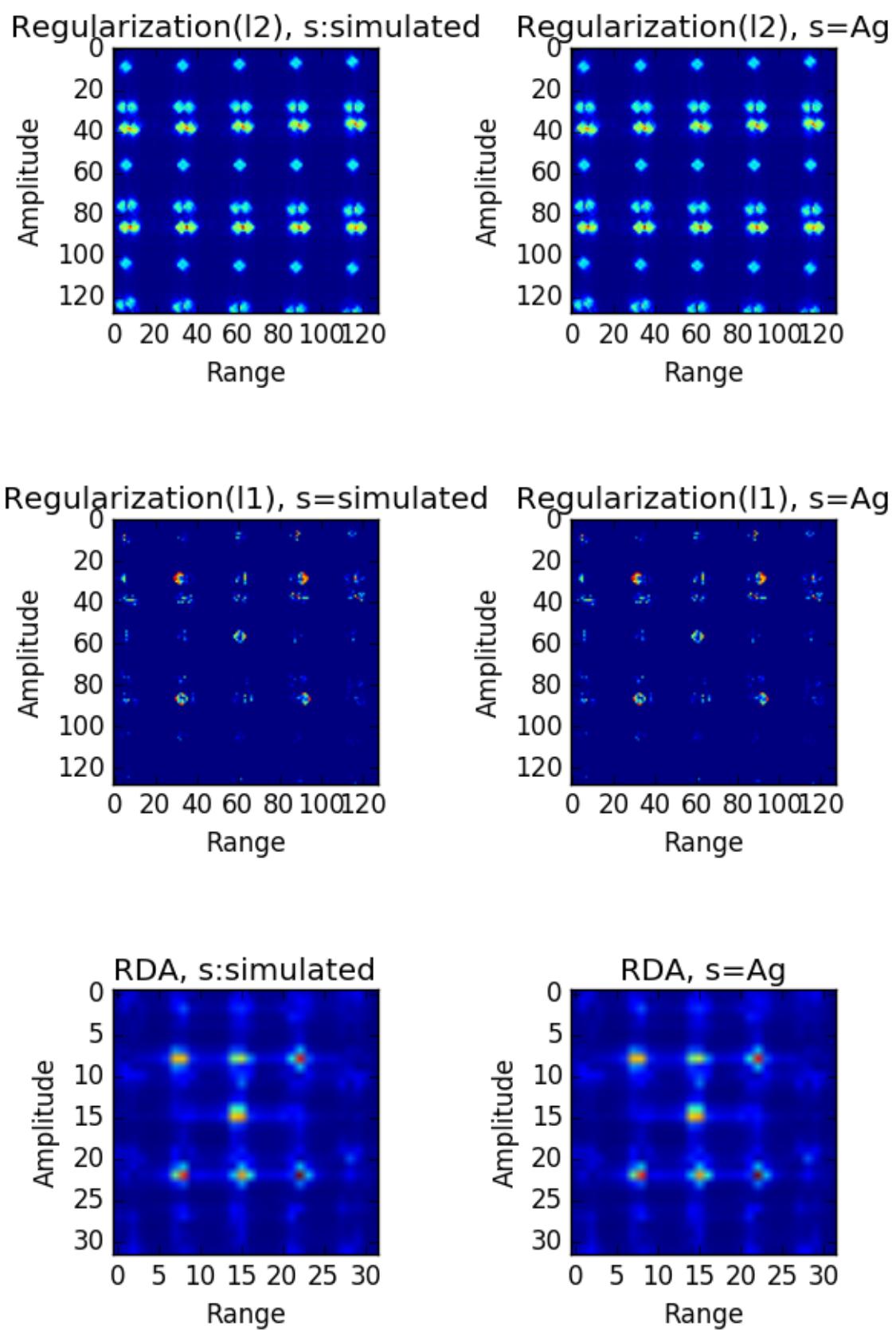
7.5.5.6.2 实验与分析

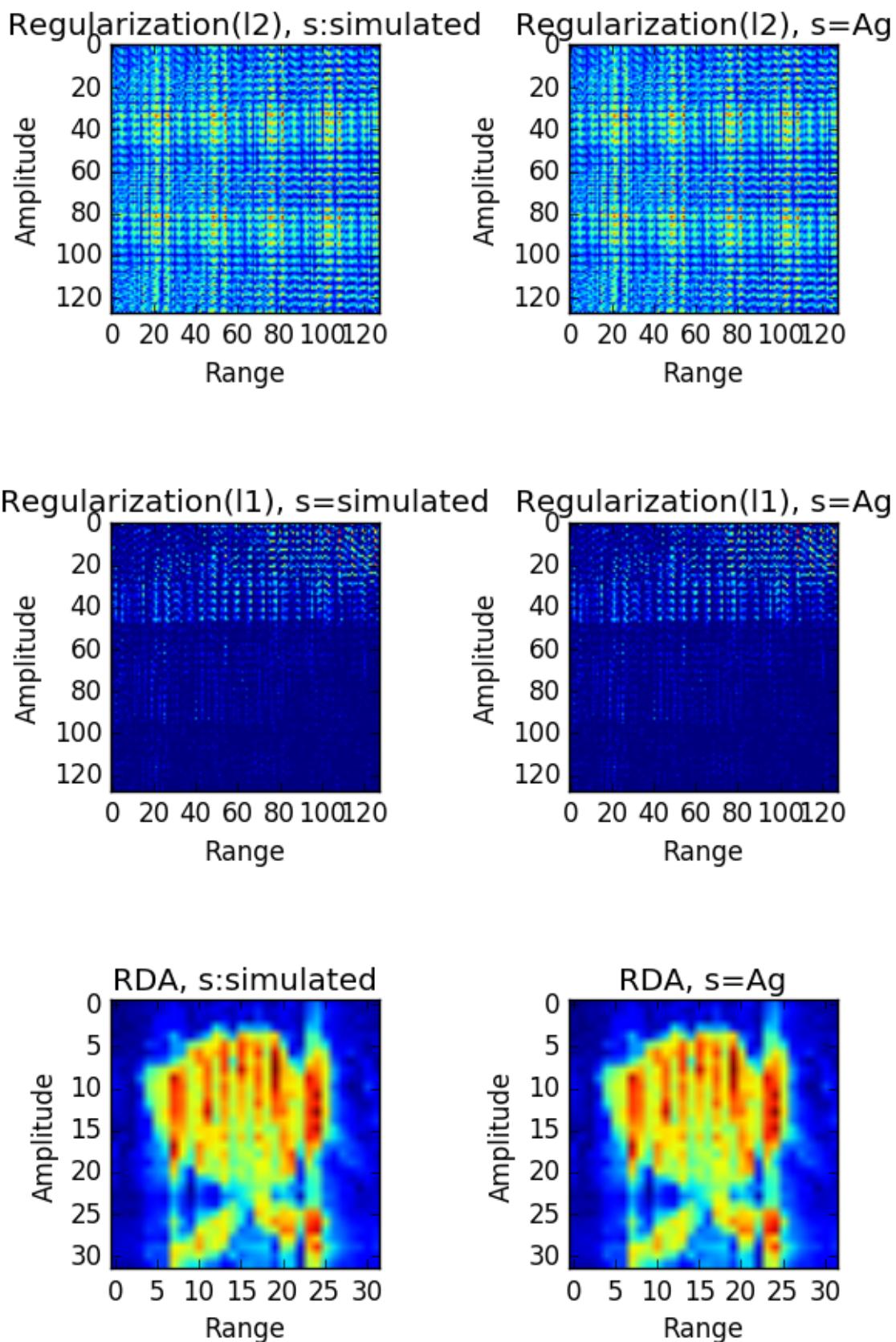
7.5.5.6.3 实验与分析

7.5.5.6.3.1 仿真数据

7.5.5.6.3.2 实验说明

- 仿真场景大小: 128×128
- 回波矩阵大小: 32×32
- 稀疏表示字典: 无, DCT, DWT

图 7.99: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

图 7.100: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

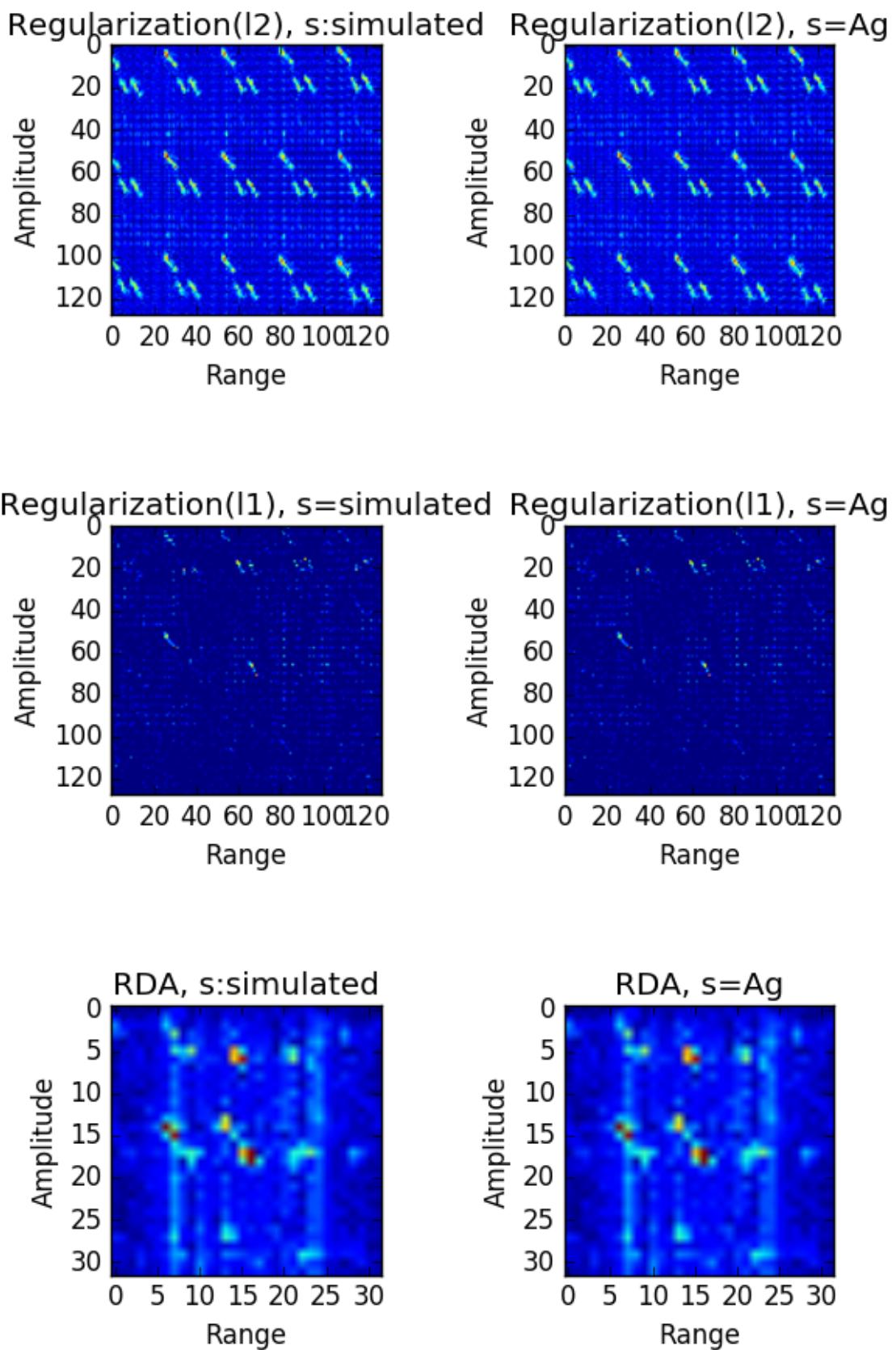


图 7.101: Imaging result of ℓ_1, ℓ_2 regularization and RDA. $\lambda = 0.001$, max iter 1000

- 优化方法: Lasso , OMP

仿真点目标场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

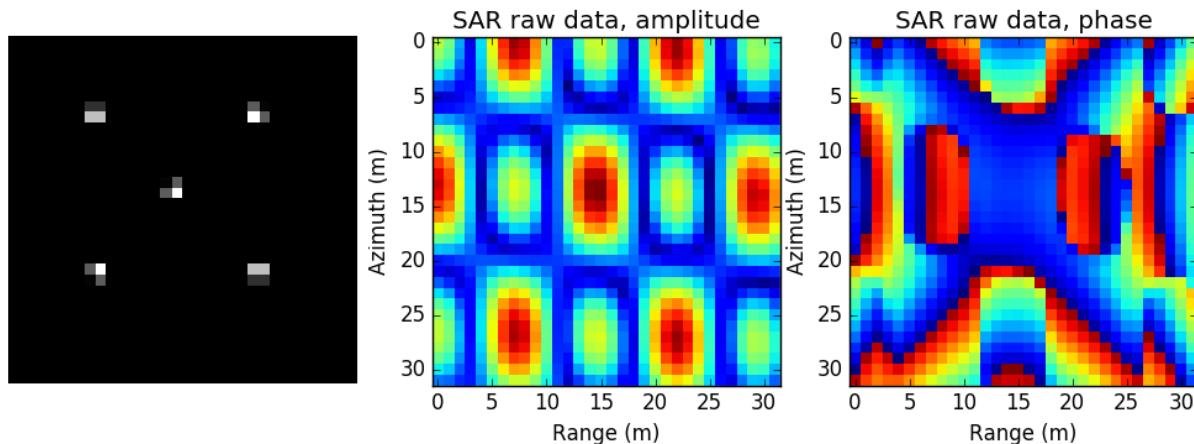


图 7.102: 仿真点目标场景图, 仿真 SAR 原始数据幅度相位图

仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

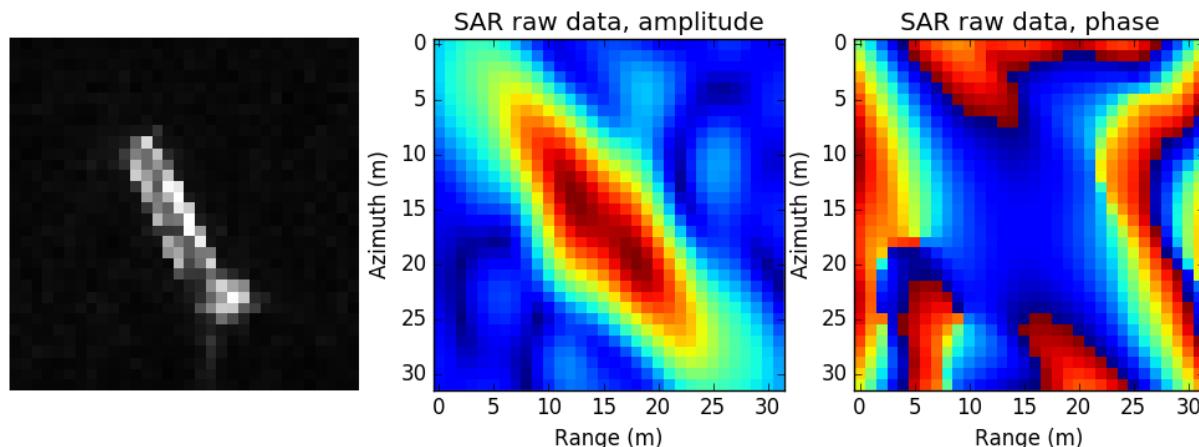


图 7.103: 仿真船只场景图及仿真生成点 SAR 原始数据幅度与相位图如下

仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下:

7.5.5.6.3.3 实验代码

7.5.5.6.3.4 实验结果

1. OMP 优化, 不采用字典进行稀疏表示, 和采用 DCT 字典进行稀疏表示的结果如下:

点目标结果

船只结果:

荷花结果:

2. Lasso 优化, 不采用字典进行稀疏表示, 和采用 DCT 字典进行稀疏表示的结果如下:

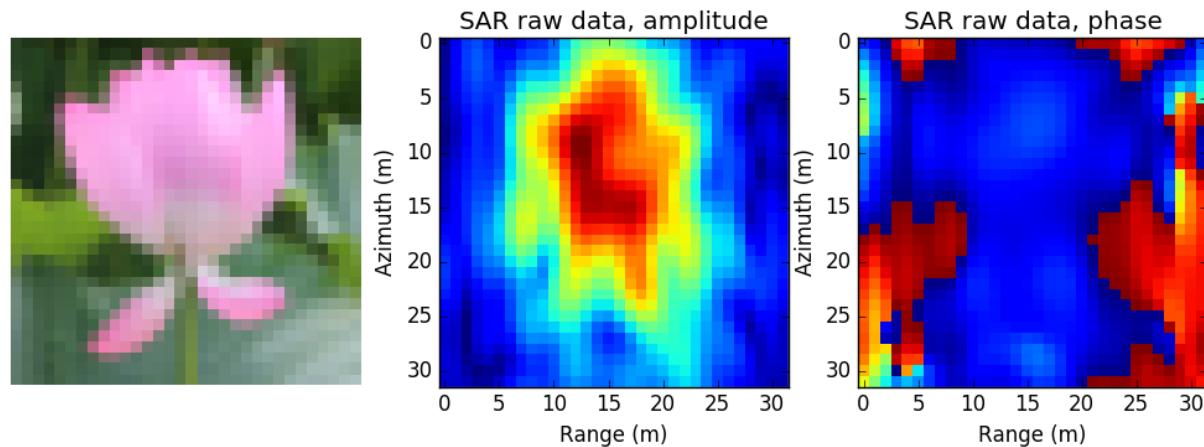


图 7.104: 仿真荷花场景图及仿真生成点 SAR 原始数据幅度与相位图如下

点目标结果

船只结果:

荷花结果:

7.5.5.6.3.5 真实数据

7.5.5.6.3.6 实验说明

7.5.5.6.3.7 实验代码

7.5.5.6.3.8 实验结果

7.5.6 深度学习与 SAR 成像

7.5.6.1 深度学习与 SAR 成像简介

7.5.6.2 学习式压缩感知成像

7.5.6.2.1 动机与贡献

注解: 动机:

1. 稀疏性与可压缩性限制:
 2. 高的恢复代价:
 3. CS
-

注解: 贡献:

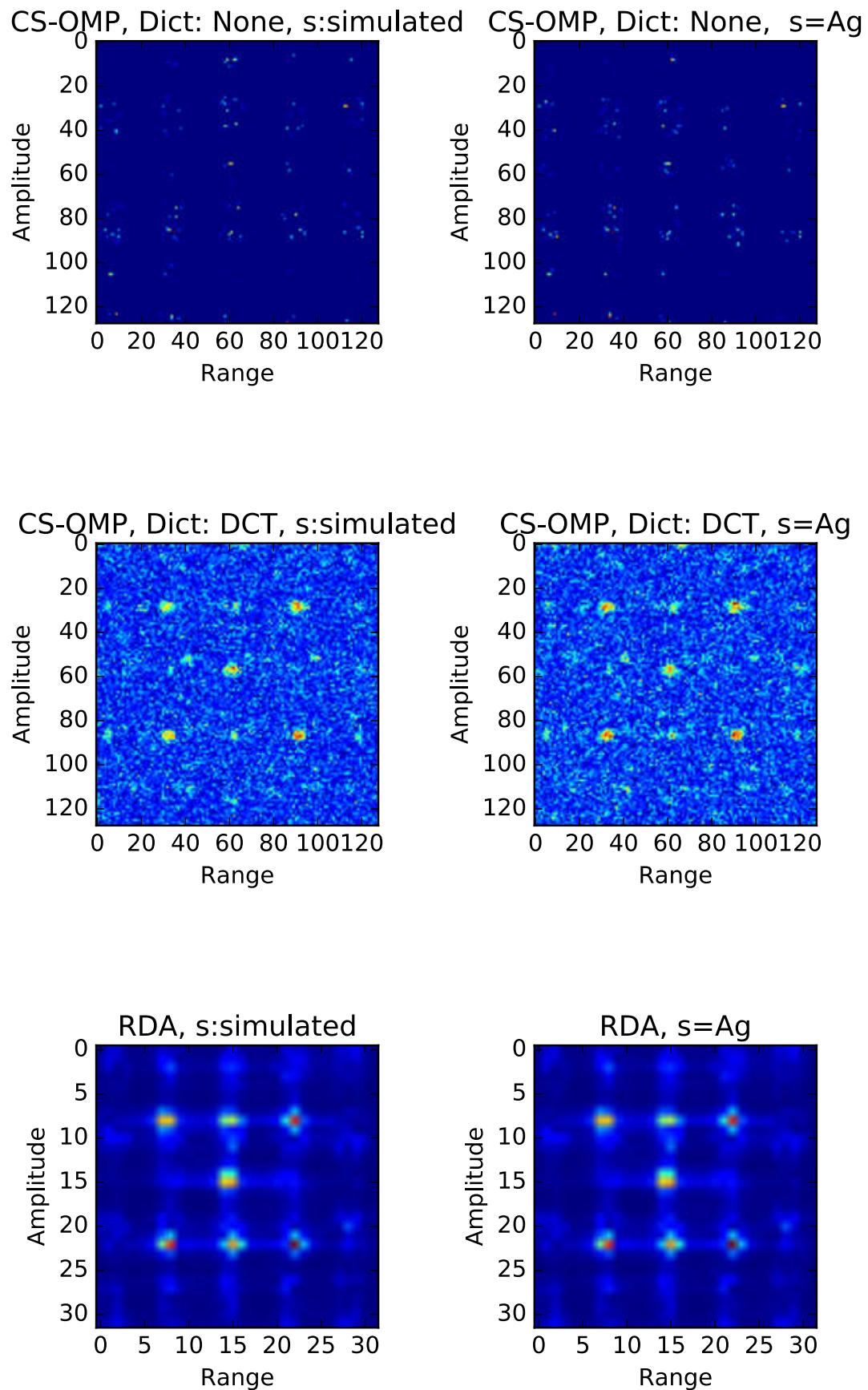


图 7.105: Compressive Sensing based and RDA Imaging Results of points (OMP).

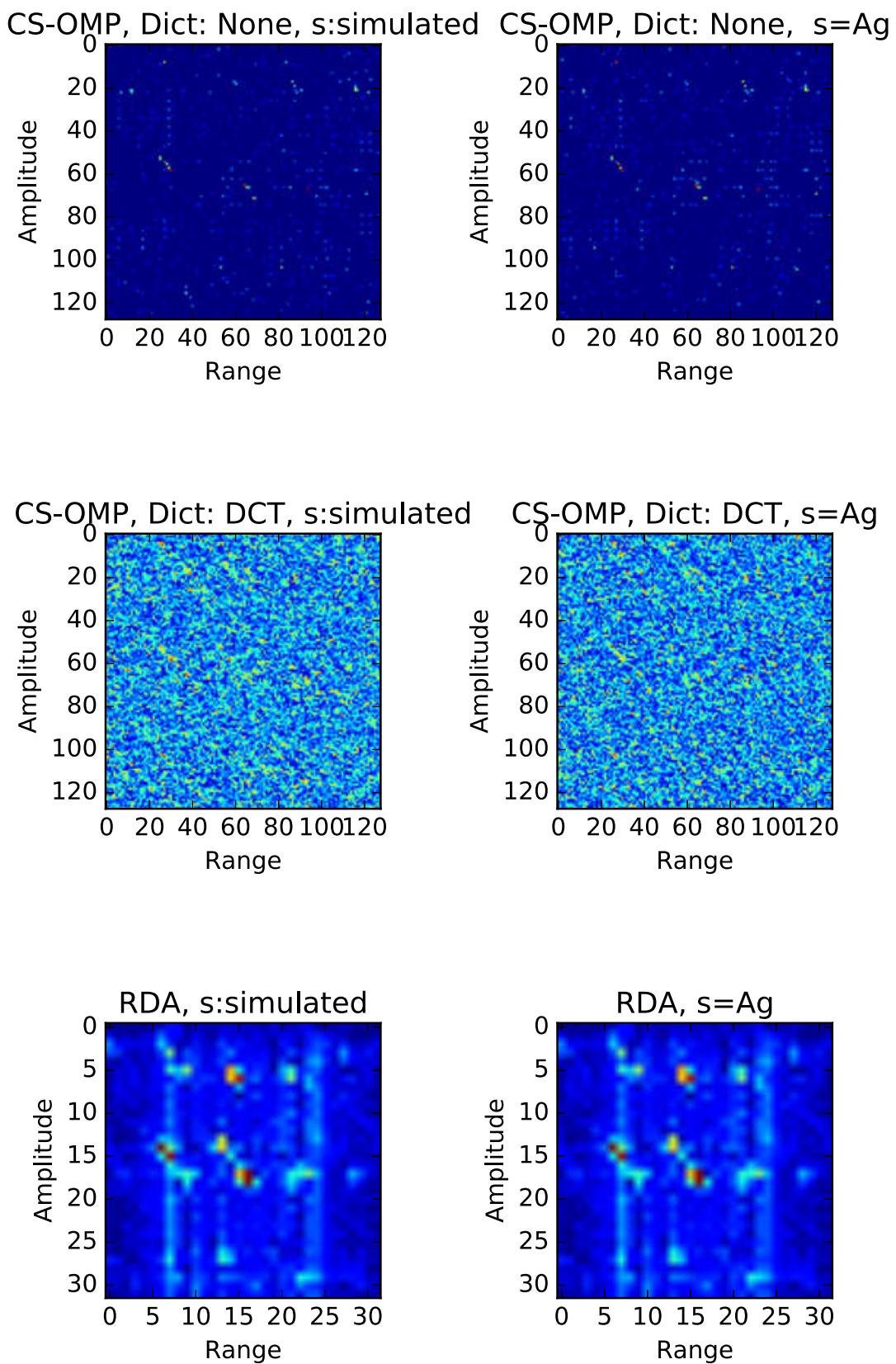


图 7.106: Compressive Sensing based and RDA Imaging Results of ship (OMP).

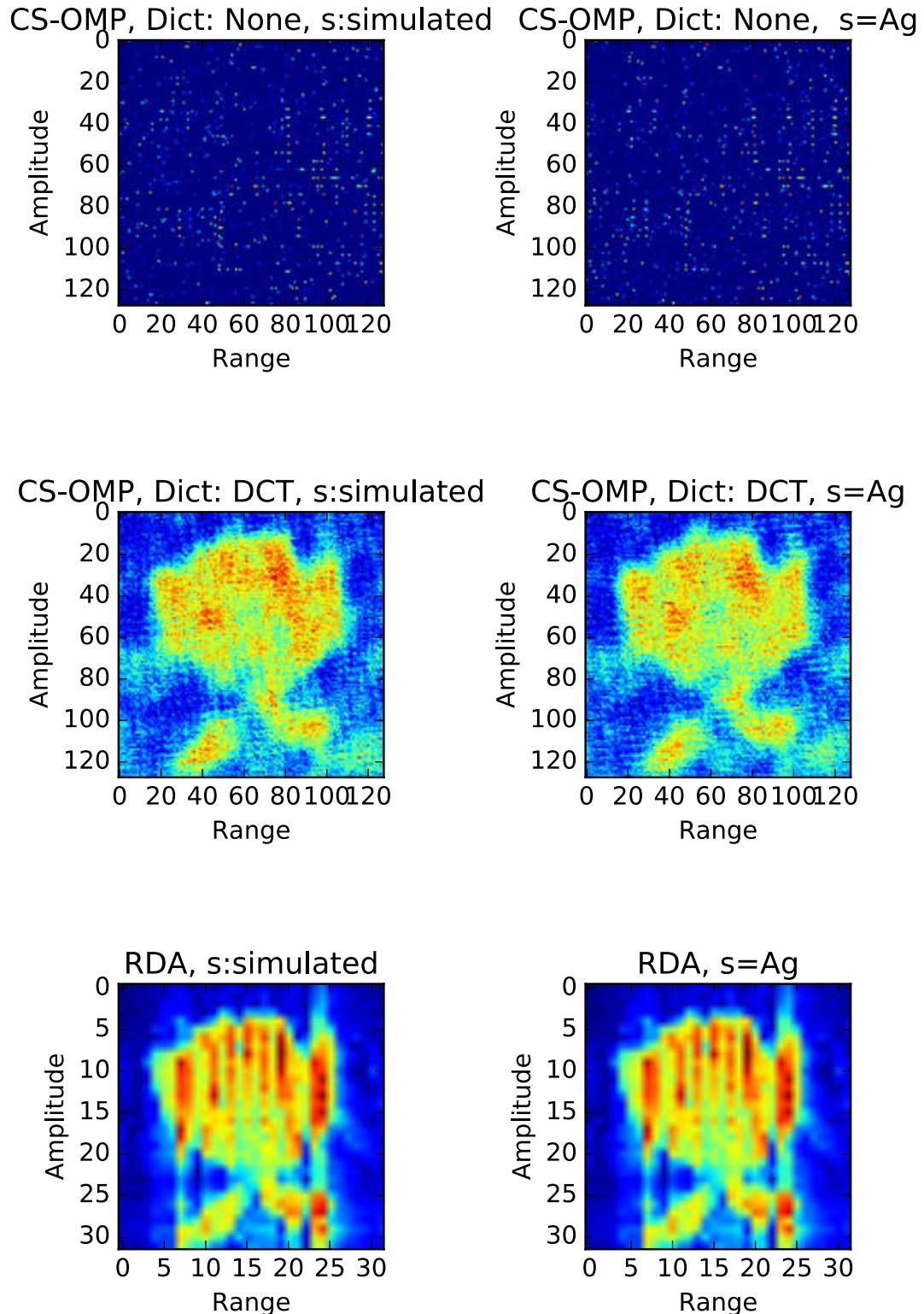


图 7.107: Compressive Sensing based and RDA Imaging Results of lotus.

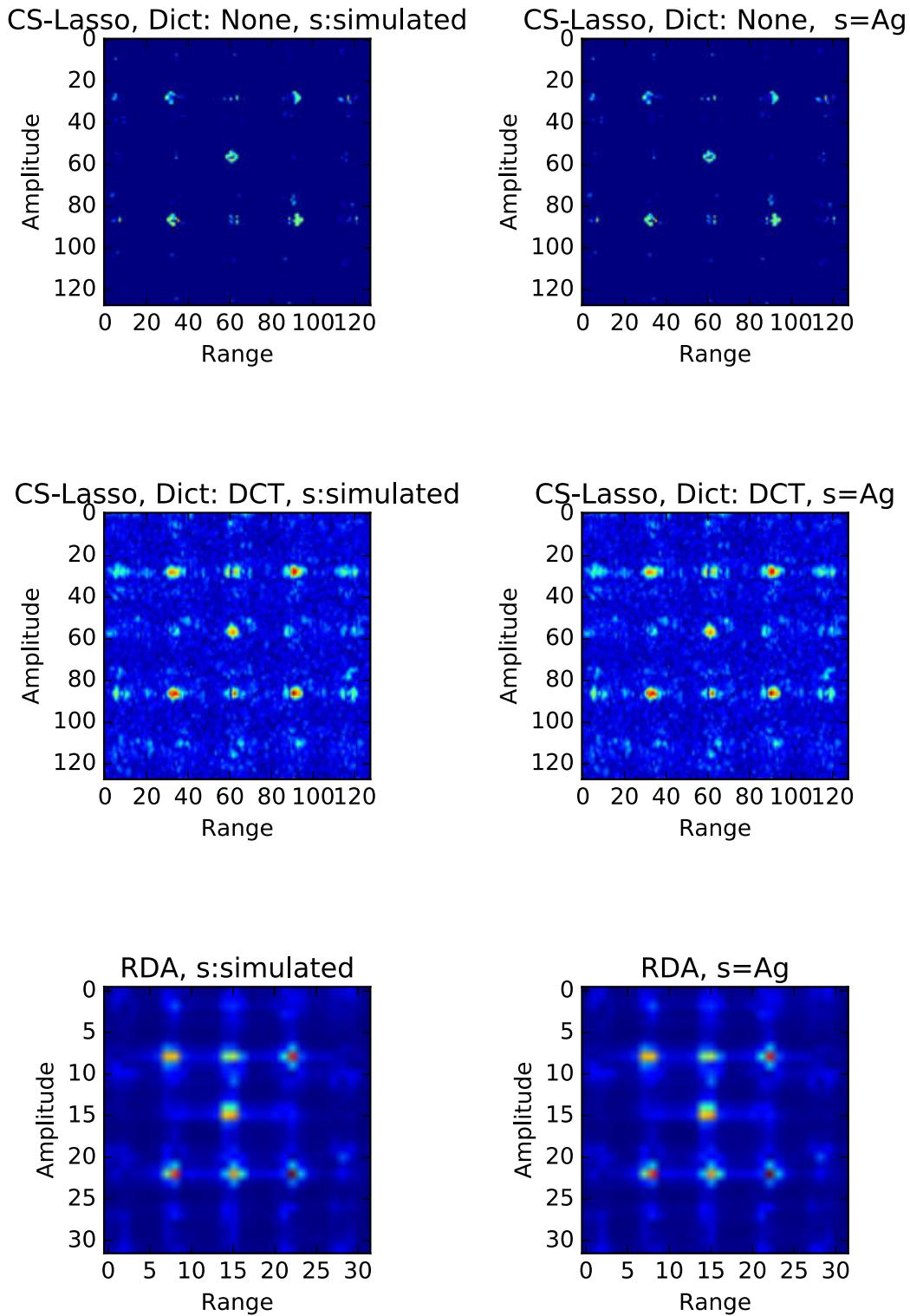


图 7.108: Compressive Sensing based and RDA Imaging Results of points (Lasso).

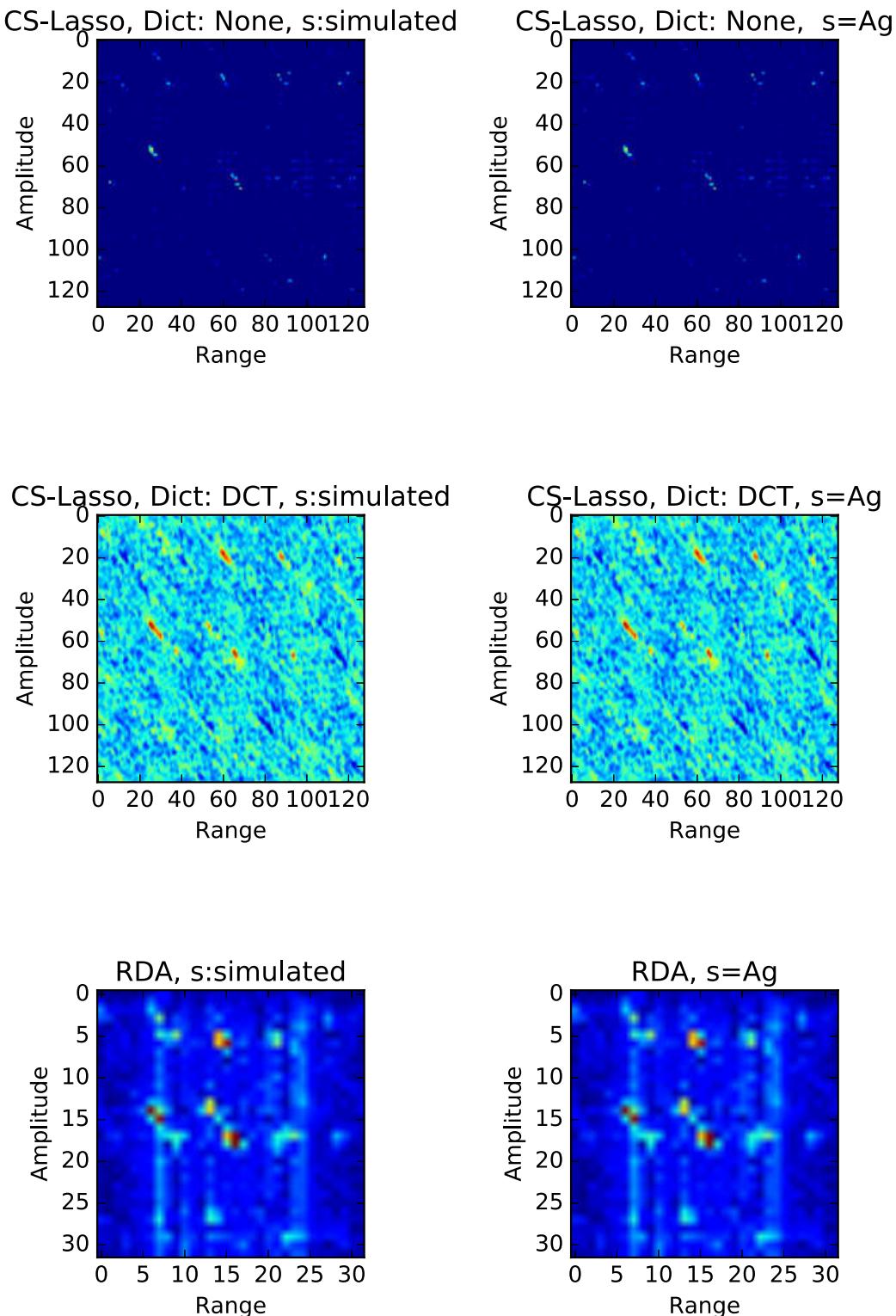


图 7.109: Compressive Sensing based and RDA Imaging Results of ship (Lasso).

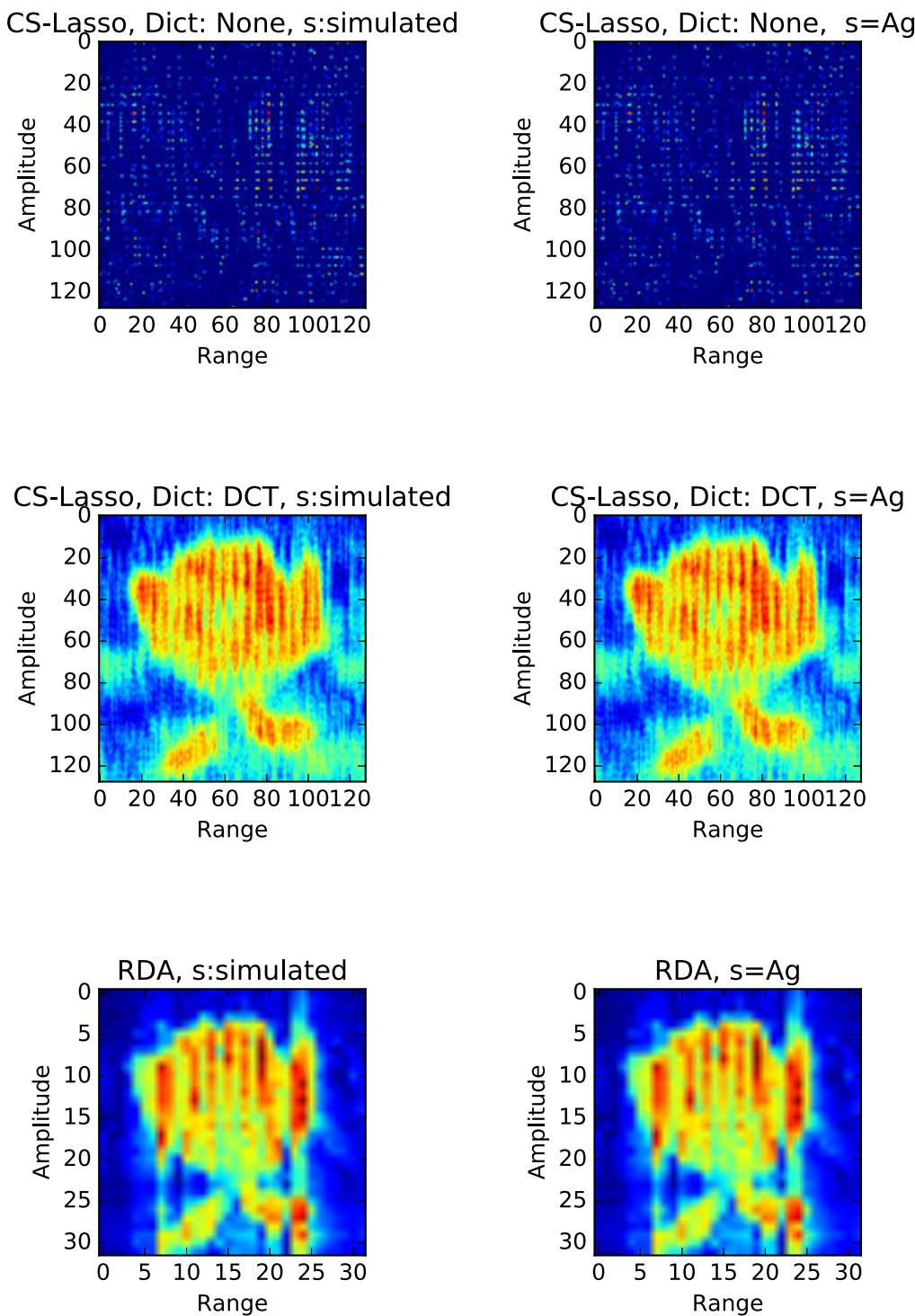


图 7.110: Compressive Sensing based and RDA Imaging Results of lotus (Lasso).

-
1. CS
 2. CS
 3. CS
-

7.5.6.2.2 原理与方法

7.5.6.2.2.1 学习式压缩感知框架

在压缩感知中, 我们需要设计观测矩阵和字典

7.5.6.2.2.2 有限等距性约束

7.5.6.2.3 学习式压缩成像

$$\min_{\mathcal{G}} \|\mathbf{y} - A\mathcal{G}(\mathbf{y})\|_2^2 + (\|A\mathcal{G}(\mathbf{y})\| - \|\mathcal{G}(\mathbf{y})\|)^2 + \lambda \|\mathcal{G}(\mathbf{y})\|_2$$

7.5.6.2.3.1 有限等距性约束

7.5.6.2.4 实验与分析

7.5.6.3 基于 IQGAN 的生成式超分辨 SAR 成像

7.5.6.3.1 动机与贡献

注解: 动机:

1. CS
 2. CS
 3. CS
-

注解: 贡献:

1. CS
 2. CS
 3. CS
-

7.5.6.3.2 原理与方法

7.5.6.3.2.1 模型框架

7.5.6.3.2.2 相位保持

7.5.6.3.2.3 超分辨成像

7.5.6.3.3 实验与分析

7.5.6.4 学习式压缩感知成像

7.5.6.4.1 动机与贡献

7.5.6.4.1.1 SAR 数据特性

由SAR回波信号及其性质(页450)小节可知,雷达接收采样后的IQ双通道原始信号可以表示为

$$\begin{aligned} s_I(\eta, \tau) &= r(\eta, \tau) \cos[\phi(\eta, \tau)] \\ s_Q(\eta, \tau) &= r(\eta, \tau) \sin[\phi(\eta, \tau)] \\ s(\eta, \tau) &= r(\eta, \tau) \exp[j\phi(\eta, \tau)] \\ &= s_I(\eta, \tau) + s_Q(\eta, \tau) \end{aligned} \quad (7.26)$$

其中,相位 $\phi(\eta, \tau)$ 为

$$\phi(\eta, \tau) = -\frac{4\pi f_c R(\eta)}{c} + \pi K_r \left(\tau - \frac{2R(\eta)}{c} \right)^2,$$

地面反射率与天线增益合成项 $r(\eta, \tau)$ 为

$$r(\eta, \tau) = g(\eta, \tau) w_a(\eta - \eta_c) w_r(\tau - 2R(\eta)/c).$$

由式7.26知

7.5.7 SAR 图像超分辨

7.5.8 实验集锦

7.5.8.1 基础分析实验

7.5.8.1.1 实验说明

通过点目标仿真,分析理解调频率,分辨率等概念.

Python实现代码,参见文件 `demo_AirboneSAR_Points.py` (https://github.com/antsfamily/iprs3.0/tree/master/examples/Air/demo_AirboneSAR_Points.py)

7.5.8.1.1.1 实验参数

实验参数如 表 7.1 所示, 类似文献 [1] 中 138 页的点目标仿真实验中的参数, 部分参数不一致.

表 7.1: 仿真的机载 SAR 平台参数

参数	符号	数值	单位
平台高度	H	10	km
雷达有效速度	V	150	m/s
场景中心斜距	R_c	20	km
雷达载频	f_0	5.3	GHz
发射脉冲时宽	T_p	2.5	s
距离向调频率	K_r	-20.0e12, +20.0e12	Hz/s
距离向采样率	F_r	60.0	MHz
距离向采样点数	N_r	320	
方位向采样率	F_a	100	Hz
方位向采样点数	N_a	256	
距离向天线孔径	L_r	12.0	m
方位向天线孔径	L_a	3.0	m
下视角	θ_d	30	$^\circ$
波束距离向宽度	θ_b	1.32, 1.32, 1.32	$^\circ$
波束斜视角	θ_s	0.0, 3.5, 21.9	$^\circ$
波束中心偏移时间	η_c	0.0, -8.1, -49.7	s
多普勒中心频率	f_{η_c}	0.0, 320, 1975	Hz

根据上表参数, 可以计算出如下参数

- 场景中心斜距: $R_c = H/\sin\theta_d \approx 20000.0m$
- 最短斜距: $R_0 = R_c \cos\theta_s \approx 18556.7m$
- 斜距分辨率: $\Delta_r = \frac{c}{2B_r} = \frac{c}{2|K_r|T_p} \approx 2.99m$
- 地距分辨率: $\Delta_x = \Delta_r / \cos\theta_i \approx 3.46m$
- 方位向距离分辨率: $\Delta_a = \Delta_y = L_a/2 = 1.5m$
- 场景中心坐标: (x_c, y_c) , $x_c = \sqrt{R_0^2 - H^2}$, $y_c = R_c \sin\theta_s$
 - $\theta_s = 0, x_c = 17320.50m, y_c = 0m$
 - $\theta_s = 3.5, x_c = 17286.62m, y_c = 1221.45m$
 - $\theta_s = 21.9, x_c = 15640.55m, y_c = 7462.73m$
- 近地点斜距: $R_{near} = H/\sin(\theta_d + \theta_b/2)$
 - $\theta_s = 0, R_{near} = 19609.59m$
 - $\theta_s = 3.5, R_{near} = 19610.05m$
 - $\theta_s = 21.9, R_{near} = 19609.59m$
- 远地点斜距: $R_{far} = H/\sin(\theta_d - \theta_b/2)$

- $\theta_s = 0, R_{far} = 20409.03m$
- $\theta_s = 3.5, R_{far} = 20409.03m$
- $\theta_s = 21.9, R_{far} = 20409.03m$
- **刈幅宽度 (swath size):** $S_x = (H/\tan(\theta_d - \theta_b/2) - H/\tan(\theta_d + \theta_b/2))\cos\theta_s = 921.9\cos\theta_s$
 - $\theta_s = 0, S_x = 923.05m$
 - $\theta_s = 3.5, S_x = 921.91m$
 - $\theta_s = 21.9, S_x = 880.48m$
- 方位向成像宽度: $S_y = VT_{sa} = 384.0m$
- **成像区域:** $(x_{min}, x_{max}, y_{min}, y_{max})$
 - $\theta_s = 0, SA = (-461.53, 461.53, -192.0, 192.0)$
 - $\theta_s = 3.5, SA = (-460.96, 460.95, -192.0, 192.0)$
 - $\theta_s = 21.9, SA = (-440.58, 439.91, -192.0, 192.0)$

7.5.8.1.2 调频方向分析

设置目标位于场景中心处, 即相对坐标 $(0,0)$, 目标强度为 1, 观察不同调频方向, 不同斜视角下的时域(原始信号), 距离多普勒域(原始信号在方位维做 FFT), 频域(原始信号在距离和方位上均做 FFT) 的幅度相位.

零斜视角, $\theta_s = 0.0^\circ$, 正调频

零斜视角, $\theta_s = 0.0^\circ$, 负调频

斜视角 $\theta_s = 3.5^\circ$, 正调频

斜视角 $\theta_s = 3.5^\circ$, 负调频

斜视角 $\theta_s = 21.9^\circ$, 正调频

斜视角 $\theta_s = 21.9^\circ$, 负调频

7.5.8.1.3 分辨率分析

设置目标为 $(0.0, 0.0), (-6, -6), (-6, 6), (6, -6), (6, 6), (9, 9)$, 目标强度均为 1, 进行如下设置.

- 方位向设置

- 采样率: $F_{sa} = 100Hz$ 和 $F_{sa} = 200Hz$
- 采样点: $N_{sa} = 256$ 和 $N_{sa} = 1024$
- 天线孔径: $L_a = 6.0m, L_a = 3.0m$ 和 $L_a = 1.5m$

- 距离向设置

- 采样率: $F_{sr} = 60.0e6Hz$
- 采样点: $N_{sr} = 320$ 和 $N_{sr} = 1280$

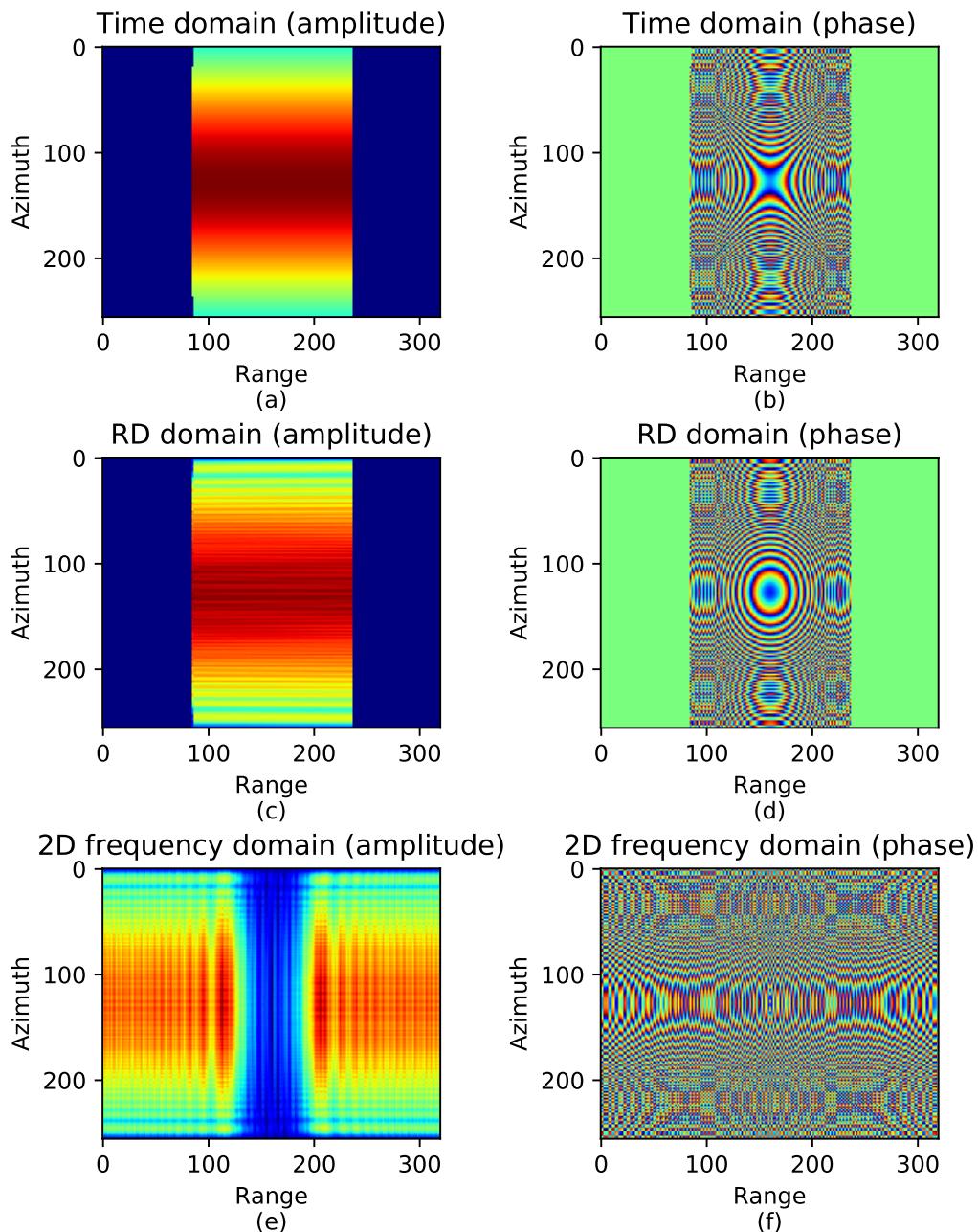


图 7.111: The property of point target in different domain (positive frequency modulation).

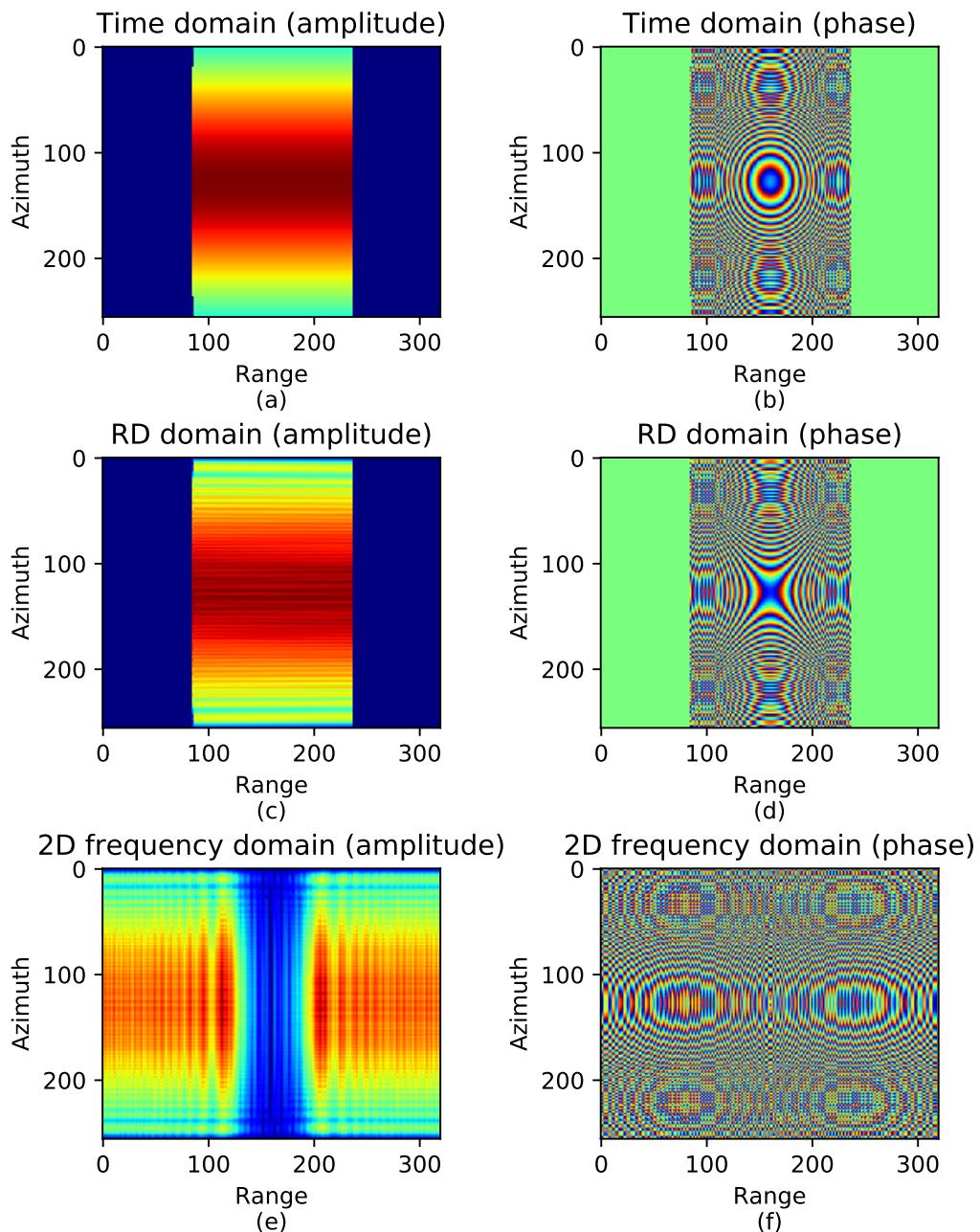


图 7.112: The property of point target in different domain (negative frequency modulation).

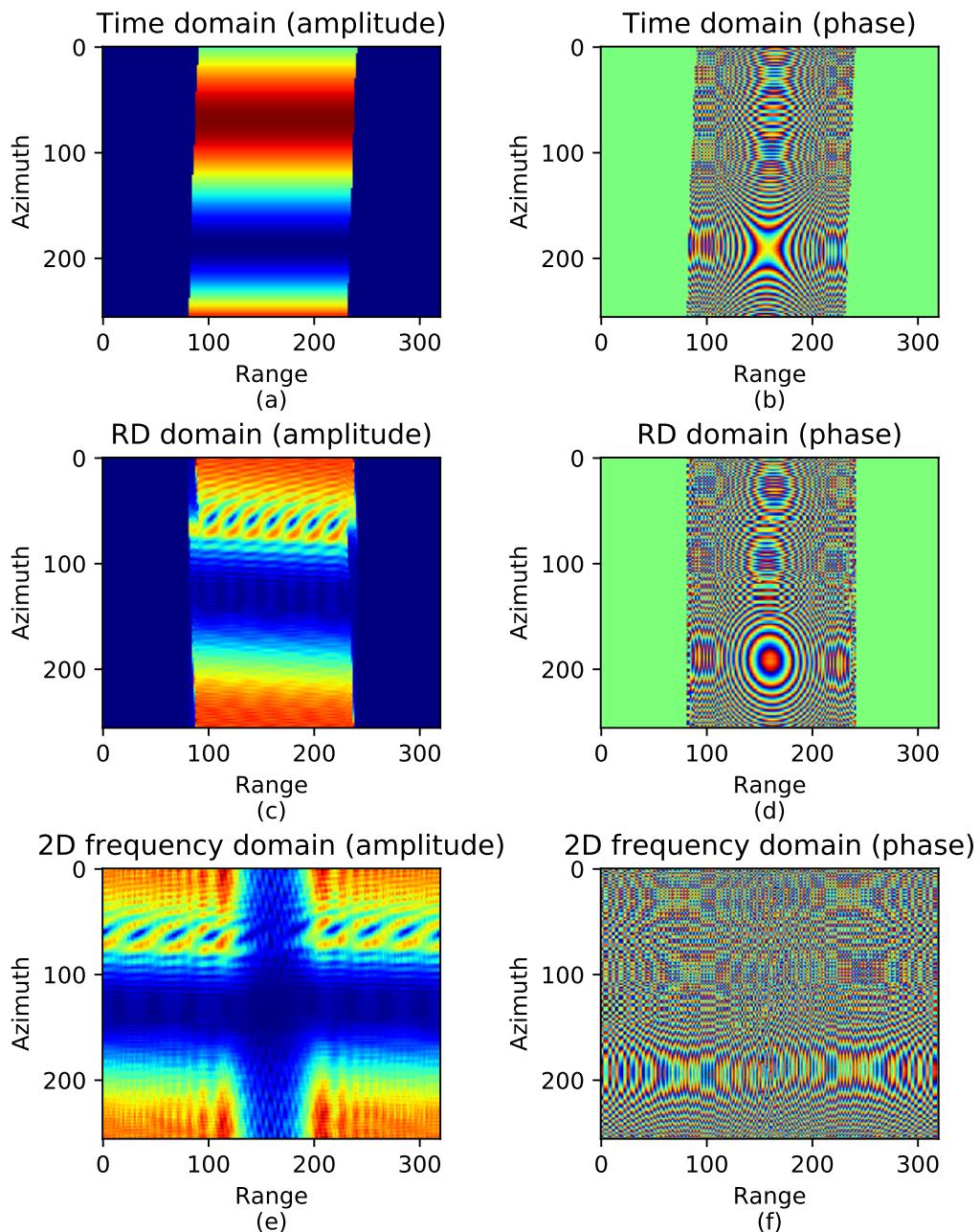


图 7.113: The property of point target in different domain (positive frequency modulation).

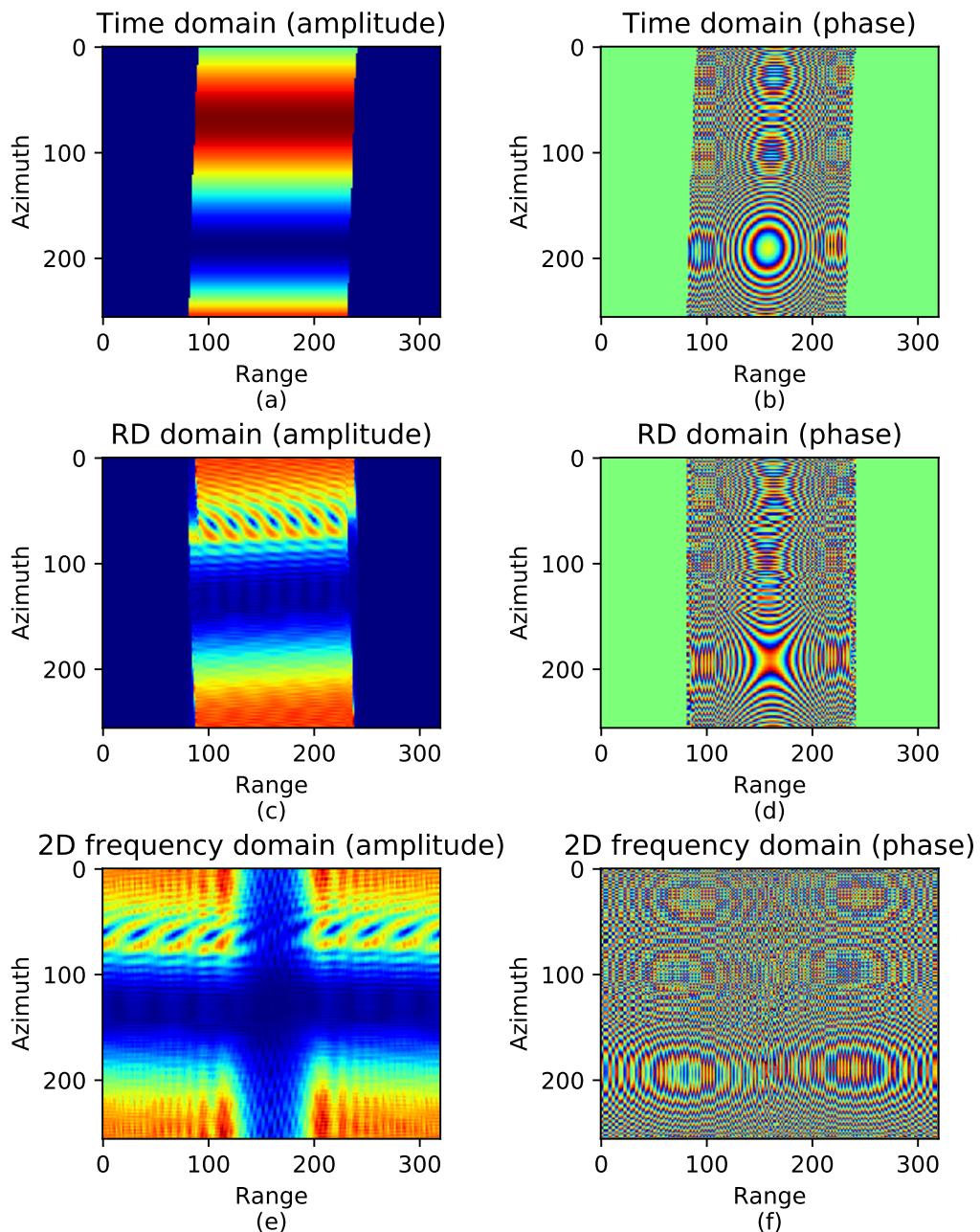


图 7.114: The property of point target in different domain (negative frequency modulation).

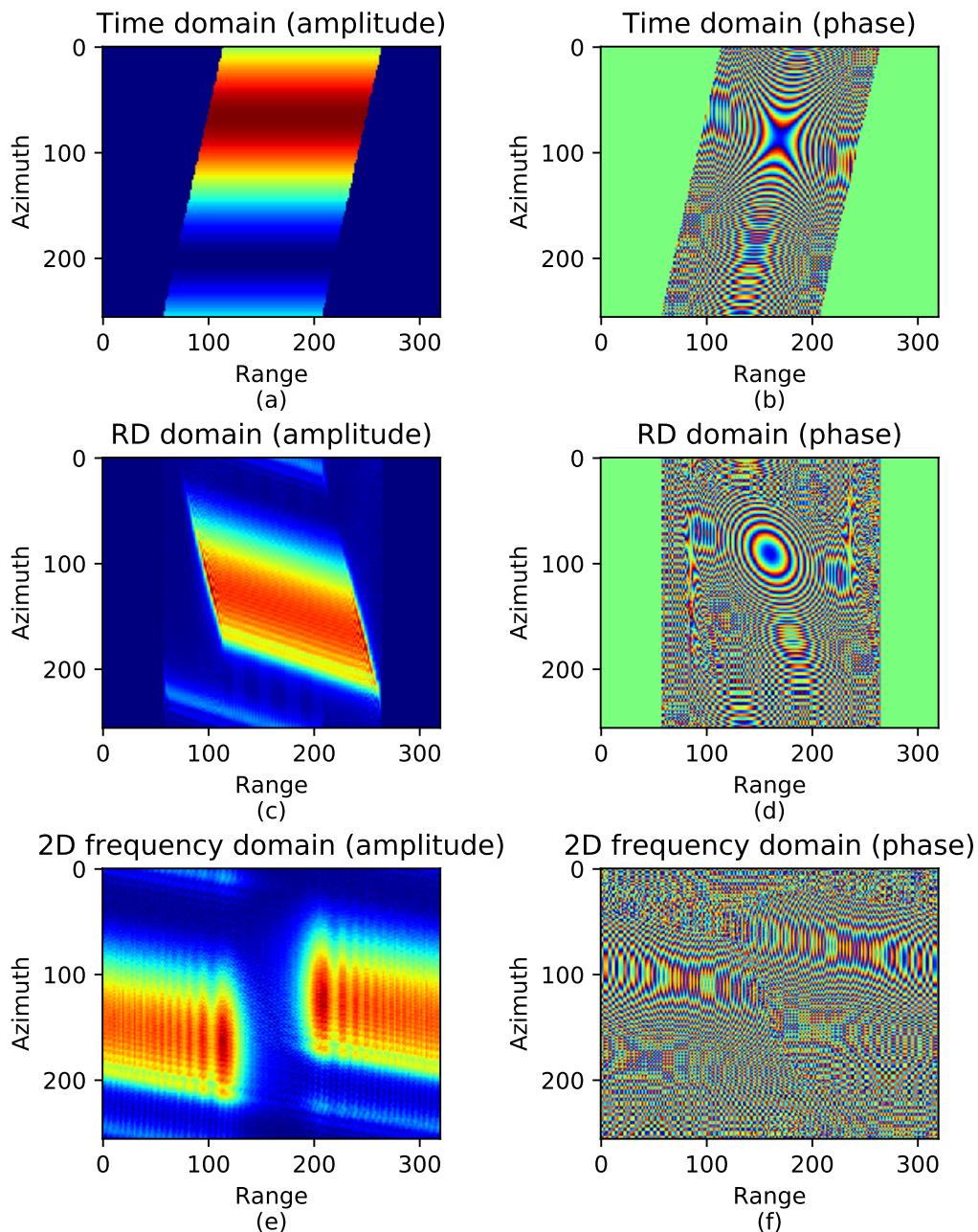


图 7.115: The property of point target in different domain (positive frequency modulation).

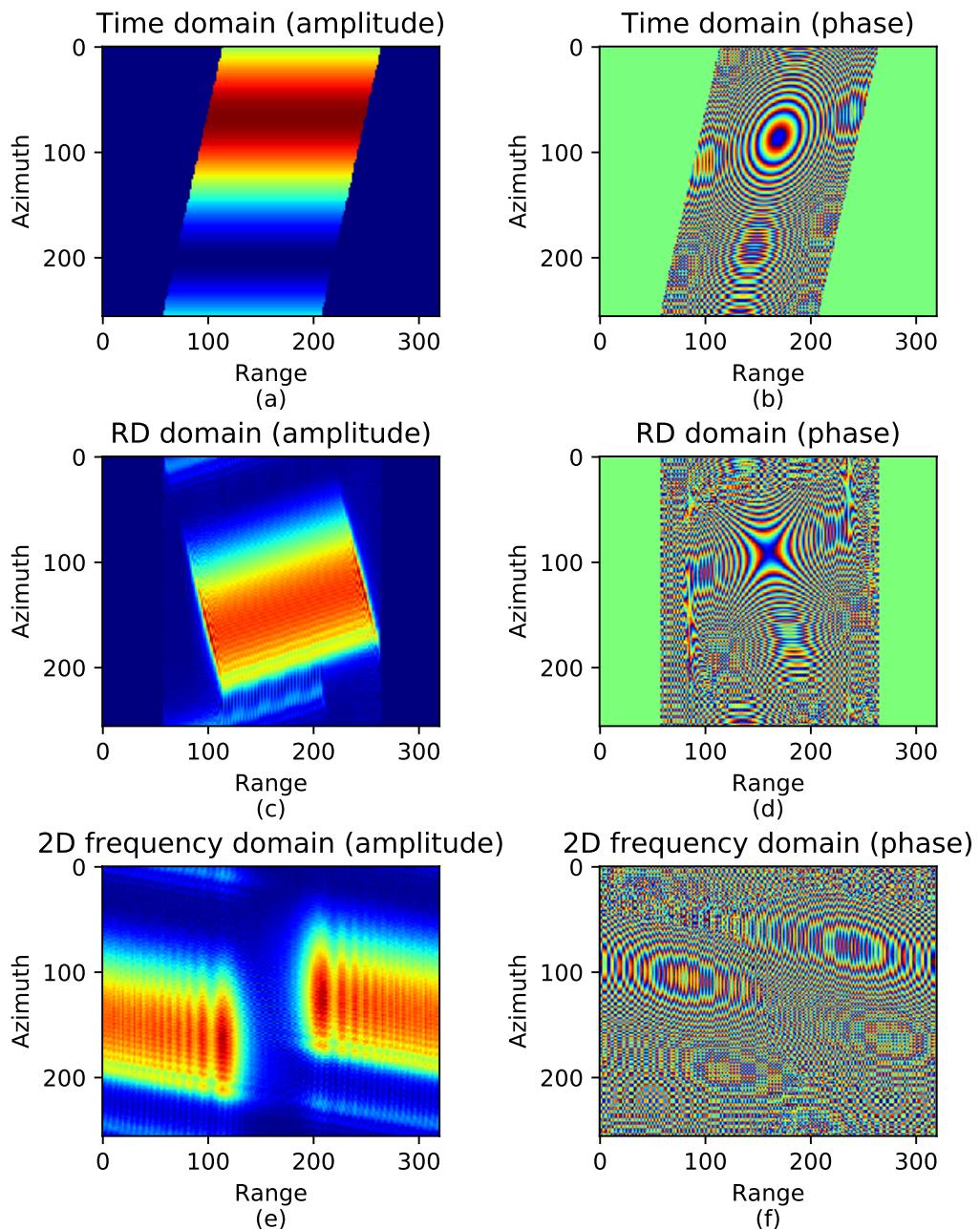


图 7.116: The property of point target in different domain (negative frequency modulation).

- 调频率: $K_r = -20.0e12 Hz/s$
- 脉宽: $T_p = 2.5e-6 s$ 和 $T_p = 5.0e-6 s$

方位向的分辨率由方位向天线孔径长度与方位向采样率决定, 其中, L_a 直接决定方位向分辨率上限 R_a , F_{sa} 决定了能多大程度地恢复方位向分辨率为 R_a 场景数据. 即使是过采样, 恢复后的最大分辨率仍然为 R_a , 但若是欠采样, 恢复后的分辨率将小于 R_a .

距离向的分辨率由距离向带宽和采样率决定, 其中, $Br = |K_r|T_p$ 决定了距离向分辨率上限 R_r , F_{sr} 决定了能多大程度地恢复方位向分辨率为 R_a 场景数据. 即使是过采样, 恢复后的最大分辨率仍然为 R_r , 但若是欠采样, 恢复后的分辨率将小于 R_r .

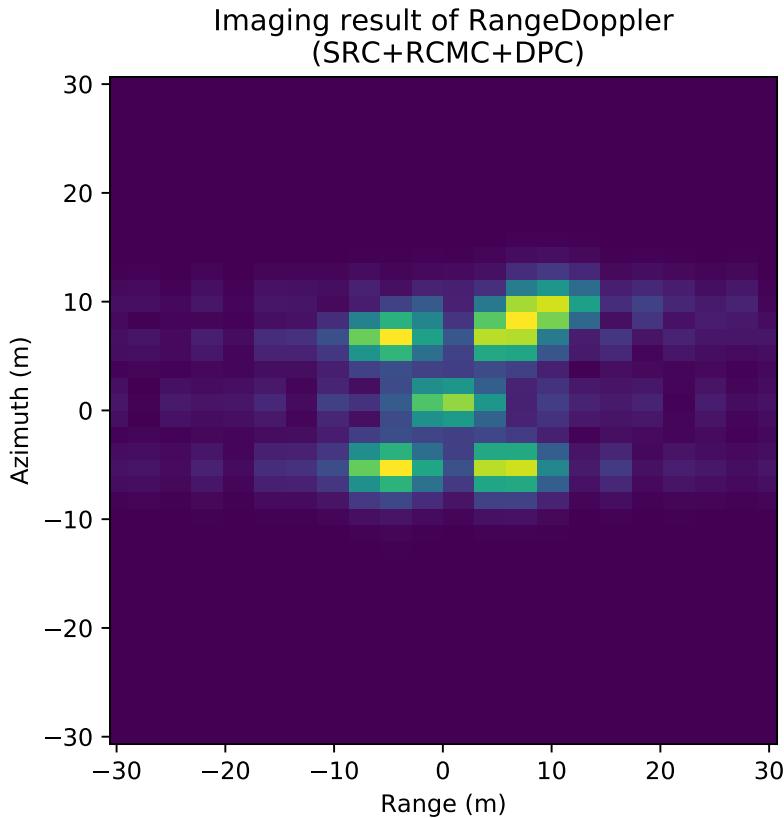


图 7.117: Imaging result of RDA.

Imaging result of RDA. $F_{sa} = 100 Hz$, $N_{sa} = 256$, $L_a = 6 m$, $F_{sr} = 60 e6 Hz$, $N_{sr} = 320$, $K_r = 10 e12 Hz/s$, $T_p = 2.5 e-6 s$

7.5.8.1.4 距离徙动校正

设置斜视角 $\theta_s = 8.5^\circ$, 调频率 $K_r = 20e12$, $L_a = 3.0$, $L_r = 12$, 方位向采样率 $F_{sa} = 100Hz$, 距离向采样率 $F_{sr} = 60MHz$, 目标位置 $(0, 0)$, 目标强度均为 1. 分析距离徙动校正的效果, 分析不同大小的 sinc 插值核 $r = 4, 8, 32$ 的影响.

图 7.120 显示的是二次距离压缩及多普勒相位补偿后的结果, 即为进行距离徙动校正, 图 7.121, 图 7.122, 图 7.123 为 sinc 插值核大小分别为 $r = 4, 8, 32$ 的时的距离徙动校正后的结果. 可见, 经过插值核越大, 距离徙动校正越精确.

图 7.124 给出了无距离徙动校正条件下, 距离多普勒成像算法的成像结果, 由图可见, 距离徙动现象很明显.

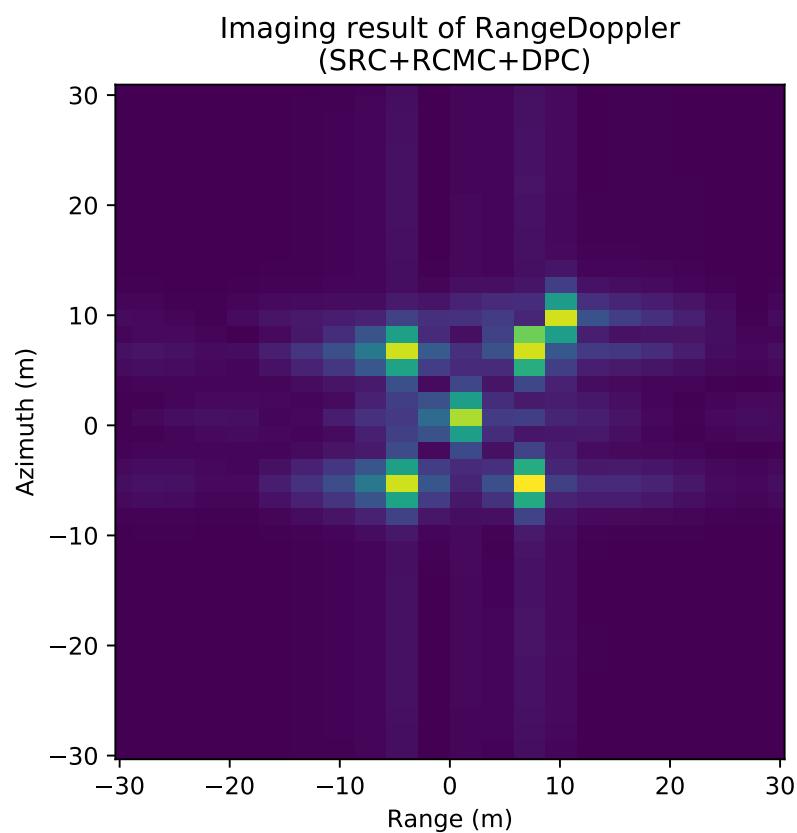


图 7.118: Imaging result of RDA.

Imaging result of RDA. $F_{sa} = 100Hz$, $N_{sa} = 256$, $L_a = 3m$, $F_{sr} = 60e6Hz$, $N_{sr} = 320$, $K_r = 20e12Hz/s$,
 $T_p = 2.5e-6s$

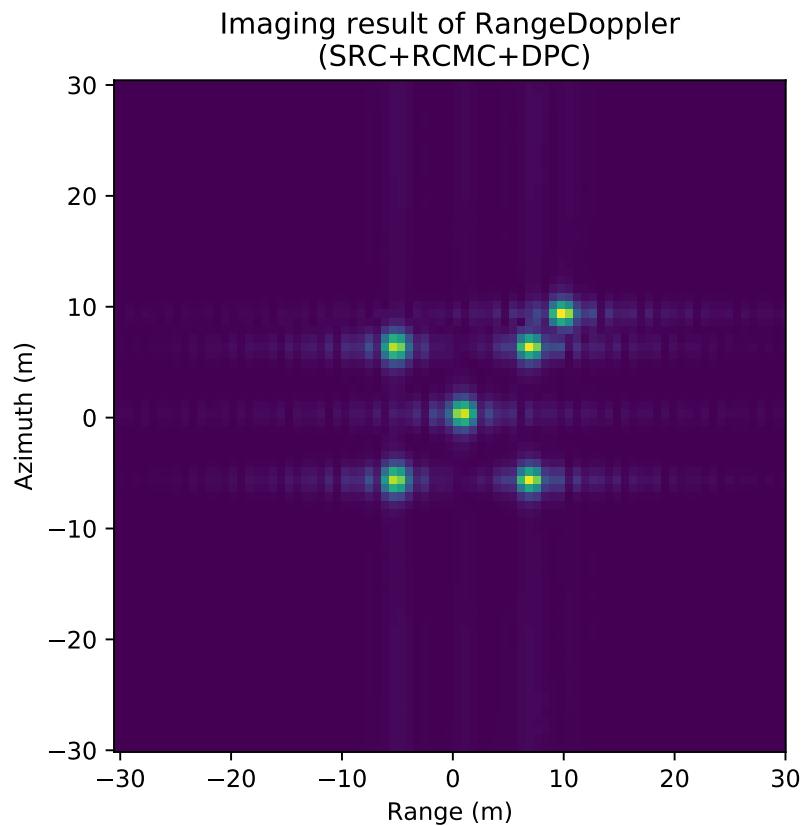


图 7.119: Imaging result of RDA.

Imaging result of RDA. $F_{sa} = 200\text{Hz}$, $N_{sa} = 1024$, $L_a = 1.5\text{m}$, $F_{sr} = 240e6\text{Hz}$, $N_{sr} = 1280$, $K_r = 40e12\text{Hz/s}$,
 $T_p = 25e-6\text{s}$

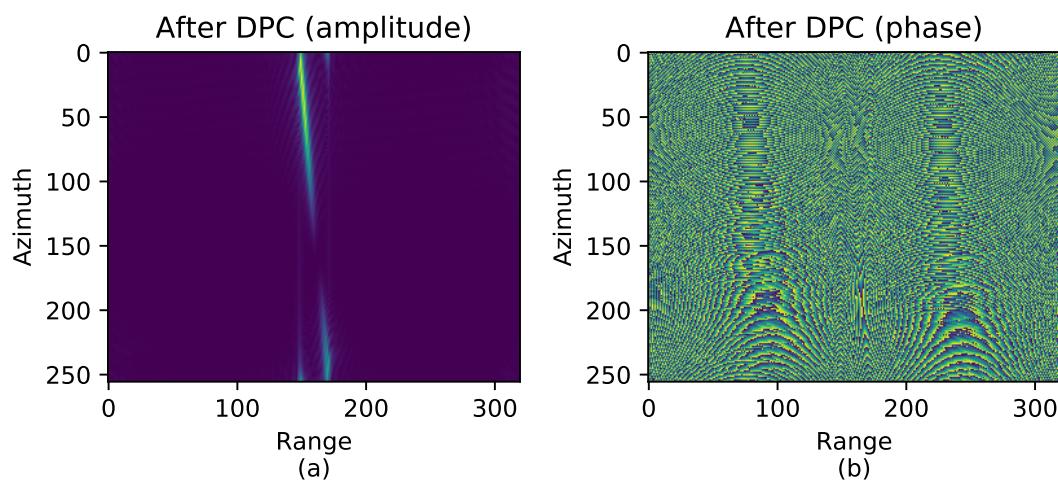


图 7.120: After RC SRC, DPC

After RC SRC, DPC

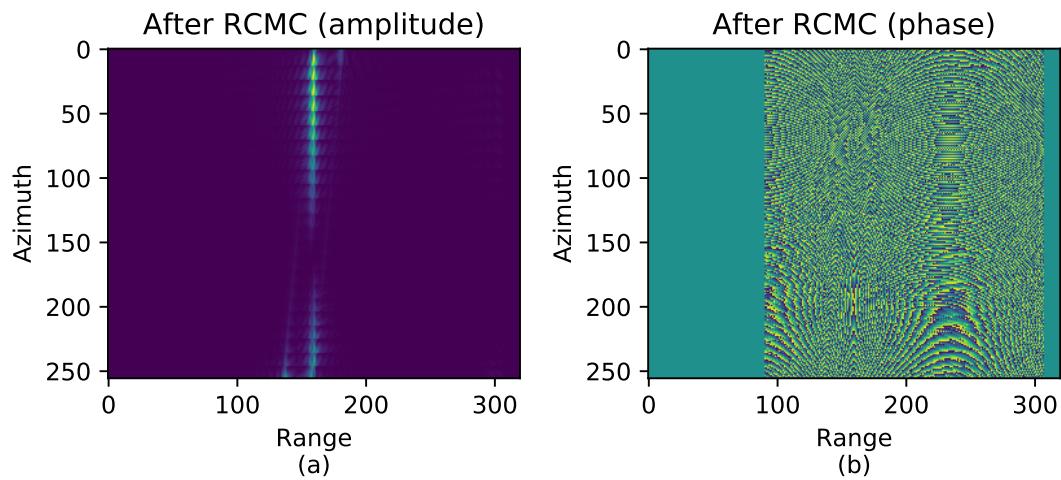


图 7.121: After RC SRC, DPC, RCMC. sinc interpolation, $r = 4$.

After RC SRC, DPC, RCMC. sinc interpolation, $r = 4$.

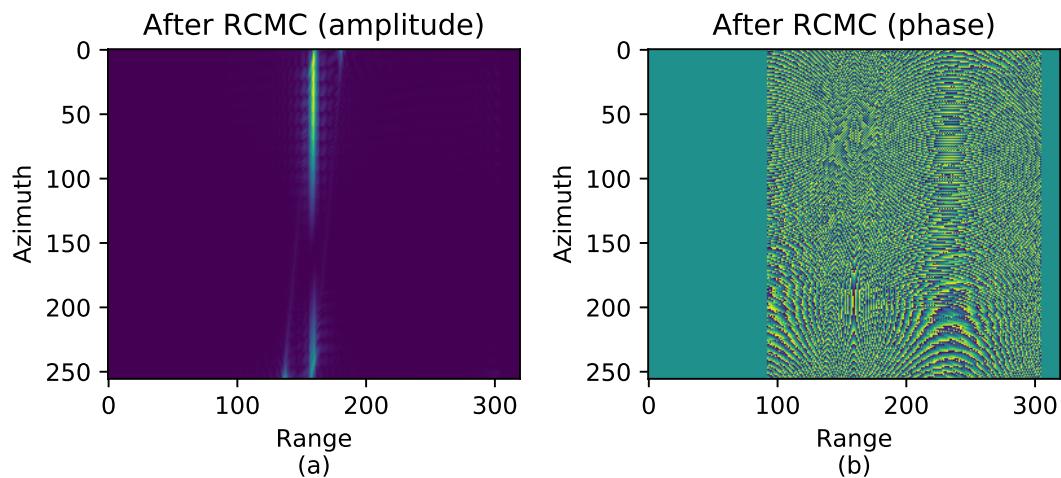
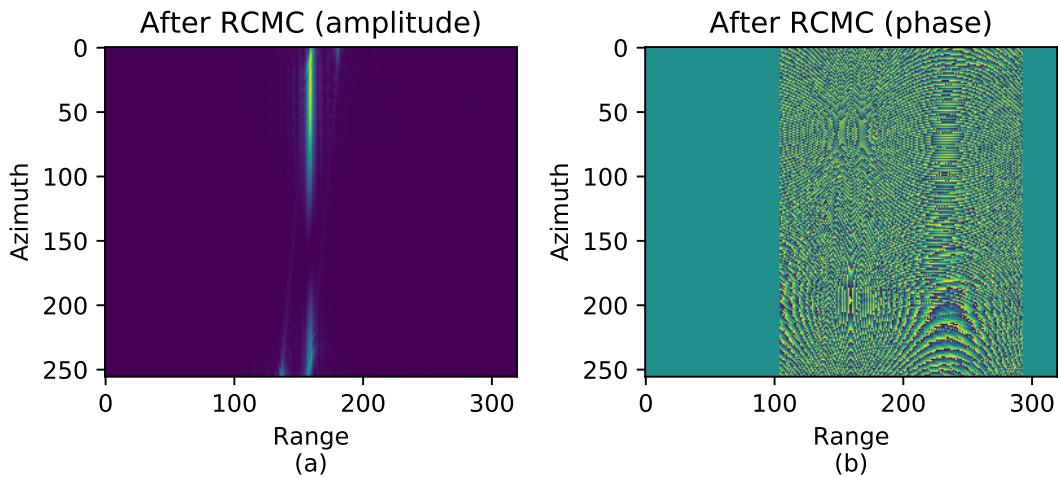


图 7.122: After RC SRC, DPC, RCMC. sinc interpolation, $r = 8$.

After RC SRC, DPC, RCMC. sinc interpolation, $r = 8$.

图 7.123: After RC SRC, DPC, RCMC. sinc interpolation, $r = 32$.After RC SRC, DPC, RCMC. sinc interpolation, $r = 32$.

同一目标分布在不同距离与方位单元. 图 7.125, 图 7.126, 图 7.127 为 sinc 插值核分别为 4, 8, 32 的条件下, 距离多普勒算法的成像结果, 对比 图 7.124 - 图 7.127, 可以发现, 基于 sinc 插值的距离徙动算法校正效果明显, 且插值核越大, 距离徙动校正越精确.

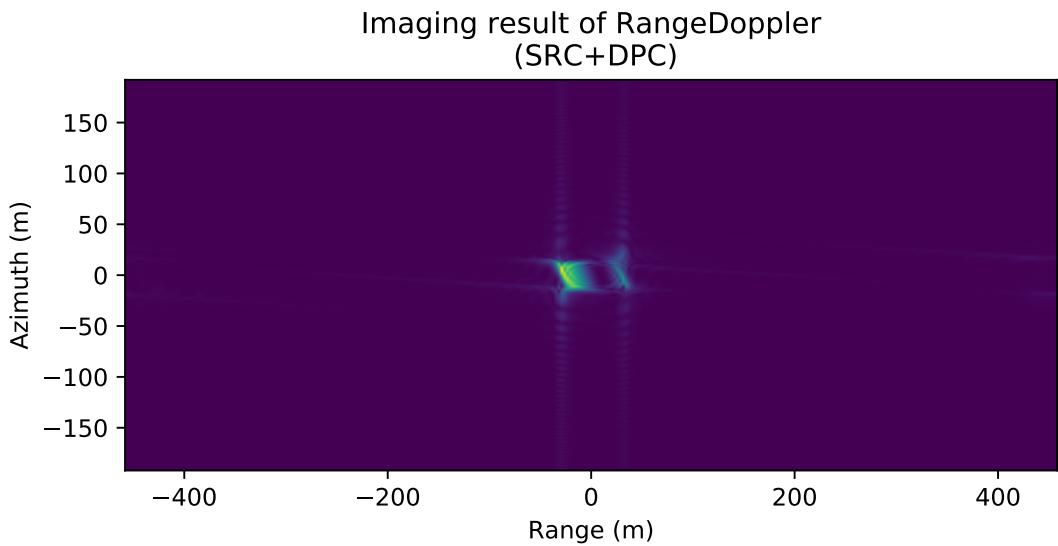


图 7.124: Imaging result of RDA.

Imaging result of RDA. Without RCMC.

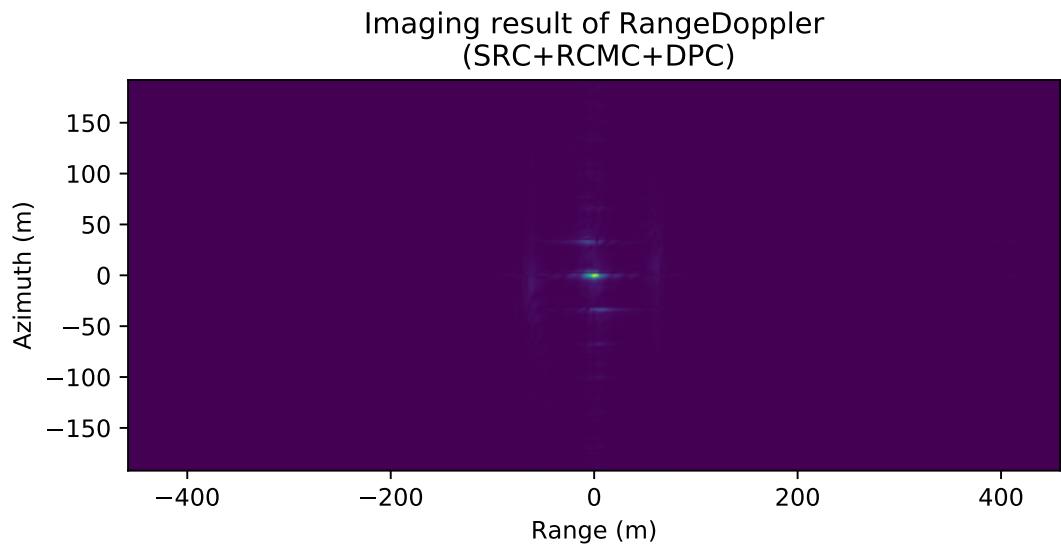


图 7.125: Imaging result of RDA. sinc interpolation ($r = 4$).

Imaging result of RDA. sinc interpolation ($r = 4$).

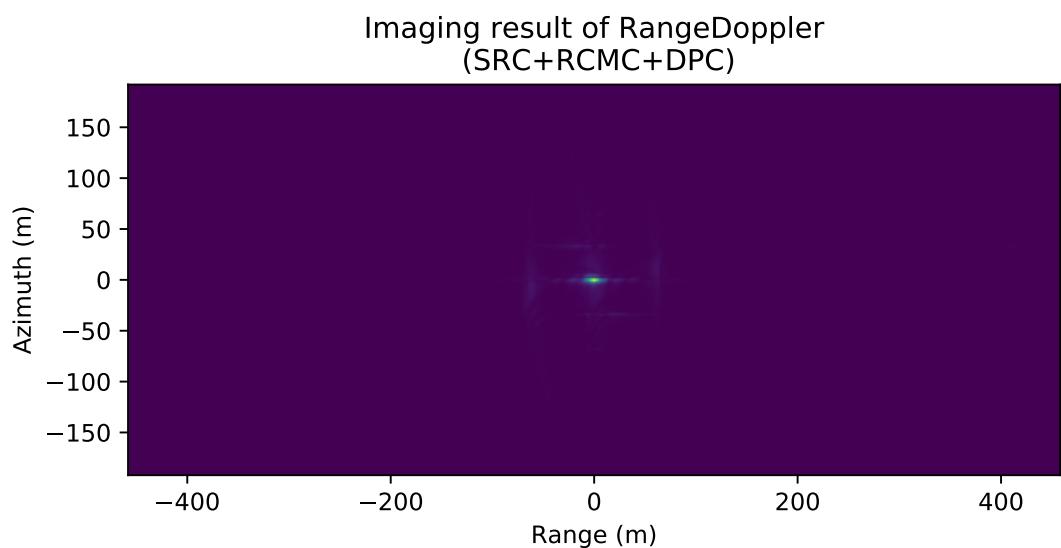


图 7.126: Imaging result of RDA. sinc interpolation ($r = 8$).

Imaging result of RDA. sinc interpolation ($r = 8$).

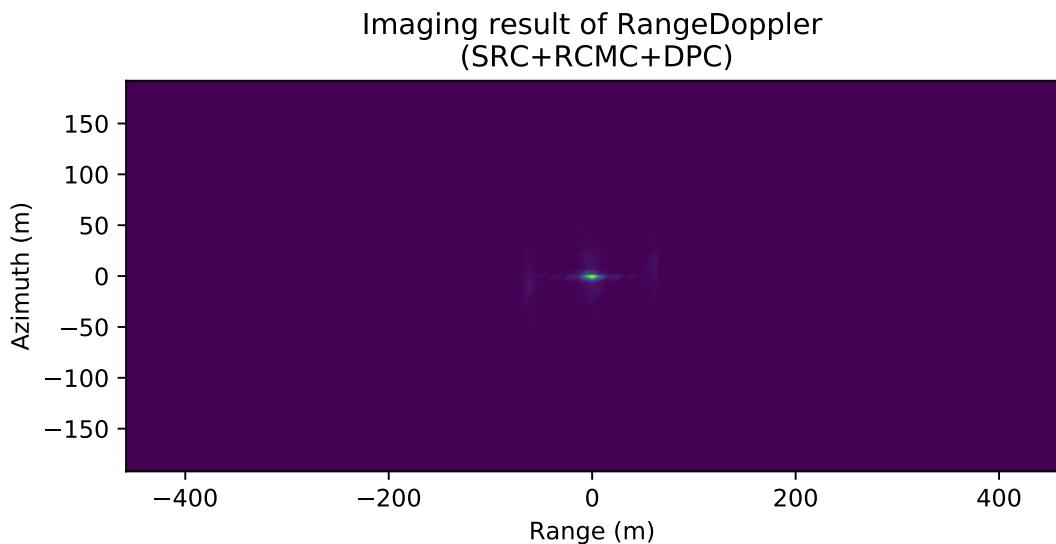


图 7.127: Imaging result of RDA. sinc interpolation ($r = 32$).

Imaging result of RDA. sinc interpolation ($r = 32$).

7.5.8.2 子区域成像实验

7.5.8.2.1 实验说明

从原始较大场景数据中, 选择不同大小的场景区域数据进行成像, 成像方法为调频变标算法 (CSA). 一种方法是将选取子区域外的数据置零后成像, 另一种是仅用选取的子区域大小数据成像.

7.5.8.2.2 仿真数据实验

7.5.8.2.2.1 点目标数据实验

7.5.8.2.2.2 实验代码

7.5.8.2.2.3 实验结果

7.5.8.2.3 真实数据实验

7.5.8.2.3.1 RADARSAT1 数据实验

实验中所用数据为 RADARSAT1 卫星获得的数据, 成像区域为史丹利公园 (Stanley Park), 有关 RADARSAT1 的参数信息, 可参考 [RADARSAT 产品介绍](#) (页 546) 小节.

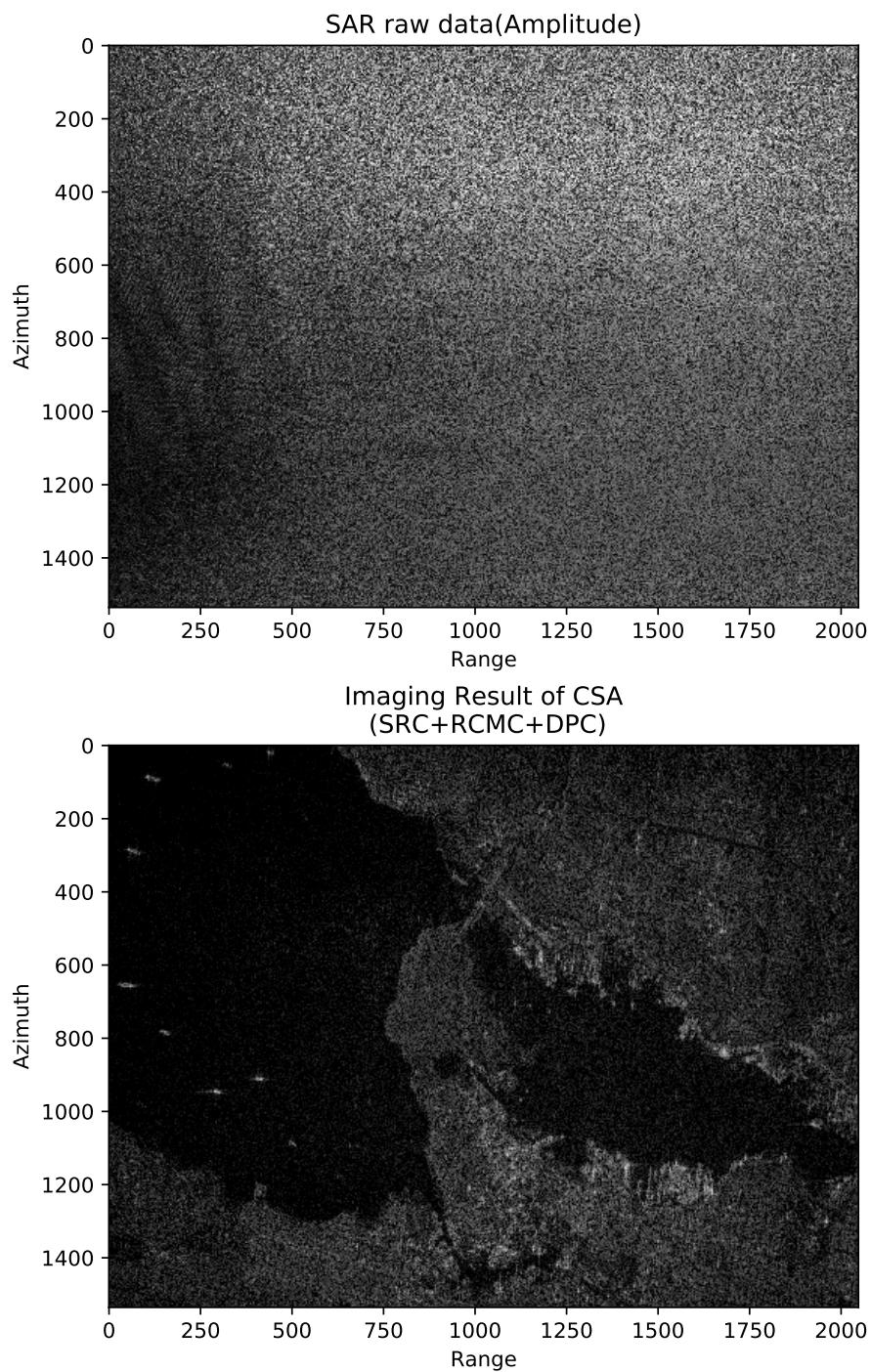


图 7.128: Stanley Park 区域原始数据幅度与成像结果

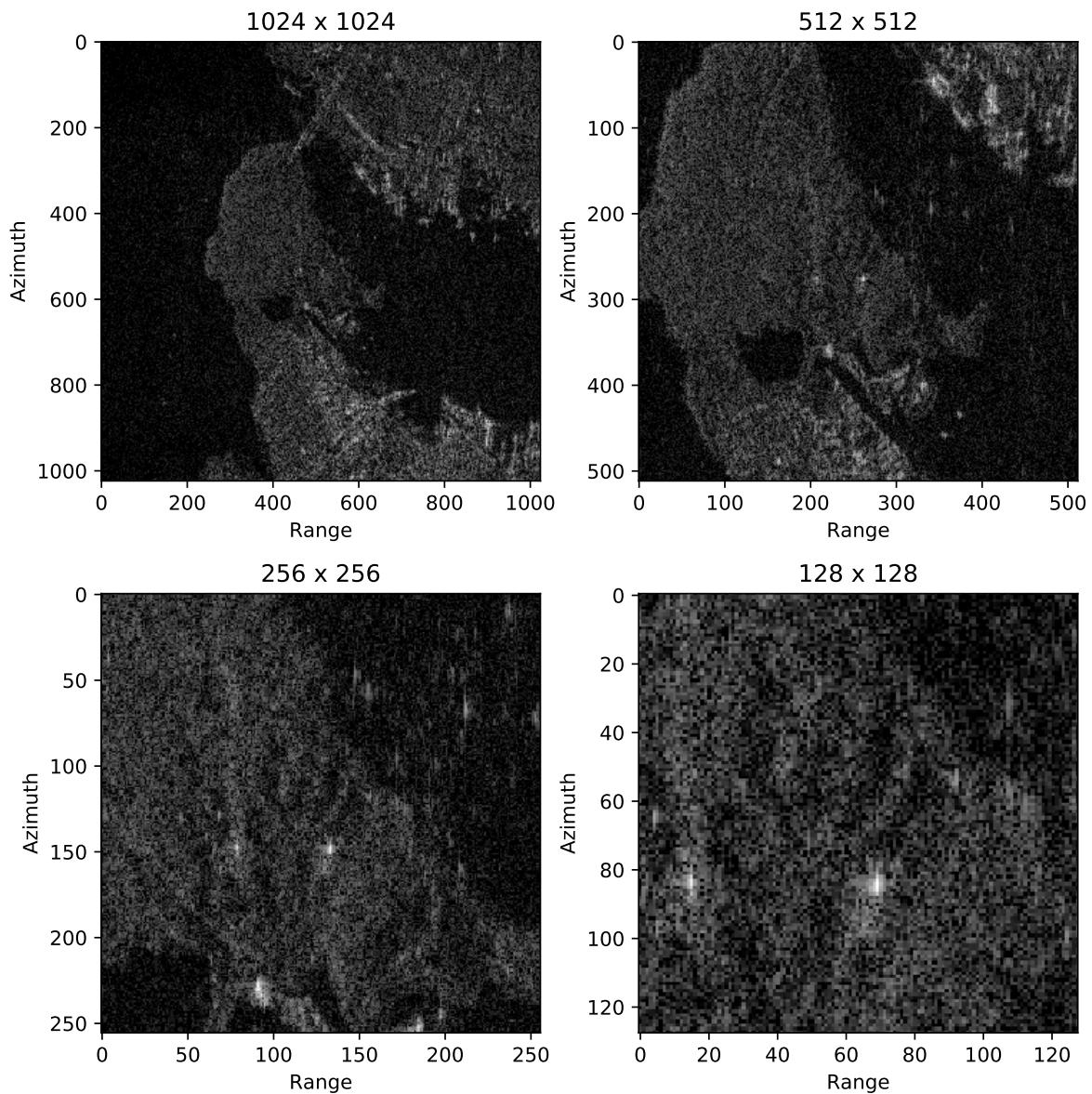


图 7.129: 对成像后的 Stanley Park 地区影像, 选取场景中心附近不同大小的区域可视化结果.

7.5.8.2.3.2 实验代码

代码 7.1: demo_RADARSAT1_SubRegion.py

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # @Date    : 2019-02-18 10:14:12
4  # @Author  : Yan Liu & Zhi Liu (zhiliu.mind@gmail.com)
5  # @Link    : http://iridescent.ink
6  # @Version : $1.0$
```

```

7
8 import iprs
9 import numpy as np
10 import scipy.io as scio
11 import matplotlib.pyplot as plt
12
13 imagingMethod = 'RDA'
14 # imagingMethod = 'OmegaK'
15 imagingMethod = 'CSA'
16
17 zpadar = (512, 512)
18 # zpadar = False
19 # zpadar = None
20 usesrc = True
21 # usesrc = False
22 usedpc = True
23 # usedpc = False
24 rcmc = False
25 rcmc = 32
26
27 fulltime = False
28 # fulltime = True
29
30 usemask = False
31 # usemask = True
32
33 sensor_name = 'RADARSAT1'
34 acquis_name = 'RADARSAT1'
35
36 sarplat = iprs.SarPlat()
37 sarplat.name = "sensor=" + sensor_name + "_acquisition=" + acquis_name
38 sarplat.sensor = iprs.SENSORS[sensor_name]
39 sarplat.acquisition = iprs.ACQUISITION[acquis_name]
40 sarplat.params = None
41
42
43 ROI = {
44     'SubSceneArea': None, # SceneArea

```

(下页继续)

(续上页)

```

45     # 'SubSceneArea': [0.5, 0.5, 0.5, 0.5],  # SceneArea/2.0
46     # 'SubEchoSize': None,    # EchoSize
47     'SubEchoSize': [1., 1.],
48     # 'SubEchoSize': [1. / 3, 1. / 4],
49     # 'SubEchoSize': [1. / 6, 1. / 8],
50     'SubEchoSize': [1. / 12, 1. / 16],
51 }
52
53 sarplat.selection = ROI
54 ROIY = sarplat.selection['SubEchoSize'][0]
55 ROIX = sarplat.selection['SubEchoSize'][1]
56
57 # sarplat.selection = None
58 NstartX = int(sarplat.acquisition['EchoSize'][1] / 2. - ROIX / 2.)
59 NstartY = int(sarplat.acquisition['EchoSize'][0] / 2. - ROIY / 2.)
60
61
62 if fulltime:
63     mask = np.zeros(sarplat.acquisition['EchoSize'])
64     sarplat.selection = None
65
66 sarplat.select()
67 sarplat.printsp()
68
69 SA = sarplat.selection['SubSceneArea']
70
71 disk = '/mnt/d/'
72 # disk = 'D:/'
73 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
74 #             ↪sardataRADARSAT1scene01AzimuthBlock1.mat'
75 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
76 #             ↪sardataRADARSAT1scene01AzimuthBlock1_Squamish.mat'
77 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
78 #             ↪sardataRADARSAT1scene01AzimuthBlock1_VancouverAirport.mat'
79 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
80 #             ↪sardataRADARSAT1scene01AzimuthBlock1_Brackendale.mat'
81 datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
82 #             ↪sardataRADARSAT1scene01AzimuthBlock1_EnglishBayShips.mat'
83 datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
84 #             ↪sardataRADARSAT1scene01AzimuthBlock1_StanleyPark.mat'
85 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
86 #             ↪sardataRADARSAT1scene01AzimuthBlock1_UBC.mat'
87 # datafile = disk + 'DataSets/sar/RADARSAT/frombooks/mat/
88 #             ↪sardataRADARSAT1scene01AzimuthBlock2_River.mat'
89
90
91
92 data = scio.loadmat(datafile, struct_as_record=True)

```

(下页继续)

(续上页)

```

84
85 temp = data['sardata']
86 temp = temp['rawdata'][0][0]
87
88 NendY = NstartY + ROIY
89 NendX = NstartX + ROIX
90
91 if fulltime:
92     Sr = np.zeros_like(temp)
93     Sr[NstartY:NendY, NstartX:NendX] = temp[NstartY:NendY, NstartX:NendX]
94     mask[NstartY:NendY, NstartX:NendX] = 1
95 else:
96     Sr = temp[NstartY:NendY, NstartX:NendX]
97
98 print(NstartY, NstartX, NendY, NendX)
99
100 Na, Nr = Sr.shape
101 print("SAR raw data: ", Sr.shape, Sr.dtype)
102
103
104 if imagingMethod is 'RDA':
105     # SI, _, _ = iprs.rda(Sr, sarplat, verbose=True)
106     SI = iprs.rda_adv(Sr, sarplat, zpadar=zpadar,
107                         usesrc=usesrc, usedpc=usedpc, rcmc=rcmc, verbose=False)
108 if imagingMethod is 'CSA':
109     # SI = iprs.csa(Sr, sarplat, verbose=True)
110     SI = iprs.csa_adv(Sr, sarplat, zpadar=zpadar,
111                         usesrc=usesrc, rcmc=rcmc, usedpc=usedpc, verbose=False)
112
113 SI = np.abs(SI)
114
115 data = {'SI': SI}
116 scio.savemat('./SI.mat', data)
117
118 print(SI.max(), SI.min())
119
120 SI = SI / SI.max()
121 print(SI.max(), SI.min())
122 SI = 20 * np.log10(SI + iprs.EPS)
123 print(SI.max(), SI.min())
124 # a = np.abs(SI.mean())
125 a = 50
126
127 SI = (SI + a) / a * 255
128 print(SI.max(), SI.min())
129 # SI = exposure.adjust_log(SI + iprs.EPS)
130 # SI = SI * 255

```

(下页继续)

(续上页)

```

131 # SI = exposure.adjust_log(SI)
132 SI[SI < 0] = 0
133 SI = SI.astype('uint8')
134 SI = np.flipud(SI)
135
136 if fulltime and usemask:
137     SI = SI * mask
138
139 print("SI.shape: ", SI.shape)
140
141 Title = 'Imaging result of RDA('
142
143 Title = 'Imaging Result of ' + imagingMethod
144
145
146 if usesrc or rcmc:
147     Title = Title + '\n (' 
148     if usesrc:
149         Title = Title + 'SRC+'
150     if rcmc:
151         Title = Title + 'RCMC+'
152 if usedpc:
153     Title = Title + 'DPC'
154
155 Title = Title + ")"
156
157
158 cmap = 'gray'
159 # cmap = 'hot'
160 # cmap = None
161
162 extent = SA
163 extent = None
164
165 plt.figure()
166 plt.subplot(211)
167 plt.imshow(np.abs(Sr), cmap=cmap)
168 plt.title("SAR raw data(Amplitude)")
169 plt.xlabel("Range")
170 plt.ylabel("Azimuth")
171 plt.subplot(212)
172 plt.imshow(SI, extent=extent, cmap=cmap)
173 plt.title(Title)
174 plt.xlabel("Range")
175 plt.ylabel("Azimuth")
176 plt.tight_layout()
177 plt.show()

```

(下页继续)

(续上页)

```

178
179 plt.figure()
180 plt.imshow(SI, extent=extent, cmap='gray')
181 plt.xlabel("Range/m")
182 plt.ylabel("Azimuth/m")
183 plt.title>Title)
184 plt.show()

```

7.5.8.2.3.3 实验结果

观察实验结果可以发现, 场景中某一目标信息会分布在回波数据的不同位置, 如果仅选取目标附近子区域成像, 会使得信息丢失, 成像模糊.

7.5.9 参考文献

7.5.10 名词术语

Depression Angle 倾角

Fucoused Data 聚焦后的数据, 即成像数据, 仍然是复数数据.

Grazing Angle 猎地角, 即波束照射方向与地平面间的夹角

Incidence Angle 入射角, 即波束照射方向与地平面的法线间的夹角

Orbital Inclination 轨道倾角 ([Orbital Inclination](http://en.volupedia.org/wiki/Orbital_inclination) (http://en.volupedia.org/wiki/Orbital_inclination)) 描述某一轨道相对于另一参考平面的倾斜角度, 如卫星的轨道倾角一般是指卫星轨道相对于地球赤道平面的倾斜角度. 如下图:

Phase History Data 即经调制解调后的 SAR 基带信号数据, 也 SAR 原始数据, 属于未聚焦数据 (*unfocused data*).

Polarized Synthetic Aperture Radar 极化合成孔径雷达 ([Polarized Synthetic Aperture Radar](http://en.volupedia.org/wiki/Synthetic-aperture_radar) (http://en.volupedia.org/wiki/Synthetic-aperture_radar), PolSAR) 是一种雷达, 用于对二维或三维场景物体 (如景观) 的成像与重建. 极化 SAR 利用.

Second Range Compression 二次距离压缩 (Second Range Compression, SRC) 针对于中等斜视下的数据, 对距离-方位目标相位历史的耦合进行补偿, 从而减小斜视或大孔径下的相位耦合畸变.

Squint Angle 斜视角 ([Squint](http://en.volupedia.org/wiki/Squint_(antenna)) ([http://en.volupedia.org/wiki/Squint_\(antenna\)](http://en.volupedia.org/wiki/Squint_(antenna)))) 斜距矢量与零多普勒平面间的夹角

Synthetic Aperture Radar 合成孔径雷达 ([Synthetic Aperture Radar](http://en.volupedia.org/wiki/Synthetic-aperture_radar) (http://en.volupedia.org/wiki/Synthetic-aperture_radar), SAR) 是一种雷达, 用于对二维或三维场景物体 (如景观) 的成像与重建. SAR 利用雷达天线在目标区域上的运动来提供比传统波束扫描式雷达更高的空间分辨率.

Unfocused Data 未聚焦的数据, 即 SAR 原始数据, 相位历史域数据.

Zero Doppler Time 随着平台或目标的运动, 目标与平台间不会产生多普勒效应的时刻.

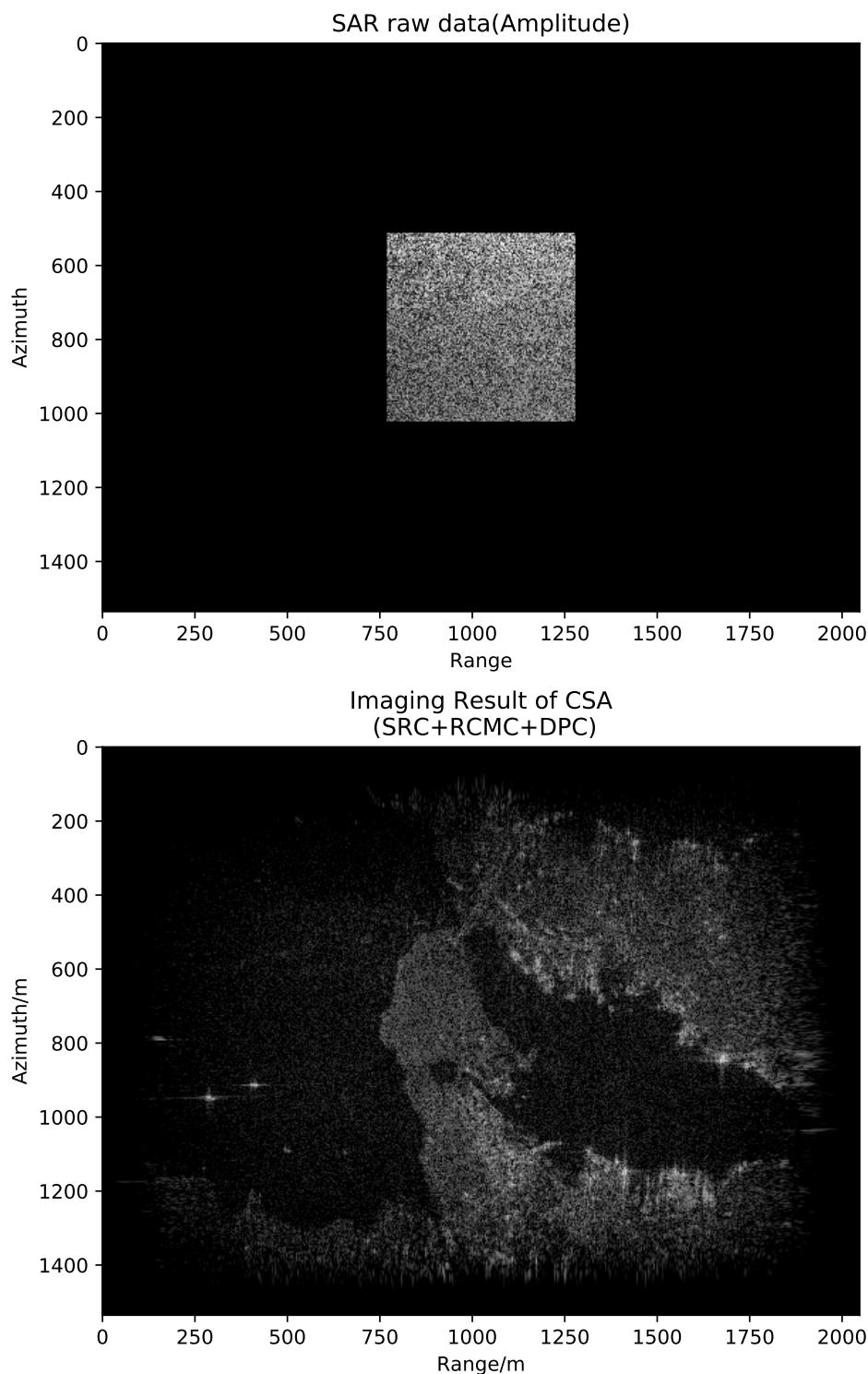


图 7.130: 选取子区域大小 512×512 , 子区域外的数据做补零处理后进行成像.

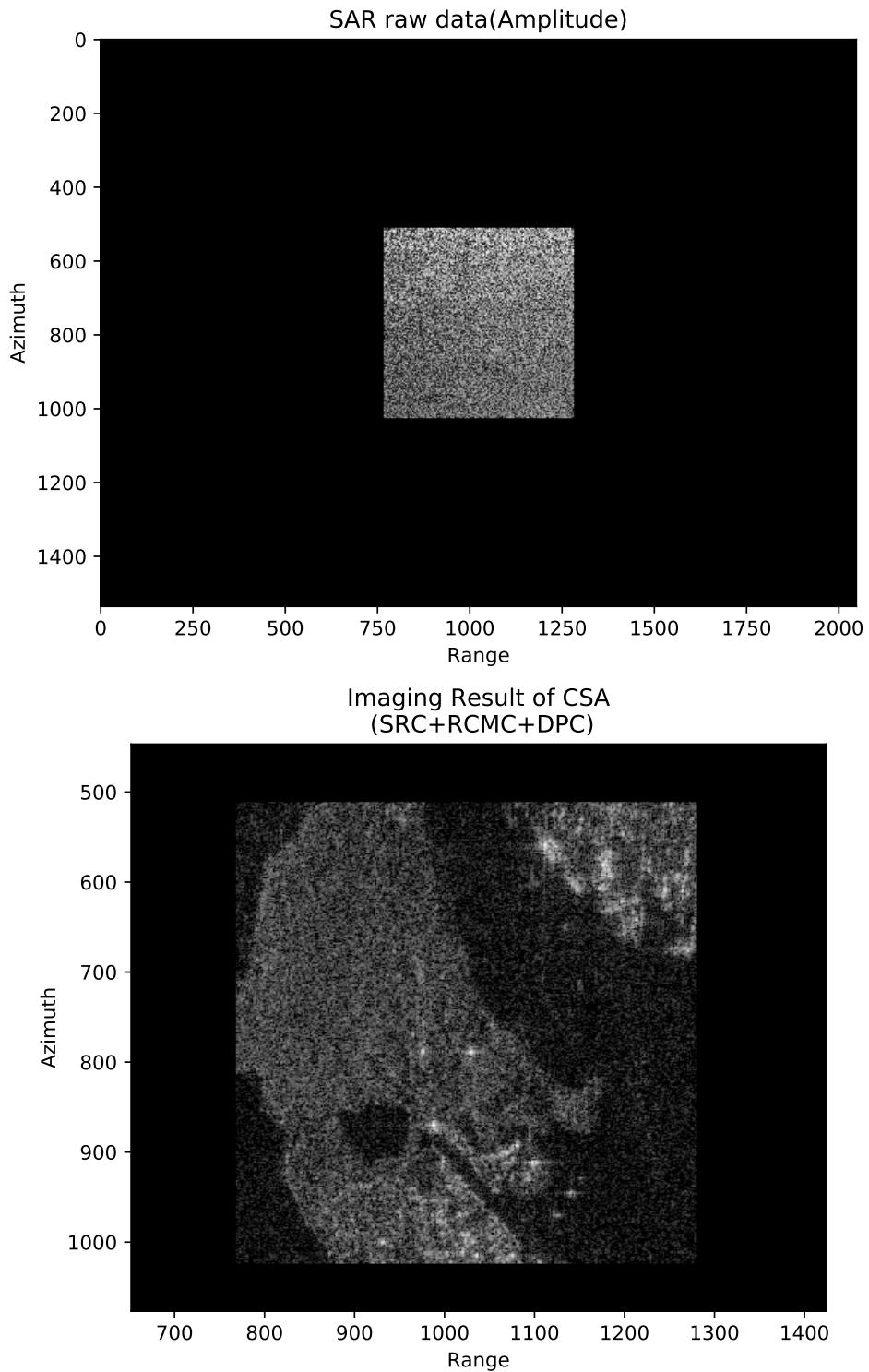


图 7.131: 选取子区域大小 512×512 , 子区域外的数据做补零处理后进行成像, 成像后子区域外成像数据被置零且进行局部放大后显示.

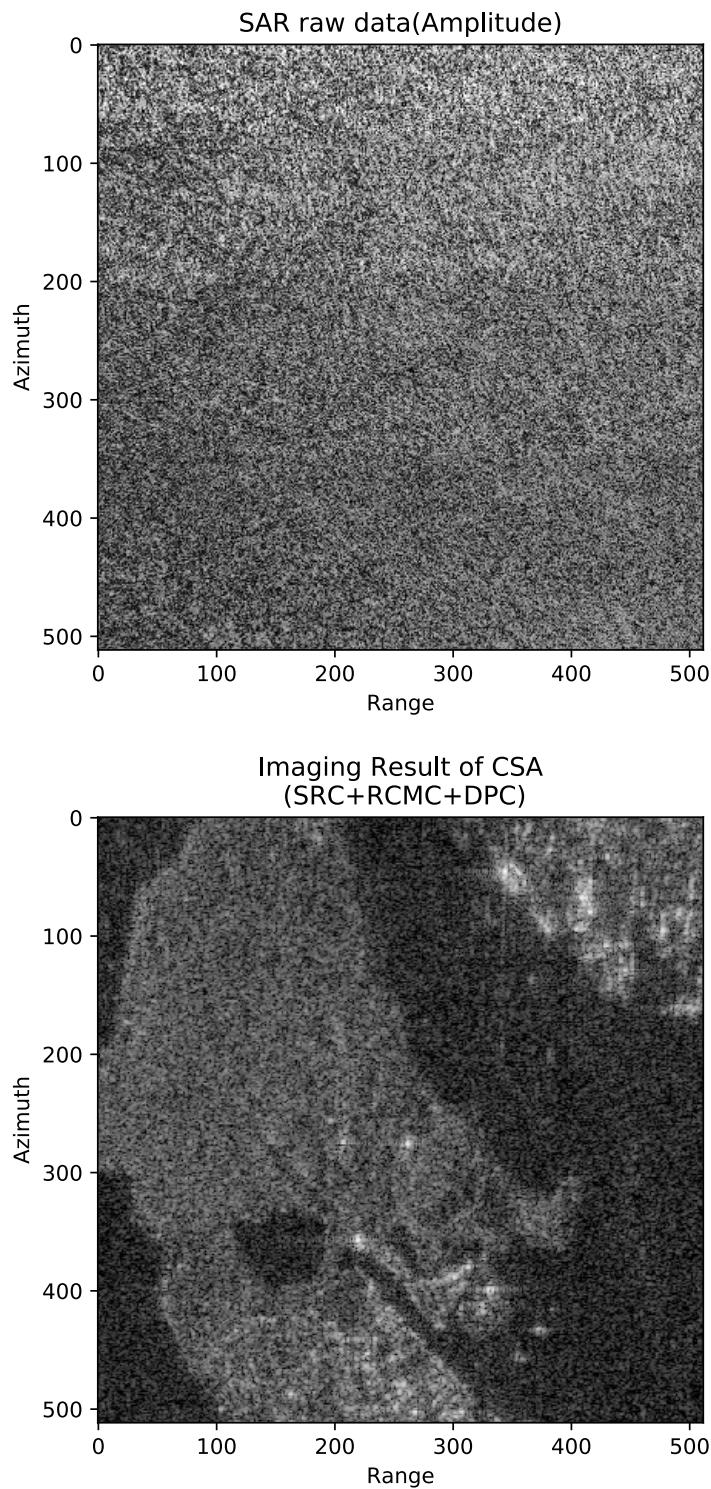


图 7.132: 选取子区域大小 512×512 , 仅选取的数据被用来成像.

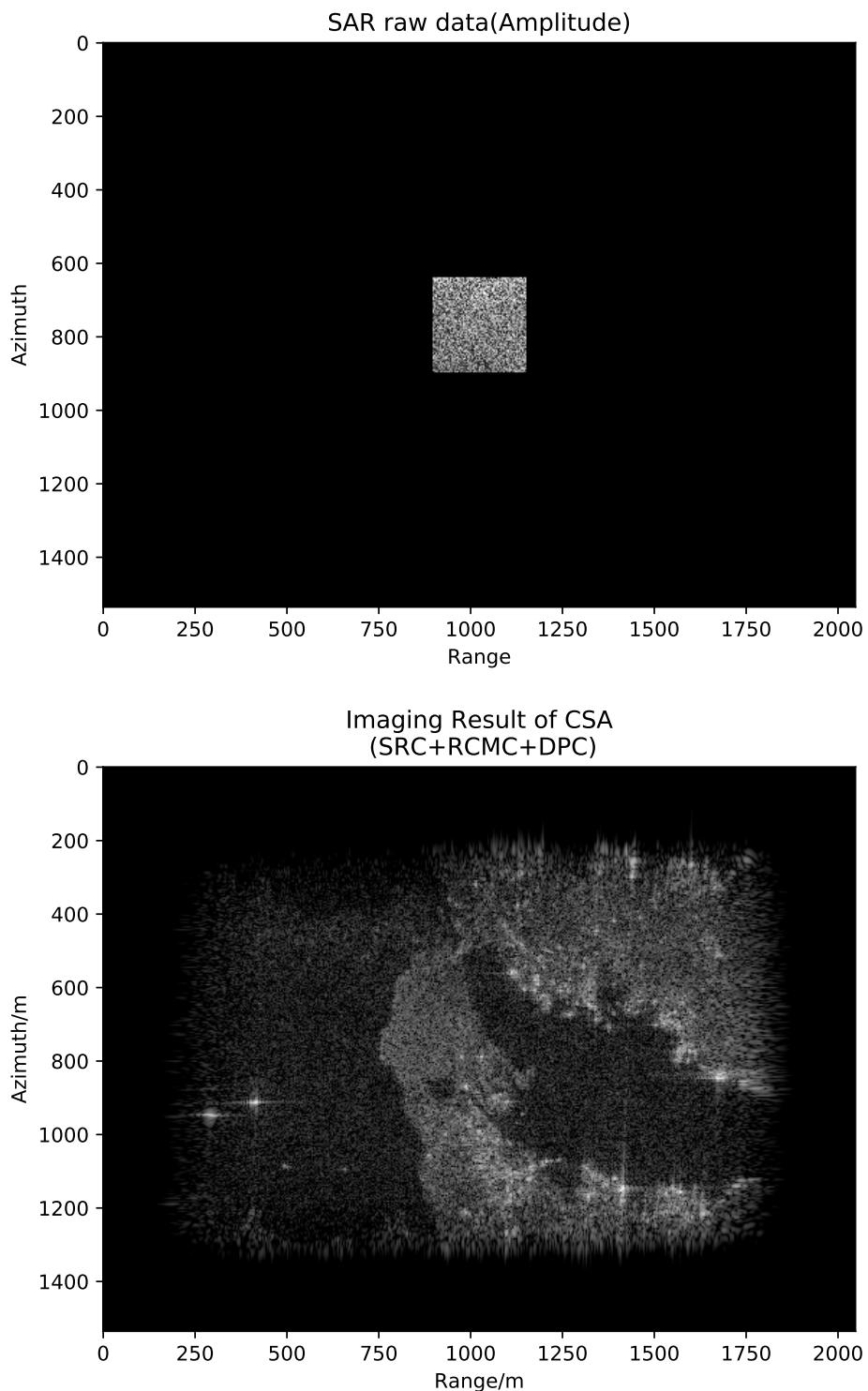


图 7.133: 选取子区域大小 256×256 , 子区域外的数据做补零处理后进行成像.

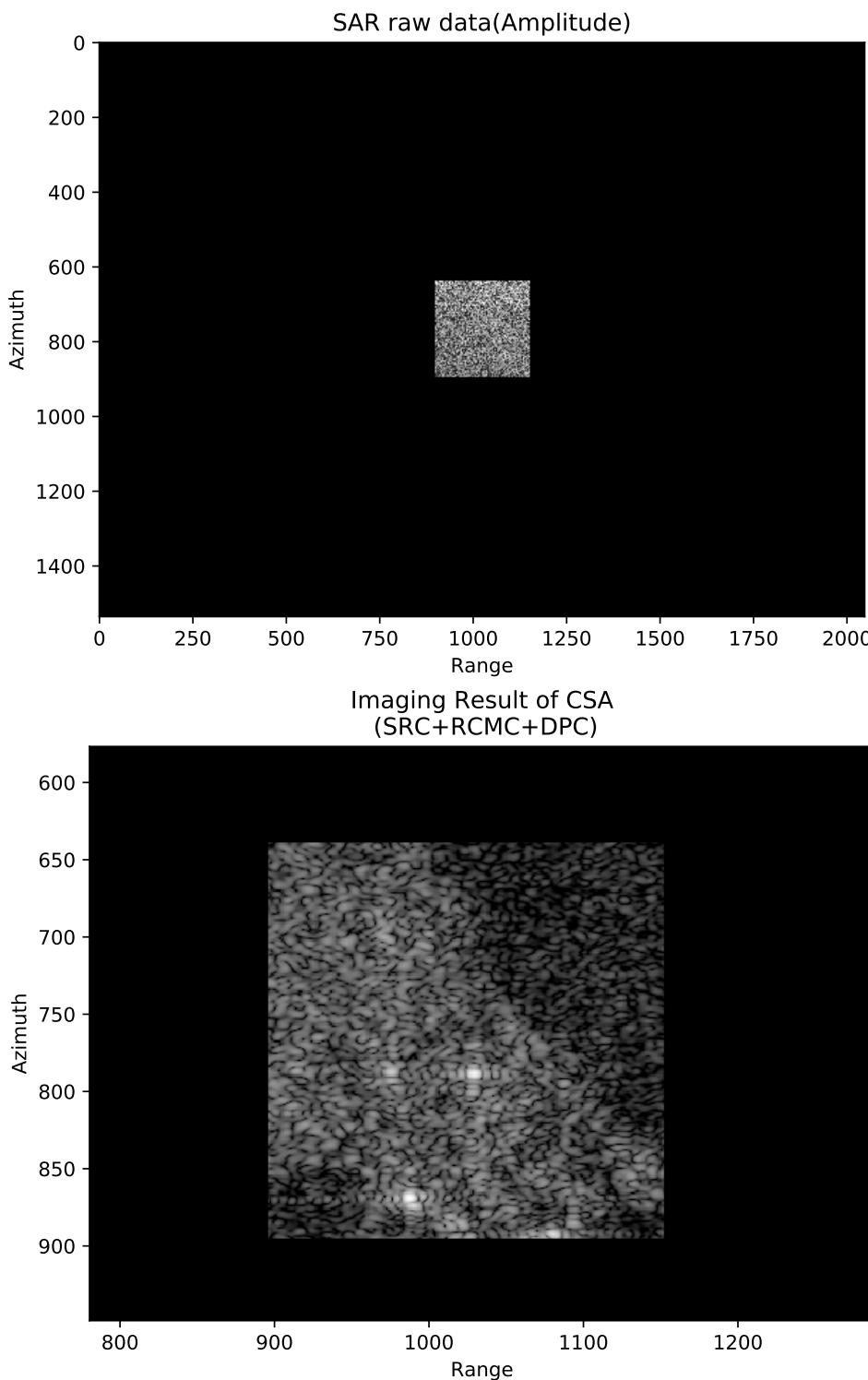


图 7.134: 选取子区域大小 256×256 , 子区域外的数据做补零处理后进行成像, 成像后子区域外成像数据被置零且进行局部放大后显示.

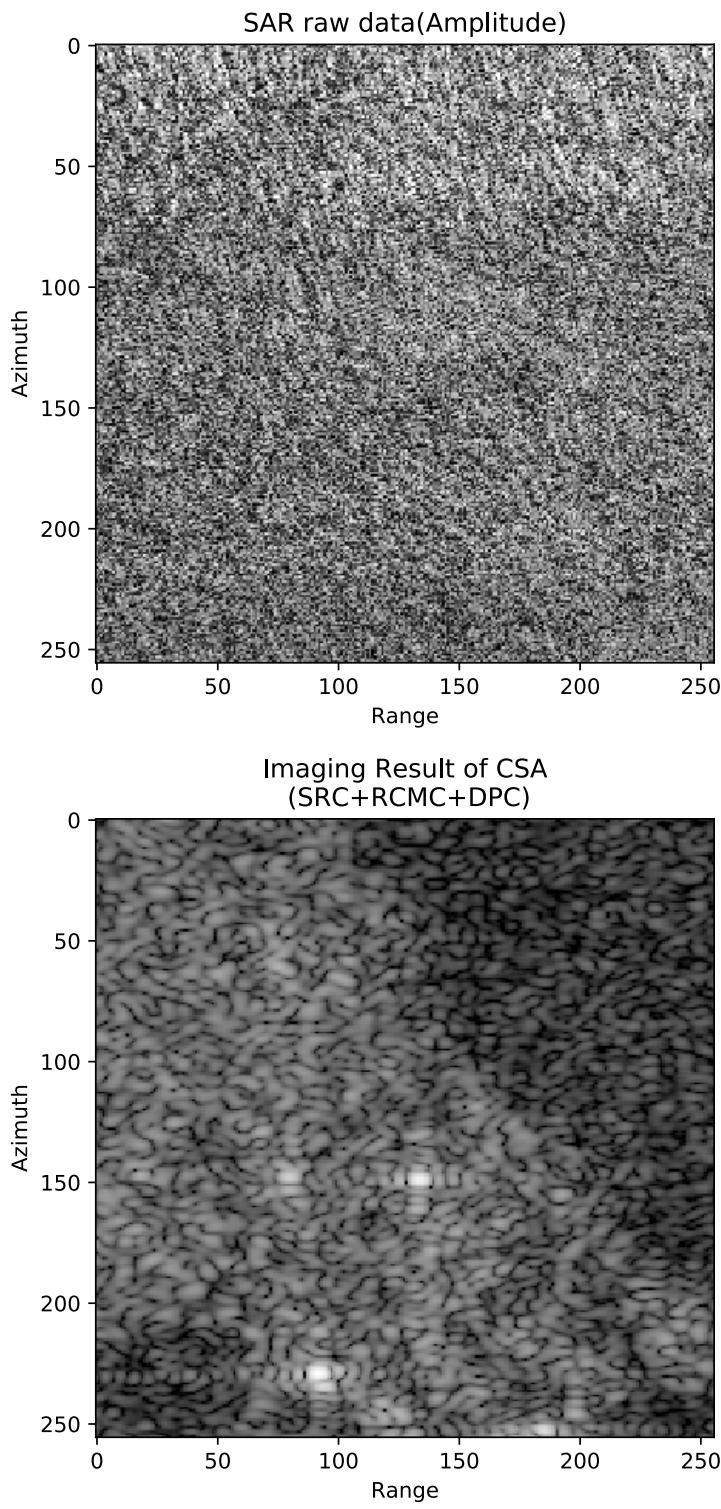


图 7.135: 选取子区域大小 256×256 , 仅选取的数据被用来成像.

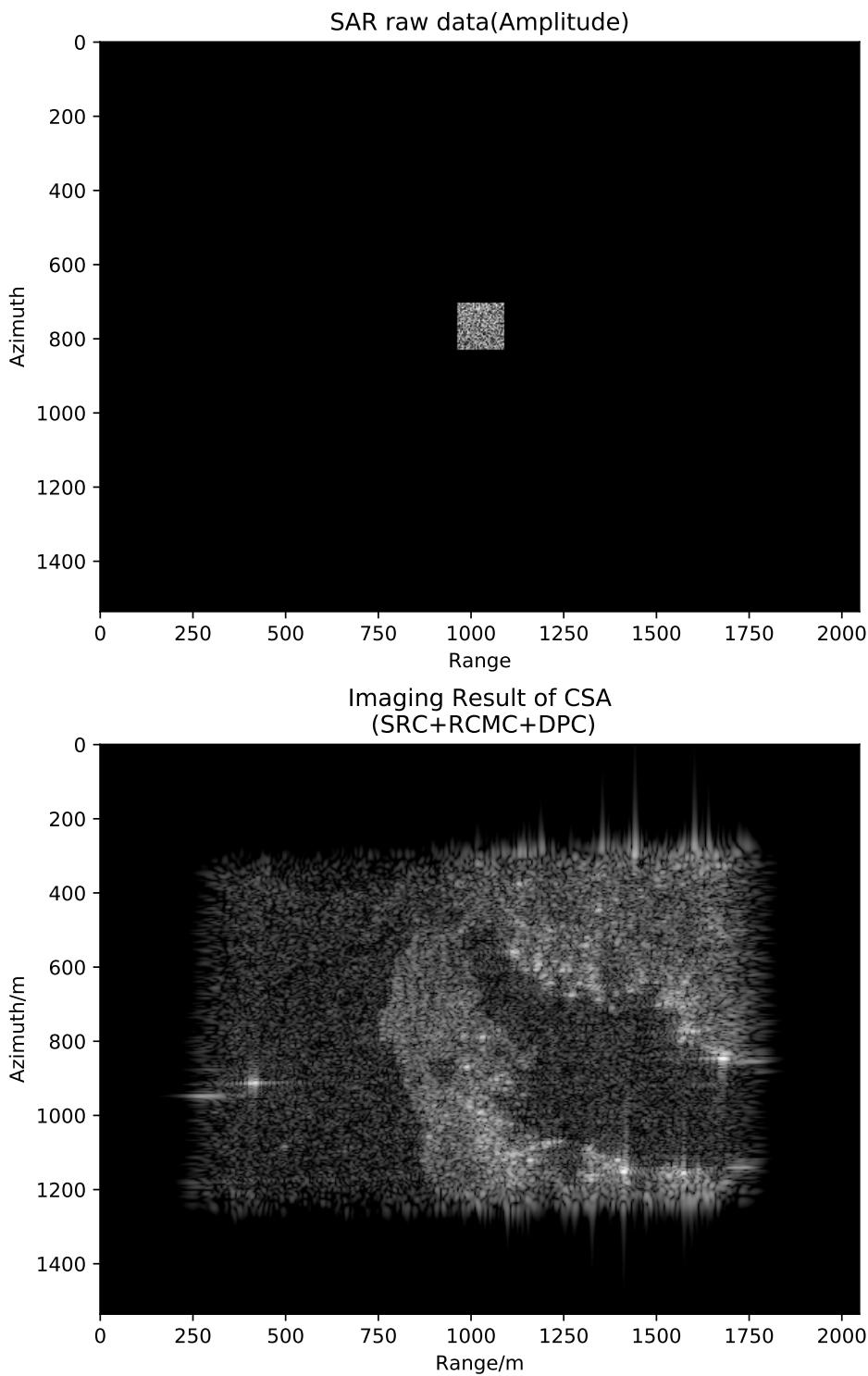


图 7.136: 选取子区域大小 128×128 , 子区域外的数据做补零处理后进行成像.

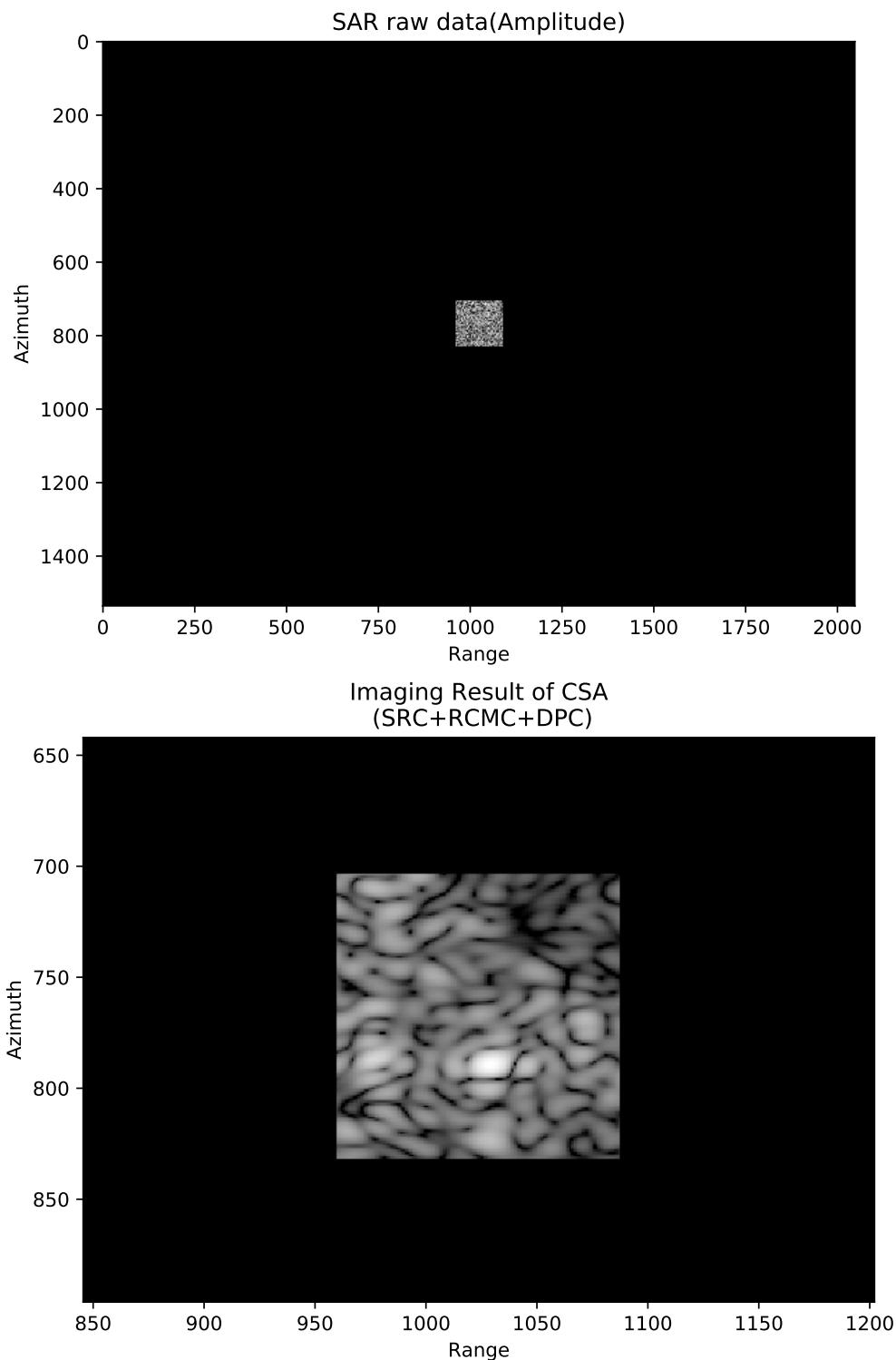


图 7.137: 选取子区域大小 128×128 , 子区域外的数据做补零处理后进行成像, 成像后子区域外成像数据被置零且进行局部放大后显示.

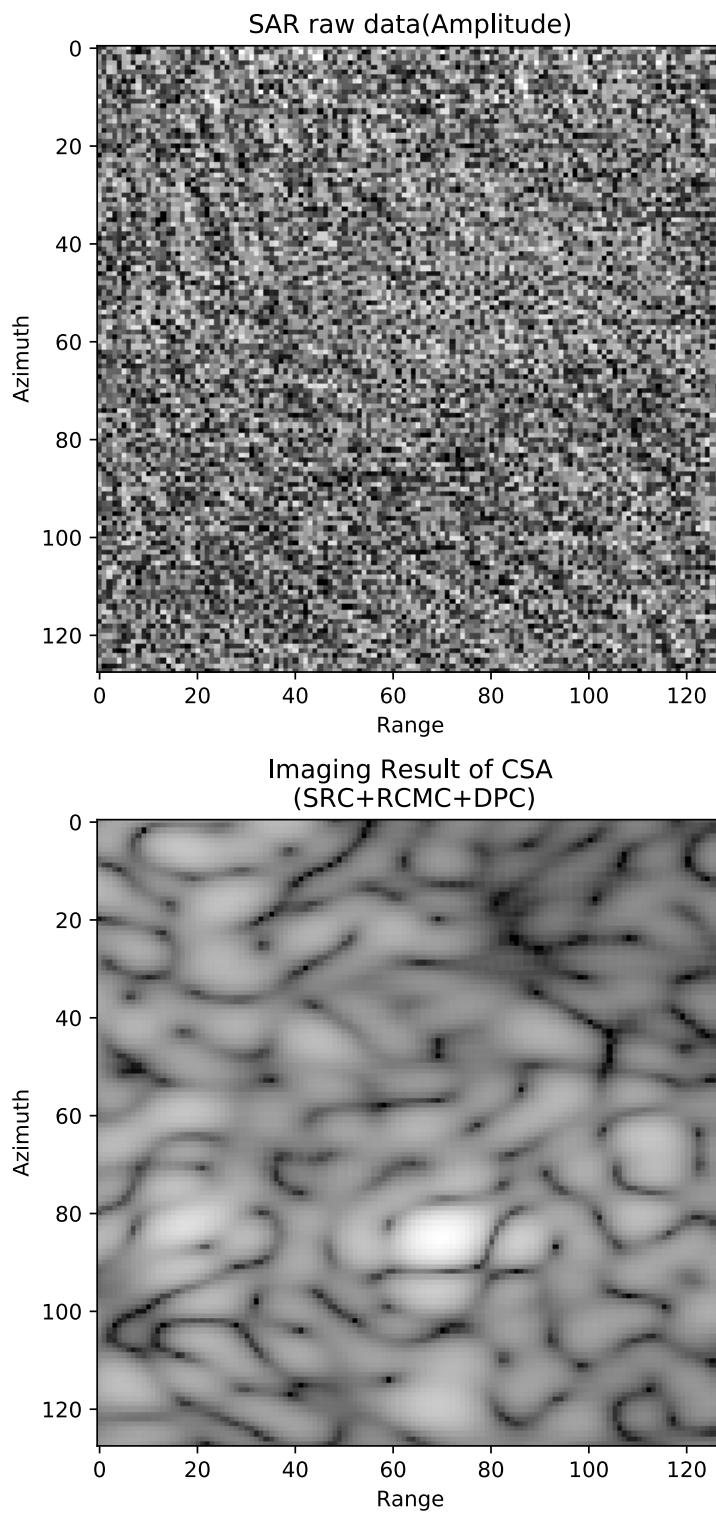


图 7.138: 选取子区域大小 128×128 , 仅选取的数据被用来成像.

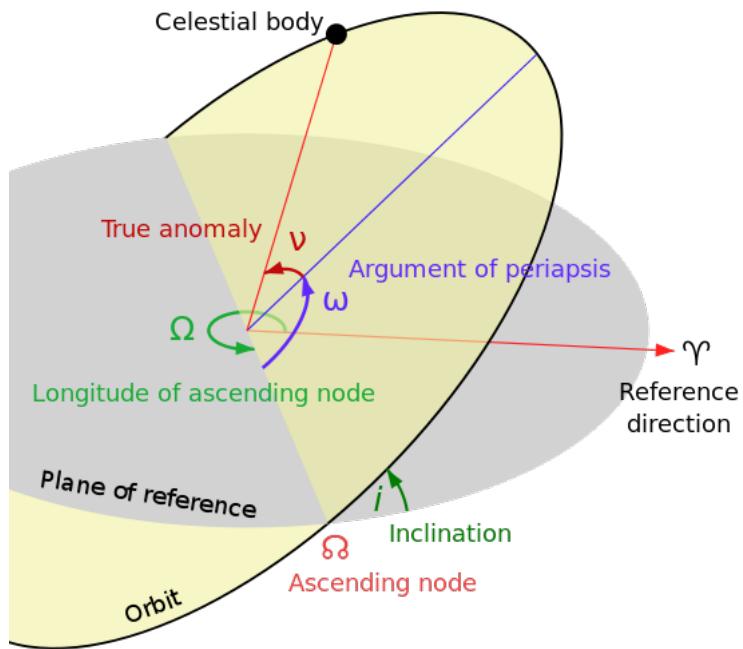


图 7.139: 轨道倾角示意图 (来自维基百科), 其中灰色区域为参考平面.

7.6 极化合成孔径雷达

7.6.1 极化 SAR 成像理论

7.6.2 极化 SAR 超分辨成像

7.6.3 极化 SAR 图像超分辨

7.7 共用技术

7.7.1 杂波抑制

7.7.1.1 动目标指示器

7.7.2 雷达散射截面积

7.7.2.1 基础概念

7.7.2.1.1 什么是雷达散射截面积

雷达散射截面积 (*Radar Cross Section*, RCS) 用于度量目标在雷达接收方向上反射的信号能量密度. 目标的 RCS 通常取决于目标的形状与材质材质, 雷达的工作参数.

7.7.2.1.2 雷达散射系数的分类

常见的有三种雷达散射系数, 第一种是雷达亮度(反射率)系数, 常称为 Beta Naught, 用 β 表示, 其在斜距方向上的单位面积的反射率是无量纲的. 第二种是标准后向散射系数 Sigma naught, 常指目标反射回来的波的强度, 常记为 σ , 单位为 dB. 第三种是归一化后向散射系数 Gamma naught, 即入射角归一化后向散射系数, 常记为 γ .

实现参考 [ASF MapReady calibration](https://github.com/asfadmin/ASF_MapReady/blob/devel/src/libasf_ardop/calibration.c) (https://github.com/asfadmin/ASF_MapReady/blob/devel/src/libasf_ardop/calibration.c)

7.7.3 名词术语

Radar Cross Section 雷达散射截面积 (Radar Cross Section, RCS) 用于度量目标在雷达接收方向上反射的信号能量密度.

7.8 补充内容

7.8.1 雷达产品

7.8.1.1 简介

7.8.1.1.1 SAR 雷达产品概览

7.8.1.2 雷达数据获取

7.8.1.2.1 合成孔径雷达

7.8.1.2.1.1 Alaska Satellite Facility

7.8.1.2.1.2 公开数据介绍

阿拉斯加卫星设施 (Alaska Satellite Facility, ASF) 下载, 处理, 存档遥感数据并向世界各地的科学家分发, 其目的是使得遥感数据可公开访问获取. [ASF 主页](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 给出了数据的获取方式, 数据的处理及可视化教程. 开放的数据覆盖星载与机载平台, 涉及 SAR, 极化 SAR, 干涉 SAR 等数据, 详细的开放数据信息可以从 [ASF SAR 公开数据列表](https://www.asf.alaska.edu/get-started-2/#daac_data) (https://www.asf.alaska.edu/get-started-2/#daac_data) 获得, 包含数据获取平台, 数据存储格式, 数据的处理软件等信息.

数据集	日期	类型
ALOS	2006-2011	SAR
Sentinel	2014-Ongoing	Spaceborne SAR
UAVSAR	2008-Ongoing	Airborne SAR
ERS-1	1991-1997	Spaceborne SAR
ERS-2	1995-2011	Spaceborne SAR
AIRSAR	1990-2004	Airborne SAR
Seasat	1978-1978	Spaceborne SAR

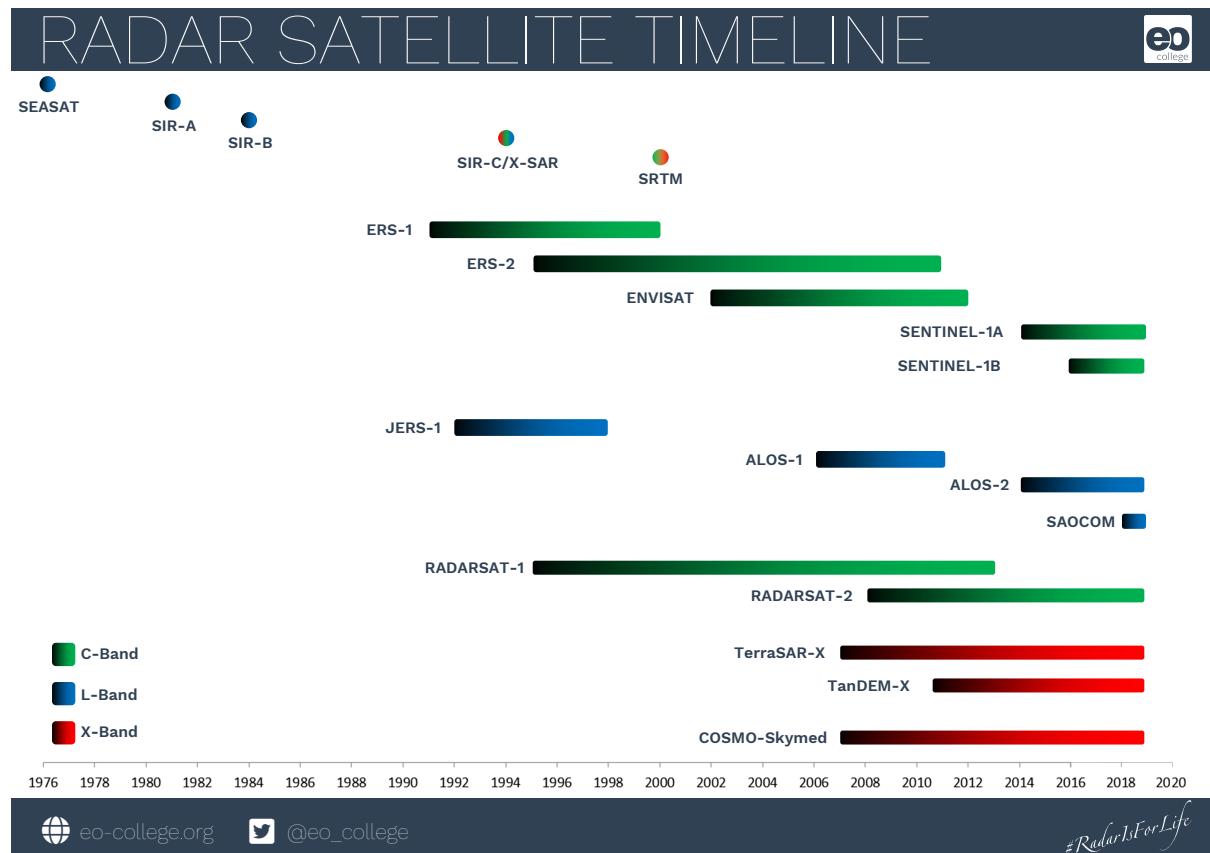


图 7.140: 常见雷达服务历程

7.8.1.2.1.3 公开数据检索与下载

7.8.1.2.1.4 公开数据格式

从 [这里](https://www.asf.alaska.edu/sar-information/data-formats-in-depth/) (<https://www.asf.alaska.edu/sar-information/data-formats-in-depth/>) 可以查看 CEOS, GeoTIFF, SAFE, HDF5, UAVSAR, AIRSAR 六种 SAR 数据格式的说明。

7.8.1.2.1.5 WInSAR

- [homepage](https://winsar.unavco.org/) (<https://winsar.unavco.org/>) : 含 InSAR 数据, 软件

7.8.1.3 雷达产品数据格式简介

7.8.1.3.1 CEOS

7.8.1.4 RADARSAT 产品介绍

RADARSAT 产品包含 RADARSAT1 和 RADASAT2 两中产品

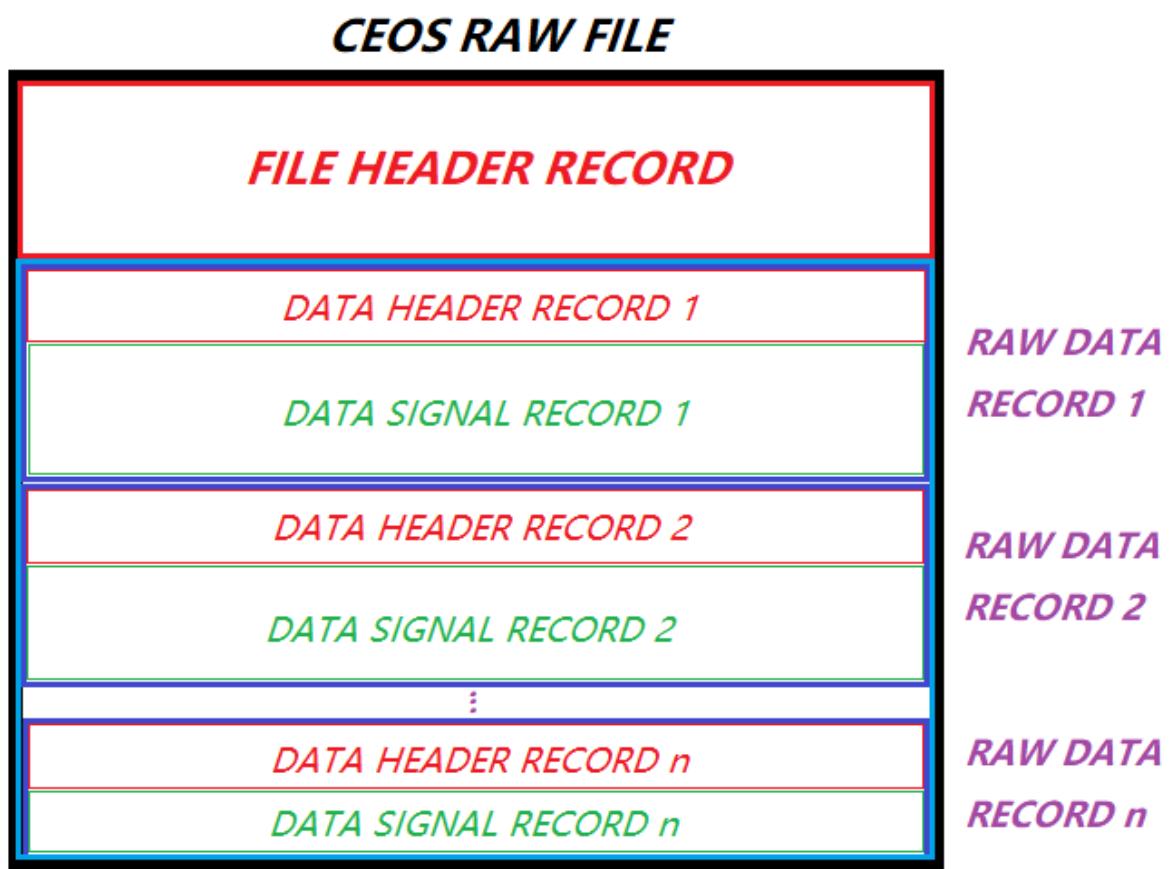


图 7.141: CEOS raw file Structure

7.8.1.4.1 RADARSAT1 产品

7.8.1.4.1.1 RADARSAT1 雷达参数与工作模式

RADARSAT SAR 平台工作参数信息可以从 [这里](http://www.asc-csa.gc.ca/eng/satellites/radarsat/radarsat-tableau.asp) (<http://www.asc-csa.gc.ca/eng/satellites/radarsat/radarsat-tableau.asp>) 查到.

RADARSAT1 的详细参数在 [这里](http://www.asc-csa.gc.ca/eng/satellites/radarsat1/components.asp) (<http://www.asc-csa.gc.ca/eng/satellites/radarsat1/components.asp>) . 可知 RADARSAT1 成像模式为右视 (right looking) 条带式 (stripmap mode) 成像, 属于 C 波段雷达, 极化方式为 HH 极化. 轨道倾角 (*Orbital Inclination*) 为 98.6° 与雷达的斜视角 (*Squint Angle*) 不同, 由于轨道倾角 (相对于地球赤道的倾角) 为 98.6°, 所以卫星处于太阳同步轨道 (参见 [太阳同步轨道](#) (页 600) 小节), 经过南北两极. 成像时用到的 RADARSAT 雷达参数如 [表 7.2](#) 所示.

表 7.2: RADARSAT1 平台参数

参数	符号	值	单位
航天器航向		344.49	°
平台纬度		48.36	°
平台经度		229.29	°
平台高度	H	793-821	km
平台速度 (等效速度)	V_r	7062	m/s
距离向天线长度	L_r	1.5	m
方位向天线长度	L_a	15	m
雷达波长	λ	0.05657	m
雷达载频	f_c	5.3	GHz
脉冲宽度	T_p	41.75	s
距离向调频率	K_r	-7.2135e+11	Hz/s
距离向带宽	B_r	30.12	MHz
距离向采样率	F_{rs}	32.317e+6	Hz
距离向采样数	N_r	9288	
方位向调频率	K_a	1733	Hz/s
方位向采样率	$F_{as} = PRF$	1256.98	Hz
方位向采样数	N_a	不定, 19438	
多普勒中心频率	f_{η_c}	-6900	Hz
成像分辨率	$\Delta_a \times \Delta_r$	8×8	m
轨道运行周期	T_o	100.7	min
卫星轨道半径	r	7149752	m
地球极地半径	R_e	6356752	m
地球平均半径	R_e	6367856	m
轨道倾角		98.6	°
近距入射角		38.64	°
中距入射角		40.15	°
远距入射角		41.61	°

不同成像模式下分辨率, 幅宽, 入射角不一致. 如入射角 (Incidence Angles) 在精细 (Fine) 成像模式下为

$34 - 47^\circ$, 在标准(Standard)成像模式下为 $20 - 49^\circ$, 扩展先进模式下为 $52 - 58^\circ$. RADARSAT 支持的波束模式以及不同模式下的幅宽与分辨率如表 7.3, 表 7.4 所示.

表 7.3: RADARSAT1 的波束模式及对应幅宽分辨率 1

波束模式	幅宽 (km)	近似分辨率 (m)
低分辨 100m	500	100
中分辨 50m	350	50
中分辨 30m	125	30
中分辨 16m	30	16
高分辨 5m	30	5

表 7.4: RADARSAT1 的波束模式及对应幅宽分辨率 2

波束模式	幅宽 (km)	近似分辨率 (m)
Fine	45	8
Standard	100	30
Wide	125	30
ScanSAR Narrow	300	50
ScanSAR Wide	500	100
Extended High	75	18-27
Extended Low	170	30

提示: 幅宽是波束一次扫描照射的宽度, 即波束距离向上在地面的投影宽度.

7.8.1.4.1.2 RADARSAT1 参数分析

该小节通过查询的 RADARSAT1 工作参数, 计算部分成像需要的参数.

下面进行一些参数的计算, 其中地球半径取极地半径 $R_e = 6356752 \text{ m}$, 引力常量 $G_e = 3.98603e14 \text{ m}^3/\text{s}^2$.

- 卫星轨道半径 $R_s = R_e + H \in [6356752 + 793000, 6356752 + 821000] = [7149752 \text{ m}, 7177752 \text{ m}]$
- 轨道运行周期 $T_o = 2\sqrt{\frac{R_s^3}{G_e}} \in [100.3 \text{ min}, 100.9 \text{ min}]$
- 卫星运行角速度 $\omega_s = 2\pi/T_o \in [0.00103785 \text{ rad/s}, 0.00104406 \text{ rad/s}]$
- 卫星飞行速度 $V_s = R_s\omega_s \in [7420.419 \text{ m/s}, 7494.042 \text{ m/s}]$

距离向调频带宽 $B_r = |K_r|T_p = 30.12e6 \text{ Hz}$, 距离向分辨率 $\Delta_r = c/2B_r = 4.98 \text{ m}$ 但是查到的是 8 m , 这是怎么回事呢, 这是因为上述计算的是斜距分辨率, 其地距分辨率为 $\Delta_x = \Delta_r/\sin\theta_i \approx \Delta_r/\sin 38.64^\circ \approx 8.08 \text{ m}$. 其中, θ_i 表示入射角.

由多普勒中心频率 $f_{\eta_c} = 2V\sin\theta_s/\lambda$ 可以计算出斜视角约为 $\theta_s \approx -1.58^\circ$, 即后侧视.

由方位向采样率及方位向采样点数可知方位向采样时间 $T_a = N_a/F_{as} = 15.464 \text{ s}$, 这些时间内, 雷达方位向飞行的距离为 $VT_a = 108987.85 \text{ m}$.

由距离向采样率及距离向采样点数知距离向采样时间为 $T_r = N_r/F_{rs} = 28.74e-3 \text{ s}$

7.8.1.4.1.3 RADARSAT1 数据读取

7.8.1.4.1.4 数据简介

数据源于书籍 [2][1] 中提供的温哥华 (Vancouver) 地区的 RADARSAT1 数据, 其场景图如 图 7.142 所示. 部分典型的地标已经在图上标出, 含史丹利公园 (Stanley Park), 哥伦比亚大学 (University of British Columbia), 温哥华机场 (Vancouver Airport) 等等.

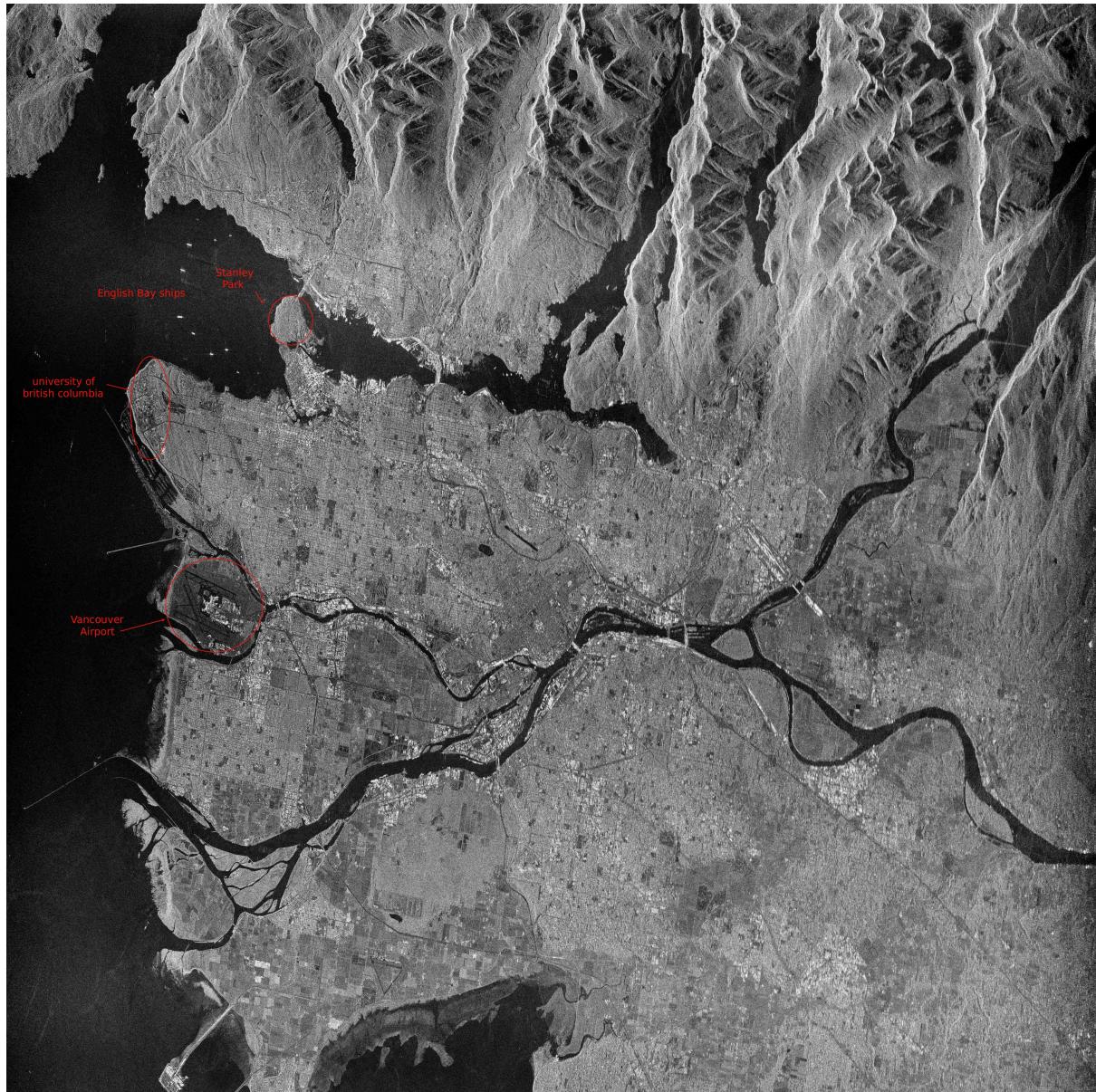


图 7.142: Image of Vancouver scence

7.8.1.4.1.5 数据格式与读取

RADARSAT1 数据采用 CEOS(参见 [CEOS](#) (页 546) 小节) 格式记录存储数据, 文件后缀是 .001, 数据及读取源码可在作者 [主页](#) (http://sar.ece.ubc.ca/SAR_Book/) 下载, 也可以使用本书作者开发的雷达处理库 iprs 读取. 代码 7.2 给出了读取到的书籍 [2] 中提供的 RADARSAT1 数据中的文件头记录信息:

代码 7.2: File header information in Radarsat1 raw data file

```

1 Record sequence number [(1, 4), '1B4', [1]]
2 1-st record sub-type code [(5, 5), '1B1', [63]]
3 Record type code [(6, 6), '1B1', [192]]
4 2-nd record sub-type code [(7, 7), '1B1', [18]]
5 3-rd record sub-type code [(8, 8), '1B1', [18]]
6 Length of this record [(9, 12), '1B4', [16252]]
7 ASCII/EBCDIC flag [(13, 14), '1A2', ['A']]
8 Blanks [(15, 16), '1A2', [' ']]
9 Format control document ID [(17, 28), '1A12', ['CEOS-SAR-CCT']]
10 Format control document revision level [(29, 30), '1A2', ['B']]
11 File design descriptor revision letter [(31, 32), '1A2', ['B']]
12 Generating software release and revision level [(33, 44), '1A12', ['MSSAR 7.7.1']]
13 File number [(45, 48), '1I4', [2]]
14 File name [(49, 64), '1A16', ['RSAT-1-SAR-Raw ']]
15 Record sequence and location type flag [(65, 68), '1A4', ['FSEQ']]
16 Sequence number location [(69, 76), '1I8', [1]]
17 Sequence number field length [(77, 80), '1I4', [4]]
18 Record code and location type flag [(81, 84), '1A4', ['FTYP']]
19 Record code location [(85, 92), '1I8', [5]]
20 Record code field length [(93, 96), '1I4', [4]]
21 Record length and location type flag [(97, 100), '1A4', ['FLGT']]
22 Record length location [(101, 108), '1I8', [9]]
23 Record length field length [(109, 112), '1I4', [4]]
24 Reserved1 [(113, 113), '1I1', [[]]]
25 Reserved4 [(116, 116), '1I1', [[]]]
26 Reserved segment [(117, 180), '1A64', [
    ↲
    ]]
27 Number of SAR DATA records (nominal) [(181, 186), '1I6', [19438]]
28 SAR DATA record length (bytes) [(187, 192), '1I6', [18768]]
29 Reserved1 (blanks) [(193, 216), '1A4', [' ']]
30 Number of bits per sample [(217, 220), '1I4', [8]]
31 Number of samples per data group (or pixels) [(221, 224), '1I4', [2]]
32 Number of bytes per data group (or pixel) [(225, 228), '1I4', [2]]
33 Justification and order of samples within data group [(229, 232), '1A4', [' ']]
34 Number of SAR channels in this file [(233, 236), '1I4', [1]]
35 Number of lines per data set (nominal) [(237, 244), '1I8', [19438]]
36 Number of left border pixels per line [(245, 248), '1I4', [0]]
37 Total number of data groups per line per SAR channel [(249, 256), '1I8', [[]]]
38 Number of right border pixels per line [(257, 260), '1I4', [0]]
39 Number of top border lines [(261, 264), '1I4', [0]]

```

(下页继续)

(续上页)

```

40 Number of bottom border lines [(265, 268), '1I4', [0]]
41 Interleaving indicator [(269, 272), '1A4', ['BSQ']]
42 Number of physical records per line [(273, 274), '1I2', [1]]
43 Number of physical records per multi-channel line [(275, 276), '1I2', [1]]
44 Number of bytes of prefix data per record [(277, 280), '1I4', [180]]
45 Number of bytes of SAR data (or pixel data) per record (nominal) [(281, 288), '1I8
    ↪', [18576]]
46 Number of bytes of suffix data per record [(289, 292), '1I4', [0]]
47 Reserved2 [(293, 340), '1A48', ['        13 4PB 49 2PB 45 4PB 21 4PB 29 4PB
    ↪']]
48 Blanks [(341, 368), '1A28', ['        ']]
49 Reserved3 [(369, 400), '1A32', ['        ']]
50 SAR Data format type identifier [(401, 428), '1A28', ['COMPLEX INTEGER*2
    ↪']]
51 SAR Data format type code [(429, 432), '1A4', ['CI*2']]
52 Number of left fill bits within pixel [(433, 436), '1I4', [4]]
53 Number of right fill bits within pixel [(437, 440), '1I4', [0]]
54 Maximum data range of pixel (max-min value for I and Q) [(441, 448), '1I8', [15]]

```

可见, 数据格式为 CEOS-SAR-CCT, 该数据共 19384 行脉冲数据, 每行为一次脉冲回波, 一个脉冲含 9288 个距离单元, 每个像素含实部 (I) 虚部 (Q) 共 2 个字节, 故每行脉冲含 18576 个字节, 文件头记录长度为 16252 字节.

危险: RADARSAT1 原始信号数据记录不是固定长度, 每 8 帧会多出一段长度为 2880 字节 (1440 个复数) 的数据, 这些数据是发射脉冲复制品 (transmitted replica), 其中心频率是 0Hz, 线性调频率为 $0.72135e + 12Hz/s$, 该信号为数字生成信号, 且 4-bit 量化存储, 不够精确, 因而本书中不使用.

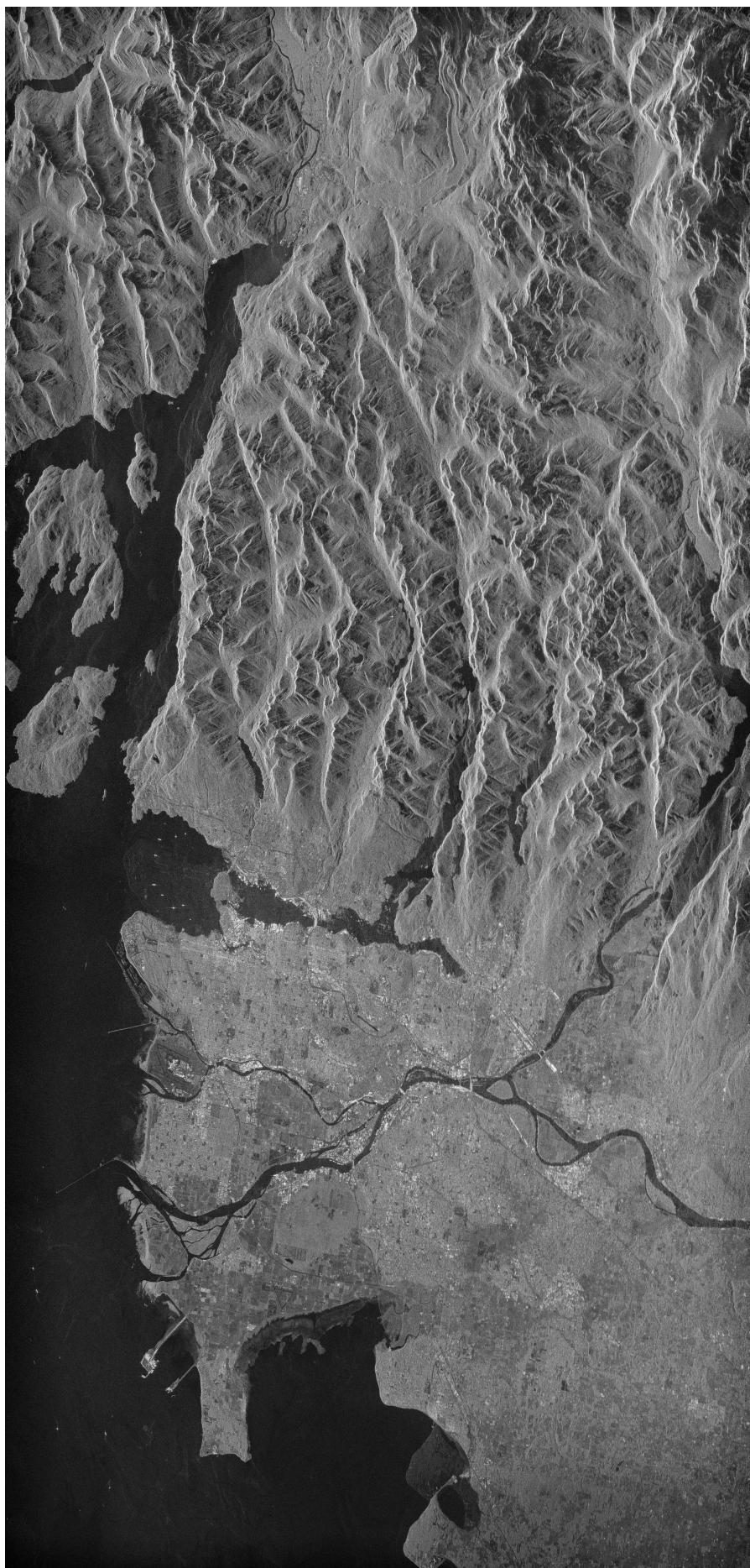
7.8.1.4.1.6 AGC 补偿

7.8.1.4.1.7 RADARSAT1 数据成像

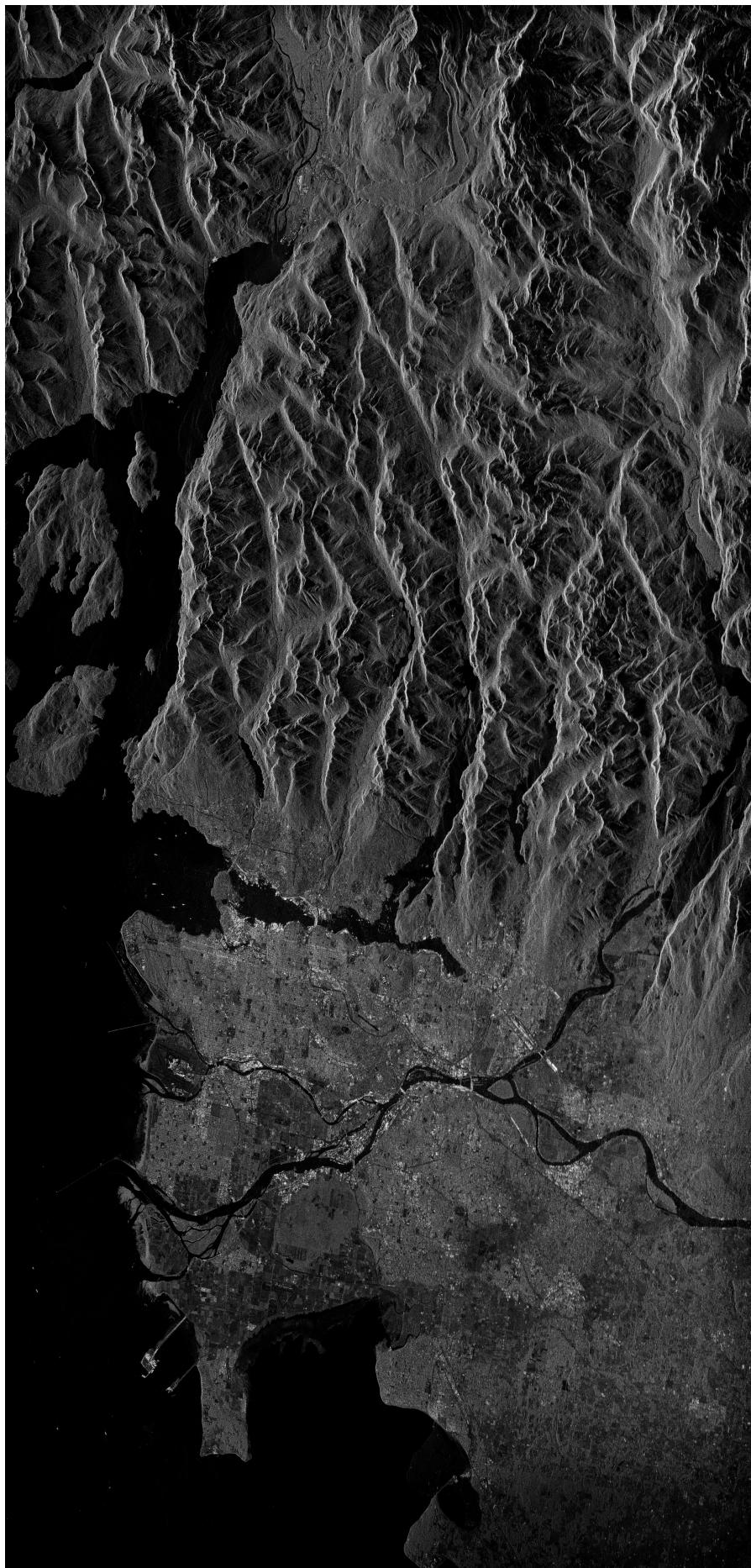
7.8.1.4.1.8 全部区域成像

图 7.143 和 图 7.144 给出了温哥华地区的单视成像结果 (进行了降采样, 距离向与方位向尺度均缩小为原来的四分之一, 插值采用 cubic 插值), 图 7.146 和 图 7.145 给出了温哥华地区的多视成像结果 (视数为 4, 多视处理后未进行降采样). 在 CSA 成像处理完成后, 还需要进行一些增强后处理操作 (本节结果未进行 RCS 换算, 仅进行对数增强, 数值截断图像增强操作, 参见 SubSection-LogContrastEnhancementDigitalImageSignalProcessing 小节), 才能得到图示结果, 若不进行, 图像会比较暗, 对比度低. 可见与书籍 [2][1] 中图 3.6 的结果基本一致.

图 7.143 所示为对数增强后, 对区间 [20, 50] 之外的数据进行截断处理的结果, 图 7.144 所示为对数增强后, 对区间 [30, 50] 之外的数据进行截断处理的结果, 图 7.146 所示为对数增强后, 对区间 [10, 50] 之外的数据进行截断处理的结果, 图 7.145 所示为对数增强后, 对区间 [20, 50] 之外的数据进行截断处理的结果.



7.8. 补充内容



554 | 7.144: Single look imaging result (with truncation parameters $a = 30, b = -50$) of Chirp Scaling Algorithm
Chapter 7. 第七章 雷达信号处理

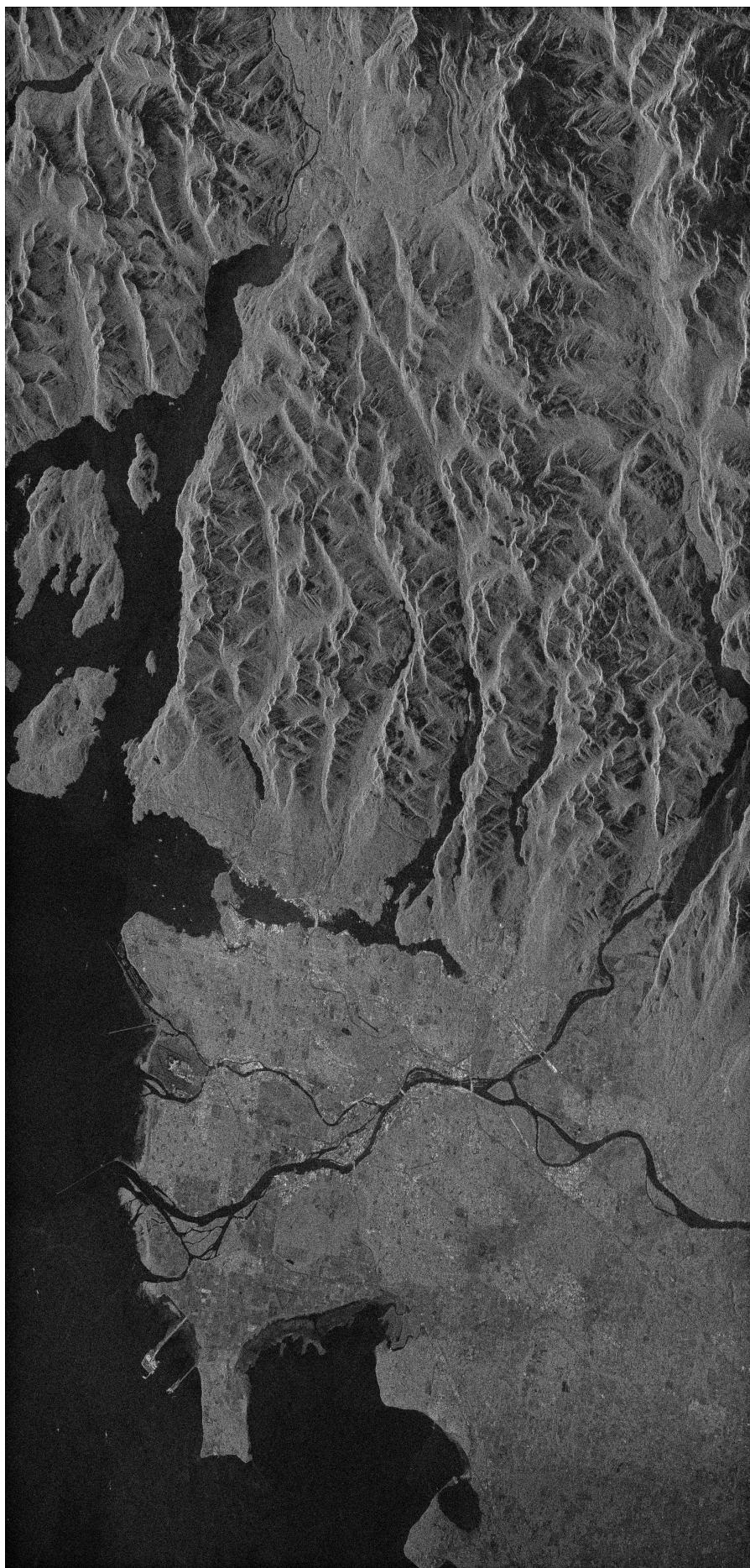


图 7.145: Multiple looks (4, 4) imaging result (with truncation parameters $a = 10, b = 50$) of Chirp Scaling Algorithm.
7.8. 补充内容 555

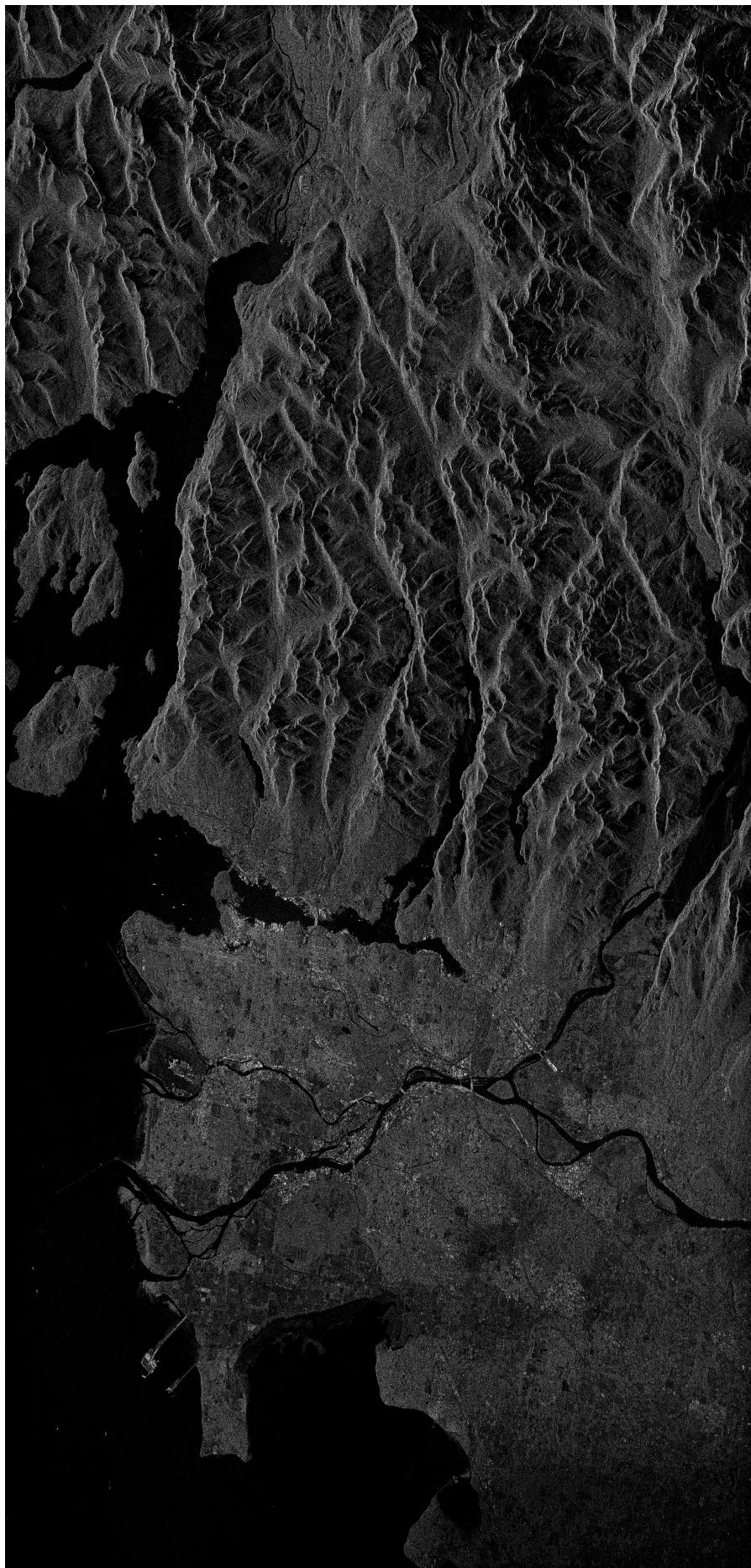


图 7.146: Multiple looks (4, 4) imaging result (with truncation parameters $\bar{C} = 20, b = 50$) of China Scanning Algorithm. Chapter 7. 第七章 雷达信号处理

警告: 首先, 本节结果未进行 RCS 转换, 其次, 使用 Python 中的 Matplotlib, PIL, Opencv 或 MATLAB 显示出来的图看起来像是有很多噪声点, 将矩阵保存成 .tiff 图像后, 用其它显示软件打开则相对平滑, 如 图 7.147 所示.

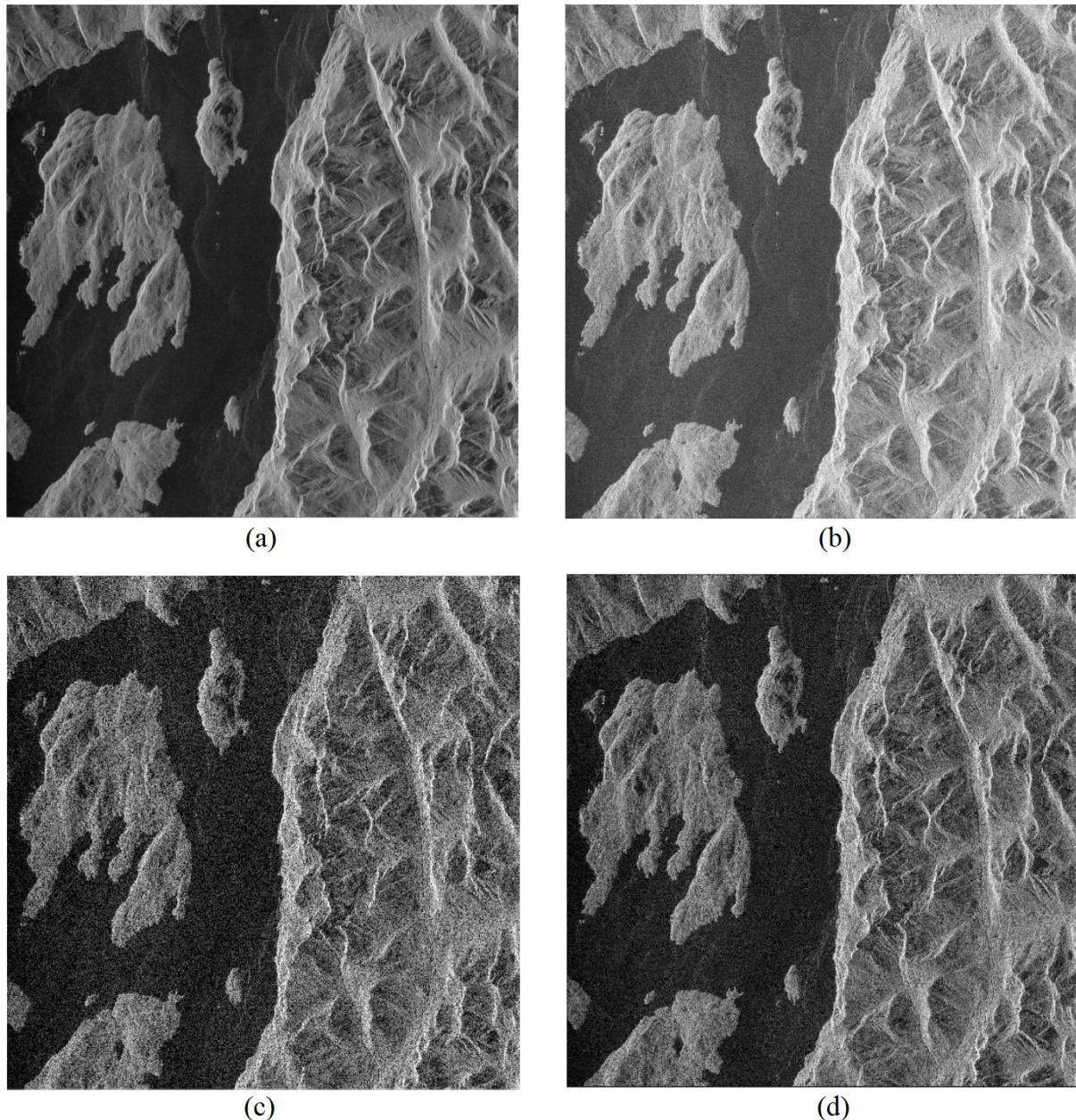


图 7.147: View imaging image with different programs. (a) Ubuntu Document Viewer, (b) ASF view program, (c) Matlab's `imshow()`, (d) Python's `plt.imshow()`.

7.8.1.4.1.9 部分区域成像

如前所述, RADARSAT1 幅宽为 45km , 距离向采样点数为 $N_r = 9280$, 每个像素点的距离分辨率为 $\delta_x = 45000/9280 = 4.85$ (注意这说明距离向过采样, 距离向的地距分辨率仍然为先前计算的 8m). 取数据中的 1536×2048 个像素作为待成像区域, 则地理区域为 $(1536 \times \delta_y) \times (2048 \times \delta_x) = 8896\text{m} \times 15886\text{m}$.

以史丹利公园区域数据为例, 如 图 7.148 所示为该区域的地图, 其地理区域区域范围测量结果已经在图中标出, 与上面计算的区域基本一致. 图 7.149 为 matplotlib 工具显示的该区域数据 CSA 成像结果, 其中, 坐标轴已经对应到实际距离, 可见与计算和地图测量结果基本一致. 图 7.149 显示的结果已经进行了下采样, 图 7.148 给出了该区域未进行下采样的成像结果.



图 7.148: Eath Map of Stanley Park

图 7.151, 图 7.152, 图 7.153 分别给出了哥伦比亚大学 (UBC), 温哥华机场 (Vancouver Airport), 斯阔米什 (Squamish Garibaldi) 地区数据的 CSA 算法成像结果.

7.8.1.4.2 RADARSAT2 产品介绍

7.8.1.4.2.1 RADARSAT2 雷达参数

7.8.1.5 ERS 系列产品介绍

7.8.1.5.1 雷达参数与工作模式

ERS1 与 ERS2 是欧洲航天局 (European Space Agency, ESA) 研发的两颗卫星, 分别于 1991 年与 1995 年发射, 两颗卫星搭载的相同参数的雷达. 雷达工作在 C 波段, 采用 HH 极化收发方式, 轨道重访周期为 35 天. 在 ERS1 与 ERS2 同时工作的期间, ESA 通过进行轨道调整和控制, 使得 ERS1 与 ERS2 间满足干涉测量要求, 从而使系统具备干涉测量的能力.

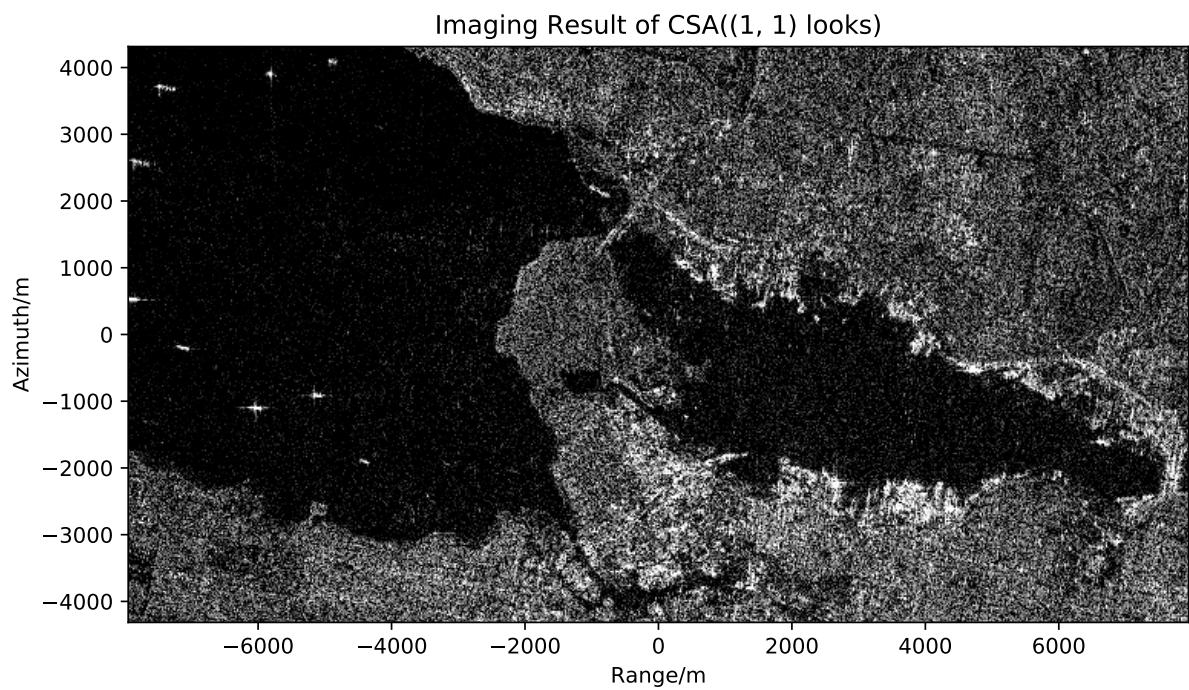


图 7.149: Imaging result of Chirp Scaling Algorithm (Stanley Park) shown by matplotlib.



图 7.150: Imaging result image of Chirp Scaling Algorithm (Stanley Park).



图 7.151: Imaging result of Chirp Scaling Algorithm (UBC).



图 7.152: Imaging result of Chirp Scaling Algorithm (Vancouver Airport).



图 7.153: Imaging result of Chirp Scaling Algorithm (Squamish Garibaldi).

具体雷达参数可以在 [这里](http://www.esa.int/Applications/Observing_the_Earth/ERS_1_and_2) (http://www.esa.int/Applications/Observing_the_Earth/ERS_1_and_2) 与 [这里](https://crisp.nus.edu.sg/ers/ers.html) (<https://crisp.nus.edu.sg/ers/ers.html>) 获得。总结如下：

表 7.5: ERS 平台参数

参数	符号	值	单位
平台高度	H	786070	m
平台速度	V	7098.0194	m/s
距离向天线长度	L_r	1	m
方位向天线长度	L_a	10	m
雷达波长	λ	0.05657	m
雷达载频	f_c	5.3	GHz
脉冲宽度	T_p	37.12	s
距离向调频率	K_r	4.18989015e+11	Hz/s
距离向带宽	B_r	15.55 ± 0.01	MHz
距离向采样率	F_{rs}	18.962468e+6	Hz
距离向采样数	N_r	5616	
方位向调频率	K_a	2122.96	Hz/s
方位向采样率	$F_{as} = PRF$	1640-1720, 1679.902	Hz
方位向采样数	N_a	不定	
多普勒中心频率	f_{η_c}	1257.769	Hz
成像分辨率	$\Delta_a \times \Delta_r$	5×24.6	m
幅宽		80400 or 102500	m
入射角	θ_i	23 (at mid-swath)	°
斜视角	θ_s	0	°

7.8.1.5.2 数据获取与读取

从 [ASF](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 上可以下载到 ERS 卫星 SAR 数据, 解压下载的 zip 数据文件, 可以得到 .raw .vol .ldr 等文件, 此种格式使用 **SNAP**, **NEST** 这些软件都无法打开, 可以使用 [GMTSAR](https://topex.ucsd.edu/gmtsar/) (<https://topex.ucsd.edu/gmtsar/>) 来读取数据或者本书作者开发的 [iprs](http://iridescent.ink/iprs3.0/) (<http://iridescent.ink/iprs3.0/>) 工具读取.

提示: 从 ASF 上下载数据很慢, 而且经常动不动中断, 后来找了个多线程下载工具, 写了个脚本终于顺利下载, 参见[多线程下载 \(页 152\)](#) 小节. 此外, GMTSAR 依赖于通用映射依赖 (Generic Mapping Tools, GMT) 软件, 需要先安装 GMT 依赖, 然而其下载速度相当慢, 可以从[国内镜像下载](http://mirrors.ustc.edu.cn/gmt/) (<http://mirrors.ustc.edu.cn/gmt/>). GMT 与 GMTSAR 的安装采用源码安装即可, 具体可参见源码目录中的自述文件, 安装过程比较简单, 不再赘述.

7.8.1.5.3 原始回波数据读取

ERS 原始 SAR 回波数据及单视复数产品文件结构描述均可以在 [这里](https://crisp.nus.edu.sg/ers/ers.html) (<https://crisp.nus.edu.sg/ers/ers.html>) 看到, 下面简要介绍原始数据的文件结构及数据提取. 本节以从 ASF 上下载的 *E2_84690_STD_L0_F137.zip* 原始回波数据文件和 *E2_84690_STD_F137.zip* 单视复数数据文件为例进行讲解.

7.8.1.5.3.1 目录及文件格式

ERS 产品采用 CEOS(参见[CEOS \(页 546\)](#) 小节) 格式存储数据,

在本书作者开发的 iprs 工具包中, 定义的 SAR 信号数据映射字典如[代码 7.3](#), 如记录长度 ('Length of this record') 字段的起止地址为 (9, 12), 占 4 个字节, 类型为 1B4, 即一个四字节的数. 又如字段 'Raw Data', 表示采样数据, 第一个样本的起止地址为 (413, 414), 占 2 个字节, 类型为 2B4, 即包含两个单字节整数.

代码 7.3: Sar Data File Signal Data Record Descriptor Dictionary defined
in iprs

```

1 SarDataFileSignalDataRecordERS = {
2     'Record sequence number': [(1, 4), '1B4', 0],
3     '1-st record sub-type code': [(5, 5), '1B1', 0],
4     'Record type code': [(6, 6), '1B1', 0],
5     '2-nd record sub-type code': [(7, 7), '1B1', 0],
6     '3-rd record sub-type code': [(8, 8), '1B1', 0],
7     'Length of this record': [(9, 12), '1B4', 0],
8     # PREFIX DATA - GENERAL INFORMATION
9     'SAR image data line number': [(13, 16), '1B4', 0],
10    'SAR image data record index (indicates the record sequence number of the
11    ↪image line)': [(17, 20), '1B4', 0],
12    'Actual count of left-fill pixels': [(21, 24), '1B4', 0],
13    'Actual count of data pixels (samples)': [(25, 28), '1B4', 0],
        'Actual count of right-fill pixels': [(29, 32), '1B4', 0],

```

(下页继续)

(续上页)

```

14     'Reserved1': [(33, 84), '1I52', 0],
15     'Spare1': [(85, 88), '1I4', 0],
16     'Spare2': [(89, 92), '1I4', 0],
17     'Reserved2': [(93, 124), '1I32', 0],
18     'Spare3': [(125, 128), '1I4', 0],
19     # PREFIX DATA PLATFORM REFERENCE INFORMATION
20     'Platform information': [(129, 192), '1B64', 0],
21     # PREFIX DATA - SENSOR/FACILITY SPECIFIC, AUXILIARY DATA
22     'Fixed code = AA in Hexadecimal notation': [(193, 193), '1B1', 0],
23     'OGRC/OBRC flag (1 or 0)': [(194, 194), '1B1', 0],
24     'ICU on board time': [(195, 198), '1B4', 0],
25     'Activity task': [(199, 200), '1B2', 0],
26     'Image format counter': [(201, 204), '1B4', 0],
27     'Sampling window start time': [(205, 206), '1B2', 0],
28     'Pulse repetition interval': [(207, 208), '1B2', 0],
29     'Calibration attenuation setting': [(209, 209), '1B1', 0],
30     'Receiver gain attenuation setting': [(210, 210), '1B1', 0],
31     'Spare4': [(211, 340), '130B1', 0],
32     '36 calibration pulses as (4bit spare 6bit Q 6bit I from MSB down to LSB)': [
33         # SAR RAW SIGNAL DATA
34         'Raw Data': [(413, 414), '2B1', 0] # [I, Q]
35     ]

```

7.8.1.5.3.2 数据读取与解析

采用本书作者开发的 iprs 工具包可以读取 ERS 原始 SAR 回波数据，使用 iprs 读取 *E2_84690_STD_L0_F137.zip* 中的 *E2_84690_STD_L0_F137.000.raw* 文件，这里给出读取的 SAR 数据文件文件描述记录，参见 代码 7.4.

代码 7.4: Sar Data File File Descriptor Record in sar raw data file of product E2_84690_STD_L0_F137.

```

1 Record sequence number [(1, 4), '1B4', [1]]
2 1-st record sub-type code [(5, 5), '1B1', [63]]
3 Record type code [(6, 6), '1B1', [192]]
4 2-nd record sub-type code [(7, 7), '1B1', [18]]
5 3-rd record sub-type code [(8, 8), '1B1', [18]]
6 Length of this record [(9, 12), '1B4', [11644]]
7 ASCII/EBCDIC flag [(13, 14), '1A2', ['A']]
8 1Blanks [(15, 16), '1A2', [' ']]
9 Format control document ID [(17, 28), '1A12', ['CEOS-SAR-CCT']]
10 Format control document revision level [(29, 30), '1A2', ['B']]
11 File design descriptor revision letter [(31, 32), '1A2', ['B']]
12 Generating software release and revision level [(33, 44), '1A12', ['SKY 5.4.8
    ↪']]

```

(下页继续)

(续上页)

```

13 File number [(45, 48), '1I4', [2]]
14 File name [(49, 64), '1A16', ['ERS2.SAR.RAWIMGY']]
15 Record sequence and location type flag [(65, 68), '1A4', ['FSEQ']]
16 Sequence number location [(69, 76), '1I8', [1]]
17 Sequence number field length [(77, 80), '1I4', [4]]
18 Record code and location type flag [(81, 84), '1A4', ['FTYP']]
19 Record code location [(85, 92), '1I8', [5]]
20 Record code field length [(93, 96), '1I4', [4]]
21 Record length and location type flag [(97, 100), '1A4', ['FLGT']]
22 Record length location [(101, 108), '1I8', [9]]
23 Record length field length [(109, 112), '1I4', [4]]
24 Reserved1 [(113, 113), '1I1', [[]]]
25 Reserved4 [(116, 116), '1I1', [[]]]
26 Reserved segment [(117, 180), '1A64', [
    ↪
    '']]
27 Number of SAR DATA records (nominal) [(181, 186), '1I6', [28603]]
28 SAR DATA record length (bytes) [(187, 192), '1I6', [11644]]
29 Reserved1 (blanks) [(193, 216), '1A4', [' ']]
30 Number of bits per sample [(217, 220), '1I4', [16]]
31 Number of samples per data group (or pixels) [(221, 224), '1I4', [1]]
32 Number of bytes per data group (or pixel) [(225, 228), '1I4', [2]]
33 Justification and order of samples within data group [(229, 232), '1A4', [' ' '']]
34 Number of SAR channels in this file [(233, 236), '1I4', [1]]
35 Number of lines per data set (nominal) [(237, 244), '1I8', [28603]]
36 Number of left border pixels per line [(245, 248), '1I4', [0]]
37 Total number of data groups per line per SAR channel [(249, 256), '1I8', [5616]]
38 Number of right border pixels per line [(257, 260), '1I4', [0]]
39 Number of top border lines [(261, 264), '1I4', [0]]
40 Number of bottom border lines [(265, 268), '1I4', [0]]
41 Interleaving indicator [(269, 272), '1A4', ['BSQ ']]
42 Number of physical records per line [(273, 274), '1I2', [1]]
43 Number of physical records per multi-channel line [(275, 276), '1I2', [[]]]
44 Number of bytes of prefix data per record [(277, 280), '1I4', [412]]
45 Number of bytes of SAR data (or pixel data) per record (nominal) [(281, 288), '1I8
    ↪', [11232]]
46 Number of bytes of suffix data per record [(289, 292), '1I4', [0]]
47 Reserved2 [(293, 340), '1A48', [
    ↪
    ' ' 1 4PB 37 2PB 33 4PB 9 4PB 17 4PB
    ↪ ']]
48 Blanks [(341, 368), '1A28', [
    ↪
    ' ']]
49 Reserved3 [(369, 400), '1A32', [
    ↪
    ' ']]
50 SAR Data format type identifier [(401, 428), '1A28', ['COMPLEX SIGNED INTEGER*2
    ↪ ']]
51 SAR Data format type code [(429, 432), '1A4', ['CIS2']]
52 Number of left fill bits within pixel [(433, 436), '1I4', [0]]
53 Number of right fill bits within pixel [(437, 440), '1I4', [0]]
54 Maximum data range of pixel (max-min value for I and Q) [(441, 448), '1I8', [
    ↪ [65535]]]

```

由 [代码 7.4](#) 知, 数据存储格式为 CEOS-SAR-CCT, 该数据共 28603 行脉冲数据, 每行为一次脉冲回波, 一个脉冲含 5616 个距离单元, 每个像素含实部(I)虚部(Q)共 2 个字节, 故每行脉冲含 11232 个字节. 该记录长度为 11644 字节, 后面的 SAR 数据记录长度亦为 11644 字节.

7.8.1.5.4 单视复数数据读取

[代码 7.5](#) 给出了 ERS 合成孔径雷达单视复数数据文件的记录格式, 按照该格式读取数据即可, 本书作者开发的 iprs 工具包可以读取 ERS 单视复数数据, 读取函数为 `read_ers_sar_slc()`, iprs 中定义的用于读取处理后的 SAR 数据映射字典如下:

```
SarDataFileProcessedDataRecordERS = {
    'Record sequence number': [(1, 4), '1B4', 0],
    '1-st record sub-type code': [(5, 5), '1B1', 0],
    'Record type code': [(6, 6), '1B1', 0],
    '2-nd record sub-type code': [(7, 7), '1B1', 0],
    '3-rd record sub-type code': [(8, 8), '1B1', 0],
    'Length of this record': [(9, 12), '1B4', 0],
    # SAR PROCESSED DATA
    'Processed Data': [(13, 16), '1C4', 0] # '2F2' for [real, imag], or '1C4' for
    ↪real + 1j*imag
}
```

警告: 需要注意的是, 下载到的单视复数数据可能仅仅是幅度图, 这种情况下, 读取到的 SAR Data format type code 值可能为 IU1 即一字节无符号整数, 而不是 [代码 7.5](#) 中所列的 CI*4. 此时将映射表中的 'Processed Data' 段的类型改为 '1B' 即可.

[代码 7.5](#): Sar Data File File Descriptor Record in sar raw data file of product E2_84690_STD_L0_F137.

```

1 ===DATA SET FILE FORMAT DEFINITION
2
3 ---1 SAR DATA FILE - FILE DESCRIPTOR RECORD (FIXED SEGMENT)
4
5 FIELD    BYTES    FORMAT    DESCRIPTION
6   ↪CONTENT
7
8   1      1-4      B4        Record sequence number
9   ↪(1)
10  2       5       B1        1-st record sub-type code
11  ↪(63)
12  3       6       B1        Record type code
13  ↪(192)
14  4       7       B1        2-nd record sub-type code
15  ↪(18)
16  5       8       B1        3-rd record sub-type code
17  ↪(18)
```

(下页继续)

(续上页)

11	6	9-12	B4	Length of this record	
	↪(10012)				—
12	7	13-14	A2	ASCII/EBCDIC flag	
	↪ A\$				—
13	8	15-16	A2	blanks	
	↪ \$S				—
14	9	17-28	A12	Format control document ID for this data file	
				format CEOS-SAR-CCT	
15	10	29-30	A2	Format control document revision level	
	↪ \$B				—
16	11	31-32	A2	File design descriptor revision letter	
	↪ \$B				—
17	12	33-44	A12	Generating software release and revision level	<..
	↪....>				
18	13	45-48	I4	File number	
	↪ \$S\$2				—
19	14	49-64	A16	File name	ERS1.SAR.
	↪ SLCIMGY				
20	15	65-68	A4	Record sequence and location type flag	
	↪ FSEQ				—
21	16	69-76	I8	Sequence number location	\$\$\$
	↪\$\$\$\$1				
22	17	77-80	I4	Sequence number field length	
	↪ \$\$\$4				—
23	18	81-84	A4	Record code and location type flag	
	↪ FTYP				—
24	19	85-92	I8	Record code location	\$\$\$
	↪\$\$\$\$5				
25	20	93-96	I4	Record code field length	
	↪ \$\$\$4				—
26	21	97-100	A4	Record length and location type flag	
	↪ FLGT				—
27	22	101-108	I8	Record length location	\$\$\$
	↪\$\$\$\$9				
28	23	109-112	I4	Record length field length	
	↪ \$\$\$4				—
29	24	113	A1	Reserved	<\$.
	↪...\$>				
30	25	114	A1	Reserved	<\$.
	↪...\$>				
31	26	115	A1	Reserved	<\$.
	↪...\$>				
32	27	116	A1	Reserved	<\$.
	↪...\$>				
33	28	117-180	A64	Reserved segment	<\$.
	↪...\$>				
34					
35					

(下页继续)

(续上页)

```

36
37 ---2 SAR DATA IMAGERY OPTIONS FILE FILE DESCRIPTOR RECORD VARIABLE SEGMENT
38
39 FIELD BYTES FORMAT DESCRIPTION
40 ↳CONTENT
41 29 181-186 I6 Number of SAR DATA records (nominal)
42 ↳$15000
43 30 187-192 I6 SAR DATA record length (bytes)
44 ↳$10012
45 31 193-216 A24 Reserved (blanks) <$. .
46 ↳...$>
47
48 SAMPLE GROUP DATA
49
50 32 217-220 I4 Number of bits per sample
51 ↳$$32
52 33 221-224 I4 Number of samples per data group (or pixels)
53 ↳$$$1
54 34 225-228 I4 Number of bytes per data group (or pixel)
55 ↳$$$4
56 35 229-232 A4 Justification and order of samples within data group <
57 ↳$....$>
58
59 SAR RELATED DATA IN THE RECORD
60
61 36 233-236 I4 Number of SAR channels in this file
62 ↳$$$1
63 37 237-244 I8 Number of lines per data set (minimum) $$
64 ↳$15000
65 38 245-248 I4 Number of left border pixels per line
66 ↳$$$0
67 39 249-256 I8 Total number of data groups per line per SAR channel $$$
68 ↳$2500
69 40 257-260 I4 Number of right border pixels per line
70 ↳$$$0
71 41 261-264 I4 Number of top border lines
72 ↳$$$0
73 42 265-268 I4 Number of bottom border lines
74 ↳$$$0
75 43 269-272 A4 Interleaving indicator
76 ↳BSQ$
77
78 RECORD DATA IN THE FILE
79
80 44 273-274 I2 Number of physical records per line
81 ↳ $1

```

(下页继续)

(续上页)

```

66    45      275-276  I2      Number of physical records per multi-channel line      -
↪ $S$
67    46      277-280  I4      Number of bytes of prefix data per record
↪ $S$S$0
68    47      281-288  I8      Number of bytes of SAR data (or pixel data)      $$S
↪ $10000
69                      per record (nominal)
70    48      289-292  I4      Number of bytes of suffix data per record
↪ $S$S$0
71    49-55   293-340  A48     reserved                                <$. .
↪ ...$>
72    56      341-368  A28     blanks                                 <$. .
↪ ...$>
73    57-60   369-400  A32     reserved                                <$. .
↪ ...$>
74    61      401-428  A28     SAR Data format type identifier      COMPLEX
↪ $INTEGER$..$
75    62      429-432  A4      SAR Data format type code          -
↪ CI*4
76    63      433-436  I4      Number of left fill bits within pixel      -
↪ $S$S$0
77    64      437-440  I4      Number of right fill bits within pixel      -
↪ $S$S$0
78    65      441-448  I8      Maximum data range of pixel        $$S
↪ $65535
79    66      449-EOR   A15564  spare                                <$. .
↪ ...$>

80
81 ===DATA RECORD

82
83 ---1 IMAGERY OPTIONS FILE - PROCESSED DATA RECORD

84
85 FIELD    BYTES      FORMAT      DESCRIPTION      -
↪ CONTENT
86
87 1       1-4        B4        Record sequence number      -
↪ (n)
88 2       5          B1        1-st record sub-type code      -
↪ (50)
89 3       6          B1        Record type code          -
↪ (11)
90 4       7          B1        2-nd record sub-type code      -
↪ (31)
91 5       8          B1        3-rd record sub-type code      -
↪ (20)
92 6       9-12       B4        Length of this record      (nominal)      -
↪ (10012)

```

(下页继续)

(续上页)

```

93      7      13-16      C4      first sample of image line      -
94      ↳(n)
95      8      17-20      C4      second sample of image line     -
96      ↳(n)
97      .      .....      .      .....      ...
98      ↳.....
5006    10009-10012 C4      last sample of image line      -
99      ↳(n)

```

读取的 *E2_84690_STD_F137* 文件中的数据大小为 9182×9182 , 实际有效数据大小为 9182×7833 , 从 7834 到 9182 列为填补的无效数据. 该数据实际上为幅度图像数据, 而不是复数数据, 由于数据较大, 故进行 8 倍降采样显示, 显示结果如 *_fig-E2_84690_STD_F137* 所示, 其中左图未去除无效右边界像素, 右图去除了无效右边界的像素. 此外, 可以看到成像算法处理完后的数据对比度较低, 可以通过增强算法进行调整.

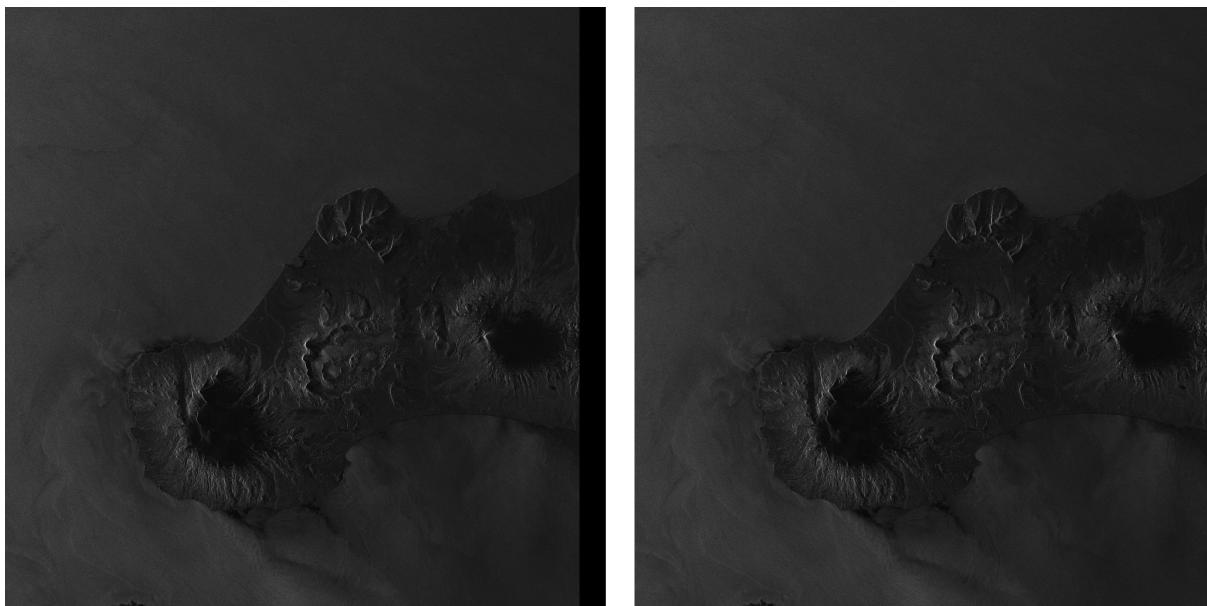


图 7.154: 数据 *E2_84690_STD_F137* 中的图像数据. (左) 未去除无效右边界像素, (右) 去除无效右边界的像素.

7.8.1.5.5 原始数据成像

本节给出 Chirp Scaling Algorithm(CSA) 在以上两景 ERS 数据上的成像结果, 有关 CSA 算法, 参见[调频变标成像](#) (页 472) 小节. 由于原始数据较大, 且方位向采样点数一般远大于距离向采样点数, 故这里仅给出降采样的成像结果图以及多视处理后的结果图. 对于数据 *E2_84690_STD_L0_F137*, 其方位向采样点数为 $N_a = 28603$, 距离向采样点数为 $N_r = 5616$, 成像后的单视复数图像矩阵大小为 28603×5616 , 将其降采样至 1024×1000 , 得到 图 7.156 所示结果. 图 7.157 显示的是方位向多视处理结果, 其中视数为 4, 处理得到 7150×5616 的数据, 将其降采样至 1024×1000 显示. 图 7.155 显示的是下载的数据 *E2_84690_STD_L0_F137* 的单视复数图像, 对比可以发现, CSA 成像结果正确.

同样地, 图 7.158 显示的是下载的数据 *E2_84686_STD_L0_F203* 的单视复数图像, *E2_84686_STD_L0_F203*



图 7.155: Single Look Complex image of ERS SAR product (E2_84690_STD_SLC_F137)



图 7.156: Single look imaging result of ERS SAR product (E2_84690_STD_L0_F137) using ChirpScaling Algorithm



图 7.157: Multiple look (4) imaging result of ERS SAR product (E2_84690_STD_L0_F137) using ChirpScaling Algorithm

数据的方位向采样点数为 $N_a = 28659$, 距离向采样点数为 $N_r = 5616$, 成像后的单视复数图像矩阵大小为 28659×5616 , 将其降采样至 1024×1000 , 得到 图 7.159 所示结果. 图 7.160 显示的是方位向多视处理结果, 其中视数为 4, 处理得到 7150×5616 的数据, 将其降采样至 1024×1000 显示



图 7.158: Single Look Complex image of ERS SAR product (E2_84686_STD_SLC_F203)

注意: 需要注意的是, 使用 CSA 成像后得到的是单视复数图像数据 I , 经过多视处理后得到降噪后的多视数据, 图 7.156, 图 7.157, 图 7.159, 图 7.160 中显示的结果, 为成像复数结果的幅度图, 未进行 RCS 换算, 由此得到的图像往往对比度较低, 不能准确还原目标的 RCS, 这里采用对数增强与截断操作增强图像 (参见 SubSection-LogContrastEnhancementDigitalImageSignalProcessing 小节). 此外, 仔细观察可以发现, 下载的单视复数结果图像与本节 CSA 算法成像结果略有不同, 后者场景区域稍大一些, 这是由于在进行 CSA 成像时, 未丢弃非完全成像边界区域.



图 7.159: Single look imaging result of ERS SAR product (E2_84686_STD_L0_F203) using ChirpScaling Algorithm



图 7.160: Multiple look (4) imaging result of ERS SAR product (E2_84686_STD_L0_F203) using ChirpScaling Algorithm

7.8.1.6 Sentinel 系列产品介绍

7.8.1.6.1 Sentinel-1 产品

从 [SCIHUB](https://scihub.copernicus.eu/) (<https://scihub.copernicus.eu/>) 和 [ASF](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 均可以下载 Sentinel 系列卫星产品数据, 有关 Sentinel-1 SAR 的信息可在 [这里](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar) (<https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar>) 查看.

7.8.1.6.1.1 Sentinel-1 雷达参数与工作模式

Sentinel-1 合成孔径雷达参数如 表 7.6

表 7.6: Sentinel-1 平台参数

Centre frequency	5.405 GHz (corresponding to a wavelength of ~5.5465763cm cm)
Bandwidth	0-100 MHz (programmable)
Polarisation	HH+HV, VV+VH, VV, HH
Incidence angle range	20°- 46°
Look direction	right
Antenna type	Slotted waveguide radiators
Antenna size	12.3 m x 0.821 m
Antenna mass	880 kg (representing 40% of the total launch mass)
Azimuth beam width	0.23°
Azimuth beam steering range	-0.9° to +0.9°
Elevation beam width	3.43°
Elevation beam steering range	-13.0° to +12.3°
RF Peak power	<ul style="list-style-type: none"> • 4.368 kW, - 4.075 kW (IW, dual polarisations)
Pulse width	5-100 μs (programmable)
Transmit duty cycle	Max 12%, SM 8.5%, IW 9%, EW 5%, WV 0.8%
Receiver noise figure at module input	3 dB
Maximum range bandwidth	100 MHz
PRF (Pulse Repetition Frequency)	1 000 - 3 000 Hz (programmable)
Data compression	FDBAQ (Flexible Dynamic Block Adaptive Quantization)
ADC sampling frequency	300 MHz (real sampling)
Data quantisation	10 bit
Total instrument mass (including antenna)	945 kg
Attitude steering	Zero-Doppler steering and roll steering

7.8.1.6.1.2 Sentinel-1 产品格式

7.8.1.6.1.3 传感器类型与产品类型

Sentinel-1 SAR 具有多种传感器模式和数据产品，列举如下：

- 传感器模式 (Sensor Mode)

- Level-0 and Level-1

- * 条带式 (Strip Map, SM): 80 km swath, 5 x 5 m spatial resolution
- * 干涉宽幅式 (Interferometric Wide Swath, IW): 250 km swath, 5 x 20 m spatial resolution
- * 超宽幅式 (Extra Wide Swath, EW): 400 km swath, 20 x 40 m spatial resolution

- Level-2

- * Wave (WV)
- * 干涉宽幅宽式 (Interferometric Wide Swath, IW)
- * 超宽幅宽式 (Extra Wide Swath, EW)

- 产品类型 (Product Type)

- Level-0

- * 原始回波数据格式 RAW

- Level-1

- * 单视复数格式 (Single Look Complex, SLC)
- * 地距探测格式 Ground Range Detected (GRD)

- Level-2

- * (Ocean, OCN) 包含 3 种成份： Ocean Swell spectra (OSW), Ocean Wind Fields (OWI), Surface Radial Velocities (RVL).

提示：

- Sentinel PDF Documentation (<https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/document-library>)
 - 用户手册参见 User Guides (<https://earth.esa.int/web/sentinel/user-guides>)
 - 技术手册参见 Sentinel Technical Guides (<https://earth.esa.int/web/sentinel/sentinel-technical-guides>)
 - 传感器类型，各级产品所采用的算法，以及各级产品数据格式参见 [这里](https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-1-sar/products-algorithms) (<https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-1-sar/products-algorithms>).
 - 能量分布参见 Willkommen bei den Energy Charts (<https://www.energy-charts.de/>)
-

Acq. Mode	Product Type	Resolution Class	Resolution Rng x Azi [m]	Pixel Spacing Rng x Azi [m]	Num Looks Rng x Azi	ENL
SM	SLC		1.7x4.3 to 3.6x4.9	1.5x3.6 to 3.1x4.1	1x1	1
	GRD	FR	9x9	3.5x3.5	2x2	3.7
		HR	23x23	10x10	6x6	29.7
		MR	84x84	40x40	22x22	398.4
IW	SLC		2.7x22 to 3.5x22	2.3x14.1	1x1	1
	GRD	HR	20x22	10x10	5x1	4.4
		MR	88x87	40x40	22x5	81.8
EW	SLC		7.9x43 to 15x43	5.9x19.9	1x1	1
	GRD	HR	50x50	25x25	3x1	2.8
		MR	93x87	40x40	6x2	10.7
WV	SLC		2.0x4.8 3.1x4.8	1.7x4.1 2.7x4.1	1x1	1
	GRD	MR	52x51	25x25	13x13	123.7

图 7.161: Sentinel1 SAR Level 1 级别产品

7.8.1.6.1.4 Sentinel 产品数据命名格式

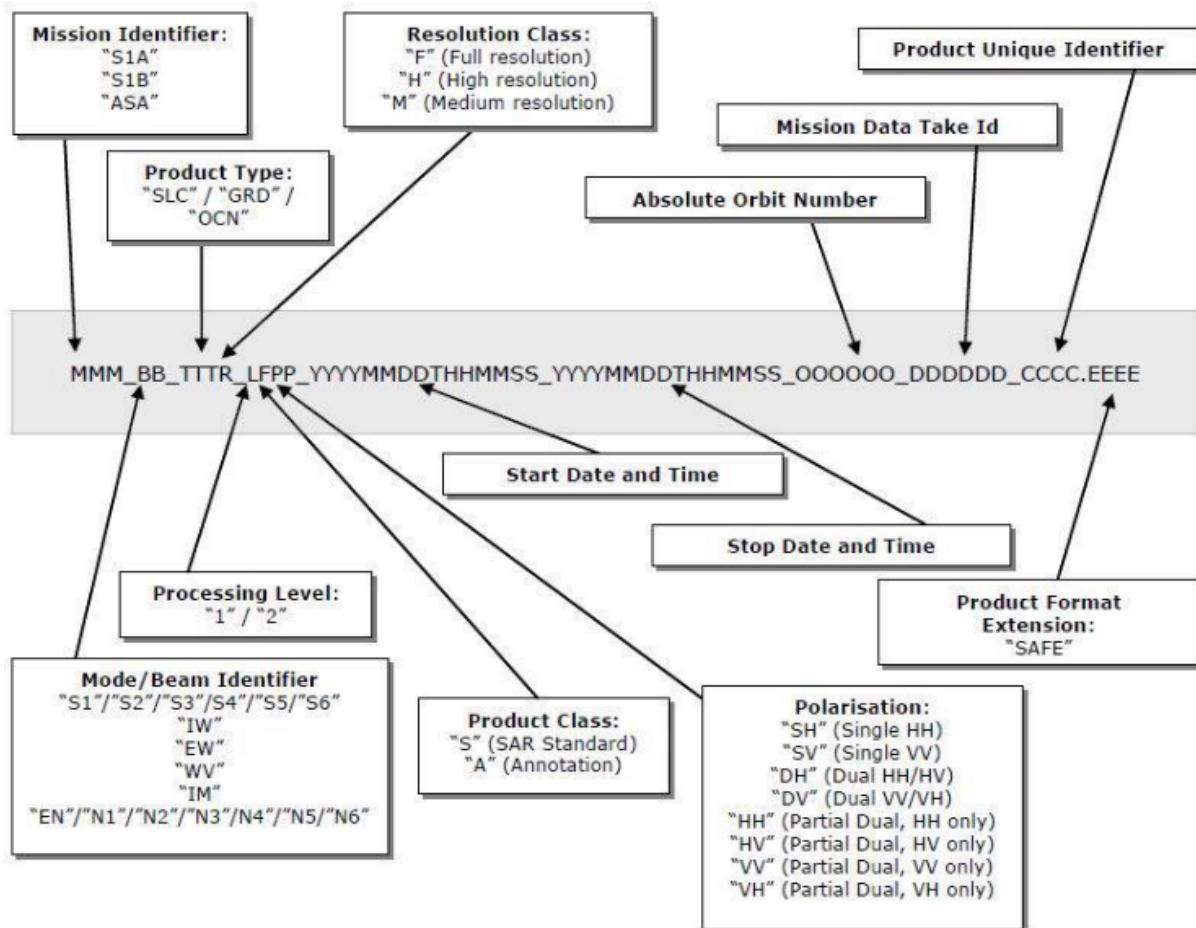


图 7.162: Sentinel 产品命名格式

举例如下

- S1A_S3_SLC__1SDV_20191202T094416_20191202T094445_030167_037285_9342 : Sentinel1A_S3_ 单视复数 _L1SAR 双 V 极化 (VV/VH)_ 起始时间 _ 结束时间 _ 绝对轨道号 _ 任务数据获取 ID_ 产品特有 ID
- S1A_S3_RAW__0SDV_20191202T094415_20191202T094446_030167_037285_0156 : Sentinel1A_S3_ 原始数据 _L0SAR 双 V 极化 (VV/VH)_ 起始时间 _ 结束时间 _ 绝对轨道号 _ 任务数据获取 ID_ 产品特有 ID

7.8.1.6.1.5 数据下载与处理

下载数据需要注册账号。从 scihub 上下载数据可根据 [Self Registration](https://scihub.copernicus.eu/userguide/SelfRegistration) (<https://scihub.copernicus.eu/userguide/SelfRegistration>) 页面说明注册账号, 具体使用方法参见 [Overview](https://scihub.copernicus.eu/userguide/) (<https://scihub.copernicus.eu/userguide/>); 从 asf 下载数据需要注册 [EARTHDATA](https://earthdata.nasa.gov/) (<https://earthdata.nasa.gov/>) 账号. 若下载过慢, 可以使用多线程工具 (如 aria2c).

提示: 从 scihub 上下载数据时, 可通过右击设置多边形搜索区域.

7.8.1.6.1.6 SNAP

- [SNAP Wiki](https://senbox.atlassian.net/wiki/spaces/SNAP/overview) (<https://senbox.atlassian.net/wiki/spaces/SNAP/overview>) + [Using SNAP in your Python programs](https://senbox.atlassian.net/wiki/spaces/SNAP/pages/24051781/Using+SNAP+in+your+Python+programs) (<https://senbox.atlassian.net/wiki/spaces/SNAP/pages/24051781/Using+SNAP+in+your+Python+programs>)
- [SANP docs](http://step.esa.int/main/doc/) (<http://step.esa.int/main/doc/>) + [SNAP Engine 2.0 API docs](http://step.esa.int/docs/v2.0/apidoc/engine/overview-summary.html) (<http://step.esa.int/docs/v2.0/apidoc/engine/overview-summary.html>)

7.8.1.6.1.7 Sentinel1 Level0 级数据解析

Sentinel1 Level0 有四种数据产品: 标准产品、校正产品、噪声产品、标记产品; 波束模式可以为 SM, IW, EW, WV; 极化模式可以为单极化和双极化. 由于块自适应量化型 (Block Adaptive Quantisation-like, BAQ) 原始数据压缩器会引入量化噪声, 造成信噪比下降, 因而 Sentinel1 数据采用弹性动态块自适应量化 (Flexible Dynamic Block Adaptive Quantisation, FDBAQ) 技术, FDBAQ 是基于熵约束块自适应量化 (Entropy Constrained Block Adaptive Quantisation, ECBAQ) 算法做出的改进.

Sentinel 产品数据格式基于欧洲标准文件格式 (Standard Archive Format for Europe, SAFE) 定制. SENTINEL-SAFE 格式包裹了一个包含二进制格式的图像数据和 XML 格式的产品元数据 (metadata), 用法相当灵活, 足以表达各级别的 Sentinel 数据产品. 通常 Sentinel Level0 级 SAFE 格式产品包含以下文件:

- `manifest.safe`: 以 XML 格式保存一般的产品信息 (平台, 任务, 仪器, 产品历史, 时序, 轨道等等)
- `xxxxx.dat`: 观测数据组件 (若为单极化则含有一个二进制数据文件, 若为双极化则含有两个二进制数据文件), 数据以大端 (big-endian) 格式存储
- `xxxx_index.dat`: 索引数据组件 (若为单极化则含有一个二进制索引文件, 若为双极化则含有两个二进制索引文件), 索引数据包含了数据逻辑块描述 (字节位置, 时间, 大小等等), 它指向了观测数据中的子块, 使得可以快速获取子块数据.
- `xxxx_annot.dat`: 注释数据组件是二进制文件, 索引数据文件 (若为单极化则含有一个二进制索引文件, 若为双极化则含有两个二进制索引文件), 索引数据包含了数据逻辑块描述 (字节位置, 时间, 大小等等), 它指向了观测数据中的子块, 使得可以快速获取子块数据.
- `support`: 表示数据文件的支持文件夹, 以 XML 格式提供观测数据和注释数据的格式或语法信息.

如图 7.163 所示, 为双极化 RAW 格式数据, 可以看到上述文件组件, 详细的 Level0 级文件格式可参考文档 [Sentinel1 Level0 Product Format Specifications](https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/document-library/) (<https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/document-library/>)

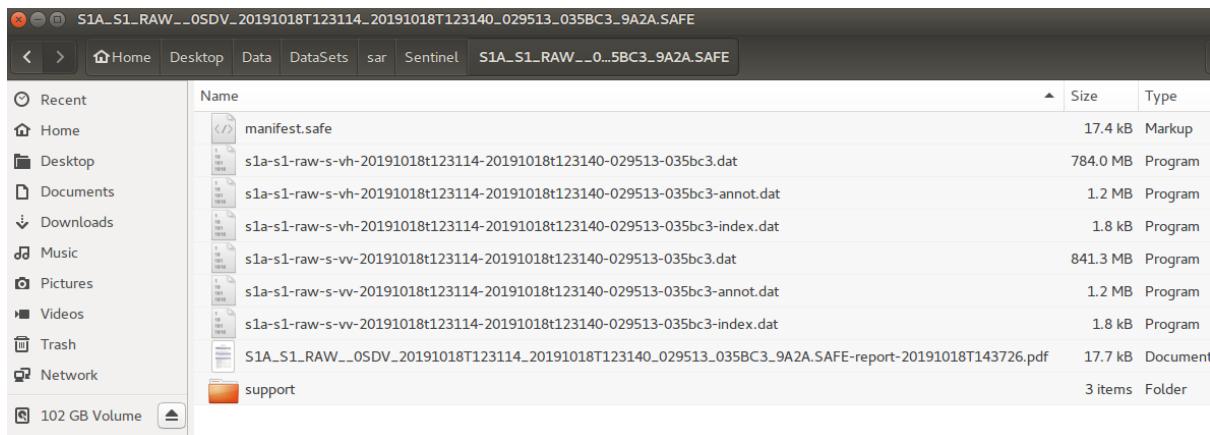


图 7.163: Sentinel1 SAR Level 0 级别产品 SAFE 文件结构示例

7.8.1.6.2 Sentinel-2 雷达

7.8.1.7 AIRSAR 产品介绍

7.8.1.7.1 AIRSAR 简介

AIRSAR (AIRborne Synthetic Aperture Radar) 是由美国国家航空航天局 (National Aeronautics and Space Administration, NASA) 的喷气推进实验室 (Jet Propulsion Laboratory, JPL) 设计和建造的一种机载合成孔径雷达。机载合成孔径雷达是一种能够穿透云层并在夜间收集数据的全天候成像工具, 更长的波长也可以穿透森林冠层和薄的沙层与积雪, 被广泛用于地形观测. AIRSAR 于 1988 年首次飞行, 并于 2004 年完成了最后一次飞行任务, 飞机以每秒 215 米的速度在平均地形上空 8 公里的高度飞行.

有关 AIRSAR 的信息可在 [这里](https://airsar.jpl.nasa.gov/index.html) (<https://airsar.jpl.nasa.gov/index.html>) 和 [这里](https://ASF.alaska.edu/data-sets/sar-data-sets/airsar/) (<https://ASF.alaska.edu/data-sets/sar-data-sets/airsar/>) 查看. 从 [ASF](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 上可以下载 AIRSAR 产品数据, 可获得的数据格式如下:

- PolSAR: 3-frequency polarimetry
- TOPSAR: C-, L-, and P-band Compressed Stokes Matrix, C-band TIFF, DEM
- ATI: Interferograms

7.8.1.7.1.1 AIRSAR 雷达参数与工作模式

表 7.7: AIRSAR 平台参数

	P-band	L-band	C-band
Flight altitude	8km	8km	8km
Flight velocity	215m/s	215m/s	215m/s
Frequency/wavelength	0.45 GHz/67 cm	1.26 GHz/23 cm	5.31 GHz/5.7 cm
Polarization	Full	Full	Full
Range Resolution	7.5 m	3.75 m	1.875 m
Swath Width (nominal)	10 km	10 km	10 km
Off-Nadir Angle (normal)	20-60°	20-60°	20-60°

AIRSAR 有三种工作模式, 分别为: 极化合成孔径雷达 (POLarimetric Synthetic Aperture Radar, POLSAR) 模式, 渐进扫描地形观测合成孔径雷达 (Terrain Observation by Progressive scans Synthetic Aperture Radar, TOPSAR) 模式, 以及沿轨干涉 (Along Track Interferometry, ATI) 模式.

POLSAR Data Operation Mode In POLSAR mode, fully polarimetric data are acquired at all three frequencies in P-, L-, C-band for 40 Mhz or 20 Mhz. The L-band also provides 80 MHz bandwidth data. Fully polarimetric means that radar waves are transmitted and received in both horizontal (H) and vertical (V) polarizations. POLSAR data are sensitive to the geometry (including vegetation) and dielectrical properties (water content) of the terrain.

TOPSAR Data Operation Mode

In TOPSAR mode, AIRSAR collects interferometric data using C- and L-band vertically-displaced antenna pairs to produce digital elevation models (DEM's). The radars which are not being used for interferometry (P-band for XTI2 or P-band and L-band for XTI1) collect quad-pol data co-registered with the C-band DEM. Interferometric data can be collected in “ping-pong” mode, where each antenna is used alternately for transmit and the effective baseline is doubled, and in “common-transmitter” mode where only one antenna is used for transmit.

ATI Data Operation Mode (Experimental)

Data collected in the along-track interferometry (ATI) mode can be used to measure ocean current velocities. Two pairs of antennas, one C-band and one L-band, separated along the body of the plane, are used to collect ATI data.

7.8.1.7.2 数据读取

7.8.1.7.3 数据成像

7.8.1.8 UAVSAR 产品介绍

7.8.1.8.1 UAVSAR 简介

UAVSAR (Uninhabited/Unmanned Aerial Vehicle Synthetic Aperture Radar) 是由美国国家航空和航天局 (National Aeronautics and Space Administration, NASA) 的喷气推进实验室 (Jet Propulsion Laboratory, JPL) 设计和建造的一种无人驾驶飞行器合成孔径雷达, 属于机载合成孔径雷达的一种, 专门被设计用于获取机载重复轨道 SAR 数据, 从而用于差分干涉测量. UAVSAR 于 2007 年 8 月发射, 服役至今 (2020 年). 有关 UAVSAR

的信息可在 [这里](https://uavstar.jpl.nasa.gov/) (<https://uavstar.jpl.nasa.gov/>) 和 [这里](https://ASF.alaska.edu/data-sets/sar-data-sets/uavstar/) (<https://ASF.alaska.edu/data-sets/sar-data-sets/uavstar/>) 查看, 从 [ASF](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) 上可以下载 UAVSAR 产品数据.

- PolSAR: MLC, Compressed Stokes Matrix, Ground Projected Complex, Pauli Decomposition
- RPI: Interferogram, Ground Projected Interferogram, Amplitude, Ground Projected Amplitude

7.8.1.8.1.1 UAVSAR 雷达参数

表 7.8: UAVSAR 平台参数

Frequency	L-band
Polarization	Full
Resolution (range)	1.8 m
Swath Width	16 km

7.8.1.8.2 工作模式与数据

AIRSAR 有两种工作模式, 分别为: 极化合成孔径雷达 (POLarimetric Synthetic Aperture Radar, POLSAR) 模式, 重复轨道干涉合成孔径雷达 (Repeat Pass Interferometry, RPI) 模式

7.8.1.8.3 数据读取

7.8.1.8.4 数据成像

7.8.2 雷达数据集

7.8.2.1 简介

7.8.2.1.1 雷达数据集概览

7.8.2.2 毫米波雷达数据集

7.8.2.2.1 The Oxford Radar RobotCar Dataset

- [The Oxford Radar RobotCar Dataset](https://robotcar-dataset.robots.ox.ac.uk/) (<https://robotcar-dataset.robots.ox.ac.uk/>)

7.8.3 雷达库与软件

7.8.3.1 简介

本节主要对当前常用的雷达信号处理库和软件进行介绍, 含毫米波雷达, 合成孔径雷达, 极化合成孔径雷达, 干涉合成孔径雷达类型的处理软件. 覆盖信号模拟生成, 目标探测, 成像与图像解译等方面.

7.8.3.1.1 雷达信号处理库与软件概览

7.8.3.2 ASF MapReady

Alaska Satellite Facility

7.8.3.2.1 简介

[MapReady](https://www.asf.alaska.edu/software-tools/#mapready) (<https://www.asf.alaska.edu/software-tools/#mapready>) 是由 ASF 开发的遥感数据处理工具, 主要用于处理各种 SAR 数据, 尤其适合处理 CEOS 格式的数据, 其源码可以在 [这里](https://github.com/asfadmin/ASF_MapReady) (https://github.com/asfadmin/ASF_MapReady) 获得. ASF MapReady 支持 ERS Level0 数据读取, 不支持 Sentinel Level0 数据读取.

注解: 由 ASF MapReady 中的源码可知, 暂不支持 Sentinel Level0 级数据, 具体参见 `src/libasf_import/import_sentinel.c` 文件中的 `import_sentinel()` 函数:

```
if (strcmp_case(productType, "RAW") == 0) asfPrintError("Product type 'RAW' currently not supported!\n");
else if (strcmp_case(productType, "SLC") == 0 || strcmp_case(productType, "GRD") == 0)
```

7.8.3.2.2 安装 ASF MapReady

7.8.3.2.2.1 Ubuntu 下配置安装

安装步骤如下:

1. 安装依赖:

```
sudo apt install gcc g++ bison flex libcurl1-dev libexif-dev libfftw3-dev
  ↪libgdal-dev libgeotiff-dev libglade2-dev libglib2.0-dev libgsl-dev libgtk2.0-
  ↪dev libjpeg-dev libpng-dev libproj-dev libshp-dev libtiff5-dev libxml2-dev
```

2. 配置安装目录: `./configure --prefix=your_installation_path`, 注意更改自己的安装目录, 如 `/mnt/e/sfw/sar/ASF_MapReady`
3. 构建: `make` 或多线程构建 `make -j16`
4. 安装: `make install`

5. 添加环境变量 `export PATH=your_installation_path/bin:$PATH`

安装完成后, 在终端输入 `ASF_mapready --help` 可查看帮助并验证安装是否成功, 若安装成功, 则可以看到如下提示::

```
:~$ ASF_mapready --help

Tool name:
  ASF_mapready

Usage:
  ASF_mapready [-create] [-input <inFile>] [-output <outFile>]
                 [-tmpdir <dir>] [-log <logFile>] [-quiet] [-license]
                 [-version] [-help]
                 <config_file>

Description:
  This program can ingest level one CEOS and GeoTIFF format data, calibrate
  it to various radiometries, perform polarimetric decompositions, perform
  Faraday Rotation correction, perform terrain correction, geocode it, and
  then export it to a variety of graphics file formats. The user is able to
  control how ASF_mapready dictates the processing flow by creating a
  configuration file, which must then be edited, which is fed into
  ASF_mapready when it is called.

:
```

7.8.3.2.3 ASF MapReady 源码分析

7.8.3.2.3.1 数据读取

```
ASF_view --> read --> read_ceos/envi/seasat/terrasar/uavstar/tiff...
```

通过分析源码可知, `ImageInfo` 结构体是和读取显示图像相关的结构体, 在 `ASF_view.h` 中定义了 `ImageInfo` 指针类型的两个变量 (`curr`, `mask`) 和一个数组 (`image_info`)

```
// Can hold five images
#define MAX_IMAGES 5
extern ImageInfo image_info[MAX_IMAGES];
// "curr" always points to the currently being displayed image info
extern ImageInfo *curr;
extern ImageInfo *mask;
extern int current_image_info_index;
extern int n_images_loaded;
```

其中, 数组 `image_info` 存储多个图像 `ImageInfo` 结构体, `curr` 指针总是指向当前要显示的图像信息。`ImageInfo` 结构体中存储了待读取和显示图像的所有信息, 可以使用 `cache.c` 文件中的 `get_pixel()` 函数读取。`ImageInfo` 结构体成员如下.

```
struct ImageInfo

typedef struct { int nl, ns; // number of lines(azimuth), number of samples(range)
    meta_parameters *meta; // meta parameters
    CachedImage *data_ci; //
    BandConfig band_cfg;
    ImageStats stats;
    ImageStatsRGB stats_r;
    ImageStatsRGB stats_g;
    ImageStatsRGB stats_b;
    char *filename;
    char *data_name;
    char *meta_name;
} ImageInfo;
```

其中 `CachedImage` 也是一个结构体, 其中存储着待读取图像的信息以及读取方法.

```
struct CachedImage

typedef struct { // Here is the ImageCache stuff. The global ImageCache that holds the loaded image is
    "data_ci" . This is all private data.

    int nl, ns; // Image dimensions.

    ClientInterface *client; // pointers to data read implementations

    int n_tiles; // Number of tiles in memory

    int reached_max_tiles; // Have we loaded as many tiles as we can?

    int rows_per_tile; // Number of rows in each tile

    int entire_image_fits; // TRUE if we can load the entire image

    int *rowstarts; // Row numbers starting each tile

    unsigned char **cache; // Cached values (floats, unsigned chars ...)

    int *access_counts; // Updated when a tile is accessed

    int n_access; // used to find oldest tile

    ssv_data_type_t data_type; // type of data we have

    meta_parameters *meta; // metadata -don't own this pointer

    ImageStats *stats; // not owned by us, not populated by us

    ImageStatsRGB *stats_r; // not owned by us, not populated by us

    ImageStatsRGB *stats_g; // not owned by us, not populated by us

    ImageStatsRGB *stats_b; // not owned by us, not populated by us
```

```
    } CachedImage;
```

其中 ClientInterface 也是一个结构体, 其中存储着待读取图像的方法.

```
struct ClientInterface
typedef struct {

    ReadClientFn *read_fn;

    ThumbFn *thumb_fn;

    FreeFn *free_fn;

    void *read_client_info;

    ssv_data_type_t data_type;

    int require_full_load;

} ClientInterface;
```

如对于 CEOS 格式, read_fn 指向函数 read_ceos_client(), 其参数如下:

```
int read_ceos_client (int row_start, int n_rows_to_get, void *dest_void, void *read_client_info,
                      meta_parameters *meta, int data_type)
• row_start 读取起始行 (azimuth)
• n_rows_to_get 读取行数 (azimuth)
• dest_void 输出缓存指针
• read_client_info 元信息指针
• meta 元信息指针
• data_type 数据类型
```

read_fn 指针指向的函数被 cache.c 文件中的 get_pixel() 函数调用, 格式如下:

```
self->client->read_fn(rs, rows_to_get, (void*)(self->cache[spot]),
                      self->client->read_client_info, self->meta, self->client->data_type);
```

函数 get_pixel() 入口参数如下

```
static unsigned char *get_pixel (CachedImage (页 586) *self, int line, int samp)
• self CachedImage 指针
• line 像素所在行
• samp 像素所在列
```

函数 get_pixel() 被函数 cached_image_get_pixel() 调用, 函数 cached_image_get_pixel() 被函数 cached_image_new_from_file() 调用, 函数 cached_image_new_from_file() 又被函数 read_file() 调用. 总结调用如下

注解: ASF MapReady 读取数据函数调用顺序为: main() -> on_new_button_clicked() -> new_file() -> load_file() -> load_file_banded() -> load_file_banded_imp()

> `read_file()` → `cached_image_new_from_file()` → `cached_image_get_pixel()` → `get_pixel()` → `read_fn` 指向的函数 (如 `read_ceos_client()`).

7.8.3.2.4 ASF MapReady 应用举例

7.8.3.2.4.1 数据导入

在终端输入 `ASF_import --help` 可查看帮助,

7.8.3.2.4.2 导出 ERS2 Level0 级数据

7.8.3.3 SAR Training Processor

7.8.3.3.1 基础教程

7.8.3.3.1.1 源码安装

在 `ASF_Mapready/src/stp` 目录下含 `STP` 源码, 修改该目录下的 `Makefile` 文件, 添加如下路径, 注意修改为自己的路径::

```
BINDIR=/mnt/e/sfw/sar/ASF_MapReady/bin/  
SHAREDIR=/mnt/e/sfw/sar/ASF_MapReady/share/asf_tools/mapready/
```

然后在 `ASF_Mapready` 源码目录直接执行如下命令安装:

```
cd ./src/stp/  
make
```

安装完成后输入 `stp` 可以启动 `STP` 软件.

7.8.3.3.1.2 源码简析

成像处理: `ardop()` → `processPatch()`

7.8.3.3.2 问题与解决

7.8.3.3.2.1 使用

7.8.3.3.2.2 内存溢出

若安装完, 在导入数据时提示如下内存溢出错误信息, 则可判定程序有内存溢出现象, 如在导入 CEOS 格式的数据时提示如下错误信息, 后经排查是 `import_ceos_raw()` 函数中的代码导致的 `strcpy(meta->sar->polarization, meta->general->bands);, meta_sar->polarization` 是一个

大小为 15 的数组, 理论上已经够用, 而从文件中读出的 meta->general->bands 信息实际超过了此大小, 如多了很多空格, 从而导致内存溢出, 适当增加 polarization 的大小即可.

```
*** buffer overflow detected ***: stp terminated
=====
Backtrace:
/lib/x86_64-linux-gnu/libc.so.6(+0x777e5)[0x7f307f5797e5]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x5c)[0x7f307f61b15c]
/lib/x86_64-linux-gnu/libc.so.6(+0x117160)[0x7f307f619160]
/lib/x86_64-linux-gnu/libc.so.6(+0x1164b2)[0x7f307f6184b2]
```

7.8.3.4 STEP 与 SNAP

7.8.3.4.1 简介

STEP (<http://step.esa.int/main/>) (Scientific Toolbox Exploitation Platform, STEP) 是欧洲航空局 (ESA) 研发的开源科学工具箱开发平台, SNAP (<http://step.esa.int/main/toolboxes/snap/>) (SentiNel Application Platform, SNAP) 是其中用于处理哨兵系列数据的开源应用软件平台, 含 Windows, Mac OS X 和 Linux 三种操作系统下的软件版本, 支持 RADARSAT1-2, Sentinel1-3 等卫星产品.

7.8.3.4.2 环境配置

7.8.3.4.2.1 安装

7.8.3.4.2.2 源码构建安装

SNAP 开源源码目录为 [senbox.org](https://github.com/senbox-org/) (<https://github.com/senbox-org/>), 其中有一些 [例子](https://github.com/senbox-org/snap-examples) (<https://github.com/senbox-org/snap-examples>) 可供参考.

7.8.3.4.2.3 安装包安装

各版本软件及 Sentinel1-3, SMOS, Radarsat2, PROBA-V 工具箱可以在 [snap download](http://step.esa.int/main/download/snap-download/) (<http://step.esa.int/main/download/snap-download/>) 下载. 本节以 Linux 版本为例, 介绍 SNAP 的安装过程.

终端执行 `./esa-snap_all_unix_7_0.sh` 命令打开安装设置界面, 选择安装路径后, 继续选择需要安装的组件 (Sentinel2-3, SMOS, Radarsat2, PROBA-V 工具箱).

接着按提示完成安装即可.

提示: 为了后期可以在 Python 中使用 SNAP, 在 SNAP 的安装设置过程中, 需要配置 Python, 参见配置 Python 以使用 snappy (页 591) 小节.

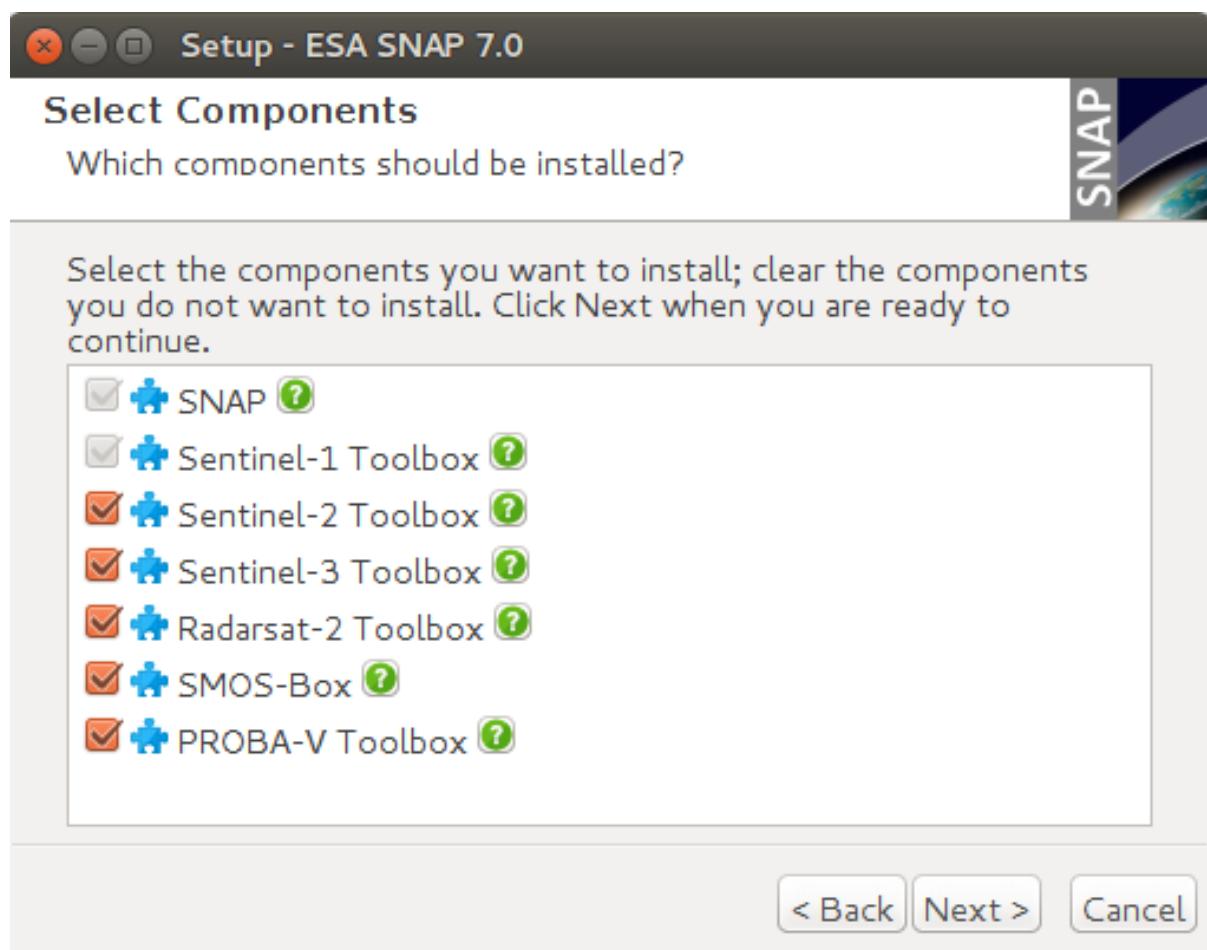


图 7.164: 在 ESA SNAP 安装界面可以选择是否安装 Sentinel2-3, SMOS, Radarsat2, PROBA-V 工具箱.

7.8.3.4.3 Level1 数据处理示例

7.8.3.4.4 Python 与 SNAP

7.8.3.4.4.1 简介

通过 SNAP 提供的 snappy 接口, 可以在 Python 中使用 SNAP API, 本节主要介绍, 如何进行 Python 与 SNAP 的交互式开发. 可以通过以下两种方式, 既可以在 Python 中调用 SNAP 函数 (Java API), 又可以扩展基于 Python 的 SNAP 插件:

- 使用标准 Python (CPython) 安装
- 使用与 SNAP 捆绑在一起的 Python 的 Jython 纯 Java 实现

如果你不准备开发桌面 GUI 环境并且想使用 Python 的科学计算库, 那么推荐使用第一种方式, 反之推荐第二种方式.

提示: 可参考 [SNAP Wiki](https://senbox.atlassian.net/wiki/spaces/SNAP) (<https://senbox.atlassian.net/wiki/spaces/SNAP>) 中的教程 [Using SNAP in your Python programs](https://senbox.atlassian.net/wiki/spaces/SNAP/pages/24051781/Using+SNAP+in+your+Python+programs) (<https://senbox.atlassian.net/wiki/spaces/SNAP/pages/24051781/Using+SNAP+in+your+Python+programs>). 此功能目前仅支持 Python2.7, 3.3, 3.4.

7.8.3.4.4.2 配置 Python 以使用 snappy

最简单的方法是在安装 SNAP 的过程中配置, 需要配置 Python 可执行文件路径, 可以通过 which python2, which python3 分别查看 Python2 和 Python3 的路径, 配置好路径后, 点击 Next 安装即可.

如果安装时没有配置 Python, 也可以手动通过命令配置. 打开终端, 使用如下命令进入 SNAP 安装目录并配置 Python 环境. 其中 <python-exe-fullpath> 为 Python 可执行文件的完整路径, <snappy-installdir> 为 snappy 的安装路径, 默认为 ~/.snap/snap-python.

为了在 Python 中方便导入 snappy 包, 可以通过 sudo gedit ~/.bashrc 打开 .bashrc 文件并添加如下环境变量:

```
export PYTHONPATH=$PYTHONPATH:~/.snap/snap-python/
```

接着, 在 Python 中, 执行 from snappy import ProductIO 来测试配置是否成功. 配置不成功将会提示错误, 如 *No module named snappy*.

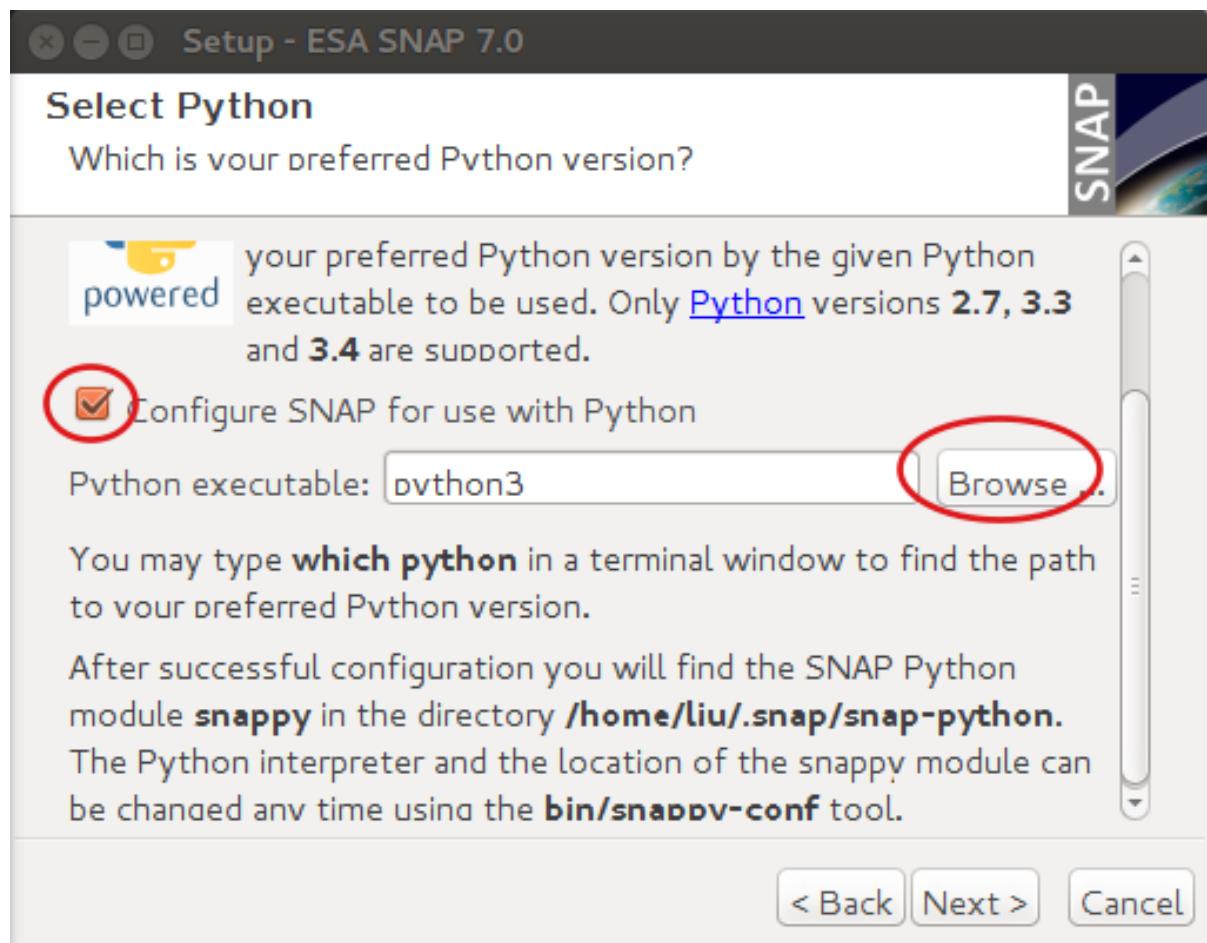


图 7.165: 在 ESA SNAP 安装过程中配置 Python 可执行文件路径.

7.8.3.4.4.3 在 Python 中调用 SNAP 函数

7.8.3.4.4.4 开发 SNAP 的 Python 处理器插件

7.8.3.4.4.5 使用 Python 接口导出原始 SAR 数据

7.8.3.4.5 总结

7.8.3.4.5.1 链接

- [Homepage of ESA](http://www.esa.int/) (<http://www.esa.int/>)
- [Homepage of STEP](http://step.esa.int/main/) (<http://step.esa.int/main/>)
- [Homepage of SNAP](http://step.esa.int/main/toolboxes/snap/) (<http://step.esa.int/main/toolboxes/snap/>)
- [Source code of SNAP](https://github.com/senbox-org/) (<https://github.com/senbox-org/>)
- [Download SNAP](http://step.esa.int/main/download/snap-download/) (<http://step.esa.int/main/download/snap-download/>)
- [SNAP Wiki](https://senbox.atlassian.net/wiki/spaces/SNAP) (<https://senbox.atlassian.net/wiki/spaces/SNAP>)
- [Documentation of SNAP](http://step.esa.int/main/doc/) (<http://step.esa.int/main/doc/>)
- [SNAP Engine API](http://step.esa.int/docs/v2.0/apidoc/engine/) (<http://step.esa.int/docs/v2.0/apidoc/engine/>)
- [forum of SNAP](https://forum.step.esa.int/) (<https://forum.step.esa.int/>) 论坛

7.8.3.5 PolSARpro

7.8.3.5.1 简介

[PolSARpro](https://www.ietr.fr/polsarpro-bio/) (<https://www.ietr.fr/polsarpro-bio/>) 由欧洲航空局 (ESA) 于 2003 年起开发维护至今, 含 Windows 和 Linux 两种操作系统下的软件版本, 支持的极化 SAR 卫星产品有: AIRSAR, ALOS2, RADARSAT2, ALOS1, GAOFEN3, RISAT, S1A_20180125, S1B_20180102.

7.8.3.5.2 基础教程

7.8.3.5.2.1 安装

从 [这里](https://www.ietr.fr/polsarpro-bio/) (<https://www.ietr.fr/polsarpro-bio/>) 选择要安装的版本, 下载软件安装包, 这里以 Linux 版为例. 解压安装包 `PoSARpro_v6.0_Biomass_Edition_Linux_Installer_20190404.zip`, 终端输入 `xterm` 命令打开 Xterm 终端. 执行“`wish PoSARpro_v6.0_Biomass_Edition_Linux_Installer.tcl`”打开安装界面, 根据提示需要安装依赖库.

代码 7.6: 安装依赖库

```
1 sudo apt install libtk-img iwidgets4 bwidget  
2 sudo apt install gcc g++ build-essential libglew-dev freeglut3-dev libfreeimage-dev  
3 sudo apt install gimp gnuplot googleearth-package imagemagick snap
```

选择好安装目录后点击 **Install** 按钮执行安装即可。安装完成后，在安装根目录可以看到如下文件夹

② ColorMap directory: 包含用户定义与修改的 PolSARpro 颜色映射文件 ② Config directory: 包含软件的所有不同配置文件 ② GUI directory: 包含所有 widget 窗口 Tcl-Tk 文件 ② Help directory: 包含 PolSARpro 帮助文件 ② License directory: 包含所有的 PolSARpro licenses 文件 ② Log directory: 包含所有会话的日志文件 ② Soft directory: 包含可使用的可执行处理文件和库 ② TechDoc directory: 包含与 PolSARpro 中使用的所有 GUI 和 C 例程相关的技术文档 ② Tmp directory: Tmp 目录在安装后为空, PolSARpro 在每个会话期间都会使用它 ② Tutorial directory: 包含 PDF 格式的 PolSARpro 教程材料

7.8.3.5.3 进阶教程

7.8.3.6 Other

7.8.3.6.1 pyroSAR

pyroSAR 是基于 Python 语言的 SAR 卫星数据处理框架，适用于大尺度数据处理，其源码可以从 [这里](https://github.com/johntruckenbrodt/pyroSAR) (<https://github.com/johntruckenbrodt/pyroSAR>) 下载，在线 API 手册可以访问 [这里](https://pyrosar.readthedocs.io/en/latest/) (<https://pyrosar.readthedocs.io/en/latest/>) 查看。

7.8.3.6.2 MATLAB SAR

MATLAB SAR 工具箱是一个基本的 MATLAB 库，可以使用 NGA 传感器独立复数数据 (Sensor Independent Complex Data, SICD) 格式读取、写入、显示和简单处理复杂的 SAR 数据。NGA 已发布该指南，以鼓励在整个国际 SAR 社区中使用 SAR 数据标准。MATLAB SAR 工具箱是对 SIX 库 (C++) 和 SarPy (Python) 的补充。

提示:

- [Matlab SAR](https://github.com/ngageoint/MATLAB_SAR) (https://github.com/ngageoint/MATLAB_SAR)
 - [SarPy](https://github.com/ngageoint/sarpy) (<https://github.com/ngageoint/sarpy>)
-

7.8.3.6.3 SarPy

7.8.3.6.4 sumo

<https://github.com/ec-europa/sumo>

7.8.3.6.5 Sentinel Level0 数据提取

- [phoqueouzygne](https://github.com/mnhrdt/phoqueouzygne) (<https://github.com/mnhrdt/phoqueouzygne>)
- [satellite-plot](https://github.com/plops/satellite-plot) (<https://github.com/plops/satellite-plot>)
- [RAT](https://github.com/birgander2/RAT) (<https://github.com/birgander2/RAT>) : ERS level0 raw
- [ASF_MapReady](https://github.com/asfadmin/ASF_MapReady) (https://github.com/asfadmin/ASF_MapReady) Sentinel Level0 , ERS level0 raw

7.8.4 参考文献

7.9 名词术语

Polarized Synthetic Aperture Radar 极化合成孔径雷达 ([Polarized Synthetic Aperture Radar](http://en.volupedia.org/wiki/Synthetic-aperture_radar) (http://en.volupedia.org/wiki/Synthetic-aperture_radar), PolSAR) 是一种雷达, 用于对二维或三维场景物体(如景观)的成像与重建. 极化 SAR 利用.

Synthetic Aperture Radar 合成孔径雷达 ([Synthetic Aperture Radar](http://en.volupedia.org/wiki/Synthetic-aperture_radar) (http://en.volupedia.org/wiki/Synthetic-aperture_radar), SAR) 是一种雷达, 用于对二维或三维场景物体(如景观)的成像与重建. SAR 利用雷达天线在目标区域上的运动来提供比传统波束扫描式雷达更高的空间分辨率.

CHAPTER 8

第八卷医学信号处理

8.1 核磁共振成像

8.1.1 核磁共振简介

核磁共振成像 (*Nuclear Magnetic Resonance Imaging*, NMRI) 也称磁共振成像 (*Magnetic Resonance Imaging*, MRI), 根据核磁共振 (Nuclear Magnetic Resonance, NMR) 也称磁共振 (Magnetic Resonance) 原理成像.

8.1.1.1 资源

- [The Basics of MRI](http://www.cis.rit.edu/htbooks/mri/) (<http://www.cis.rit.edu/htbooks/mri/>) : Online book
- [Sparse MRI](https://people.eecs.berkeley.edu/~mlustig/Software.html) (<https://people.eecs.berkeley.edu/~mlustig/Software.html>) : src
- [Michael \(Miki\) Lustig](https://people.eecs.berkeley.edu/~mlustig/index.html) (<https://people.eecs.berkeley.edu/~mlustig/index.html>) : MRI, CS, data, src(BART, SigPy, Sparse MRI, ...)
- [CompressedSensingMRI](https://github.com/akn264/CompressedSensingMRI) (<https://github.com/akn264/CompressedSensingMRI>) : src
- [mripy](https://github.com/peng-cao/mripy) (<https://github.com/peng-cao/mripy>) : A python based MRI reconstruction toolbox with compressed sensing, parallel imaging and machine-learning functions

8.1.2 NMR 系统概述

8.1.3 NMR 成像理论

8.1.4 参考文献

8.1.5 名词术语

Magnetic Resonance Imaging 磁共振成像 (Magnetic Resonance Imaging, MRI)

Nuclear Magnetic Resonance Imaging 核磁共振成像 (Nuclear Magnetic Resonance Imaging, NMRI)

8.2 名词术语

Medical Imaging 医学成像 (Medical Imaging, MI)

CHAPTER 9

第九卷天文学

9.1 天体测量学

9.2 天体力学

9.3 天体物理学

9.3.1 地球物理学

9.3.1.1 引言

地球物理学被普遍归为地球科学的分支, 作者认为, 应该以全宇宙的视角研究地球, 地球只是宇宙中的一个天体, 可以归类到天体物理学中.

9.3.1.2 人造地球卫星

9.3.1.2.1 卫星轨道参数

9.3.1.2.2 常见卫星轨道

- 太阳同步轨道, 倾角总在 98 度处 (http://www.sohu.com/a/207994146_466840)
- 地球同步、地球静止、半同步、太阳同步、极地、莫尼卡轨道 (<https://blog.csdn.net/ljie45655/article/details/101951774>)

常见的人造卫星轨道有 地球同步轨道 (Earth Synchronous Orbit, 轨道倾角 $0^\circ - 90^\circ$), 地球静止轨道 (Earth Orbit, 轨道倾角 0°), 半同步轨道 (轨道倾角 $0^\circ - 180^\circ$),

9.3.1.2.2.1 地球同步轨道

轨道高度 35786 公里 (据地球中心 42164 公里), 倾角 $0^\circ - 90^\circ$, 轨道周期与地球自转周期相同, 故称为地球同步轨道

9.3.1.2.2.2 半同步轨道

轨道高度 20200 公里 (据地球中心 26578 公里), 倾角 $0^\circ - 180^\circ$, 轨道周期为地球自转周期一半, 故称为半同步轨道

9.3.1.2.2.3 极地轨道

极地轨道是指卫星运行轨道平面与赤道平面夹角 (轨道倾角) 为 90° 的卫星轨道, 卫星可以到达南北极上空, 加上地球自转, 极地卫星可以观测到地球上的任意位置.

9.3.1.2.2.4 太阳同步轨道

太阳同步轨道 (Sun Synchronous Orbit) 是指卫星轨道平面绕地球自转轴的旋转方向和角速度与地球绕太阳公转的方向和平均角速度相同的卫星轨道, 即卫星以相同的当地太阳时间经过任何给定点, 亦即任一时刻, 太阳与地球和太阳与卫星间的夹角相同. 太阳轨道倾角约 98° . 适当调整卫星的倾角 i , 轨道高度 a , 可使卫星旋转角速度等于地球公转的平均角速度 $0.986^\circ/day$

$$\cos i = -4.7737 \times 10^{-15} \times (1 - e^2) \times a^{7/2}$$

半长轴 a 的单位为 km.

CHAPTER 10

第十卷应用实践

10.1 变化检测

10.1.1 引言

10.1.1.1 资源

- [Awesome-RS-changedetection](https://github.com/wenhwu/Awesome-RS-changedetection) (<https://github.com/wenhwu/Awesome-RS-changedetection>)
- [helobilly 个人主页](https://helobilly.github.io/works/) (<https://helobilly.github.io/works/>)
- [武汉大学 CVEO](https://cveo.github.io/) (<https://cveo.github.io/>)

10.1.1.2 数据集

- [OSCD](https://rcdaudt.github.io/oscd/) (<https://rcdaudt.github.io/oscd/>) : Onera Satellite Change Detection Dataset, 可从 [这里](https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download) (<https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download>)
- [SZTAKI](http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) (http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) : A Ground truth collection for change detection in optical aerial images taken with several years time differences, 用户名：sztaki, 密码：deva
- [Grupo Hiperespectral](https://gitlab.citius.usc.es/hiperespectral) (<https://gitlab.citius.usc.es/hiperespectral>) : 高光谱变化检测数据
- [ABCDdataset](https://github.com/gistairc/ABCDdataset) (<https://github.com/gistairc/ABCDdataset>)
- [Inria Aerial Image Labeling Dataset](https://project.inria.fr/aerialimagelabeling/files/) (<https://project.inria.fr/aerialimagelabeling/files/>)
- [Massachusetts Buildings Dataset](https://www.cs.toronto.edu/~vmnih/data/) (<https://www.cs.toronto.edu/~vmnih/data/>) Building Extraction
- [Building change detection dataset](http://study.rsgis.whu.edu.cn/pages/download/building_dataset.html) (http://study.rsgis.whu.edu.cn/pages/download/building_dataset.html)

- [MtS-WH](http://sigma.whu.edu.cn/resource.php) (<http://sigma.whu.edu.cn/resource.php>) : 武汉多时相场景变化检测数据集, 数据集主要用于进行场景变化检测的方法理论研究与验证. MtS-WH 主要包含两幅 7200 x 6000 像素大小的高分辨率遥感影像, 覆盖范围为中国武汉市汉阳区. 影像分别获取于 2002 年 2 月和 2009 年 6 月, 分辨率为 1m, 包含 4 个波段 (蓝, 绿, 红和近红外波段). 每个时相训练集包括 190 张影像, 测试集包括 1920 张影像.
- [OSCD](https://rcdaudt.github.io/oscd/) (<https://rcdaudt.github.io/oscd/>) : Onera Satellite Change Detection Dataset, 多光谱 (13 波段), 可从 [这里](https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download) (<https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download>)
- [SZTAKI](http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) (http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) : A Ground truth collection for change detection in optical aerial images taken with several years time differences, 用户名: sztaki, 密码: deva
- [Grupo Hiperespectral](https://gitlab.citius.usc.es/hiperespectral) (<https://gitlab.citius.usc.es/hiperespectral>) : 高光谱变化检测数据
- [ABCDdataset](https://github.com/gistairc/ABCDdataset) (<https://github.com/gistairc/ABCDdataset>)

10.1.1.3 文章与代码

- Detecting Urban Changes with Recurrent Neural Networks from Multitemporal Sentinel-2 Data: [pdf](https://sagarverma.github.io/others/CD_IGARSS_2019.pdf) (https://sagarverma.github.io/others/CD_IGARSS_2019.pdf) [code](https://github.com/granularai/fabric) (<https://github.com/granularai/fabric>)
- Unsupervised Deep Slow Feature Analysis for Change Detection in Multi-Temporal Remote Sensing Images: [pdf](https://arxiv.org/abs/1812.00645) (<https://arxiv.org/abs/1812.00645>) [code](https://github.com/rulixiang/DSFANet) (<https://github.com/rulixiang/DSFANet>)

10.1.2 基于深度学习的变化检测

10.1.2.1 基于深度学习的变化检测方法概述

10.1.2.1.1 目前进展

10.1.2.1.2 下一步方向

10.1.2.2 基于孪生网络的变化检测

10.1.2.2.1 基于全卷积孪生网络的变化检测

[?] 孪生网络, 如 图 10.1 所示

10.1.2.2.2 基于多尺度卷积孪生网络的变化检测

[?]

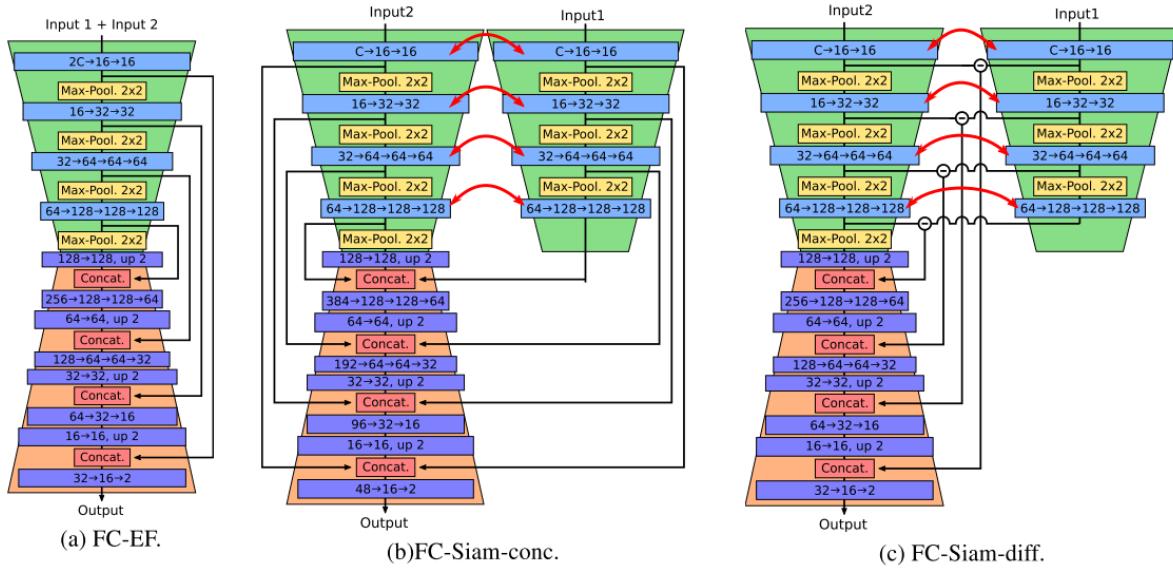


图 10.1: 基于全卷积孪生网络的变化检测网络结构示意图

10.1.2.2.3 基于扩张金字塔

10.1.2.3 自步变化学习

10.1.2.3.1 动机与贡献

在变化检测中, 变化与非变化区域均存在多尺度特性, 对于小尺度变化在大尺度感受野下很容易消失, 同样小尺度感受野很难捕捉大尺度变化信息。此外, 在变化检测中, 变化区域往往比较小, 非变化区域较多, 变化与非变化类别不平衡, 由于人工在标记样本时, 判断变化与否的标准不一致, 样本学习的难易程度相差较大。同时学习, 不能得到鲁棒性较好的结果。

1. 针对于变化信息的多尺度特性, 设计了一种多尺度变化检测网络 (ASPPYnet), 用于捕捉不同粒度下的变化信息;
2. 针对于样本变化类别不均衡, 标记样本学习难易程度不一的问题, 借鉴自步学习思想, 引入从易到难的变化学习机制;
3. 理论与实验结果证明, 引入的多尺度特征结构与自步变化学习训练机制极大程度上改善了模型的性能。

10.1.2.3.2 模型与方法

10.1.2.3.3 实验与分析

10.2 路径规划

10.2.1 引言

10.2.1.1 什么是路径规划

路径规划 (*Path Planning*)

10.2.1.2 常用方法

常见的路径规划算法有基于 **几何分析** (Geometric Analysis), **回报** (Reward-based Algorithms) 的方法, 基于搜索的 **网格搜索** (Grid-based Search), **间隔搜索** (Interval-based Search) 方法, 基于 **多目标优化** (Multi-Objective Optimization) 的方法, 基于 **遗传算法** (genetic algorithm), **神经网络** (Neural Network) 等人工智能算法.

10.2.1.3 有用的资源

10.2.1.3.1 book

- [Planning Algorithms](http://planning.cs.uiuc.edu/) (<http://planning.cs.uiuc.edu/>)
- [Robot Motion Planning and Control](http://homepages.laas.fr/jpl/book.html) (<http://homepages.laas.fr/jpl/book.html>)

10.2.1.3.2 software

- [OMPL](http://ompl.kavrakilab.org/) (<http://ompl.kavrakilab.org/>) : The Open Motion Planning Library
- [OpenRAVE](http://openrave.org/) (<http://openrave.org/>) : OpenRAVE provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications.
- [MSL](http://msl.cs.uiuc.edu/msl/) (<http://msl.cs.uiuc.edu/msl/>) : Motion Strategy Library. The Motion Strategy Library (MSL) allows easy development and testing of motion planning algorithms for a wide variety of applications.
- [MPK](https://ai.stanford.edu/~mitul/mpk/) (<https://ai.stanford.edu/~mitul/mpk/>) : A C++ library and toolkit for developing single- and multi-robot motion planners.

10.2.1.3.3 library

- [Path-Planning-Simulator](https://github.com/sahibdhanjal/Path-Planning-Simulator) (<https://github.com/sahibdhanjal/Path-Planning-Simulator>)

10.2.2 覆盖规划

10.2.2.1 引言

10.2.2.1.1 什么是区域覆盖规划

在路径规划中, **覆盖规划** (Coverage Planning) 是指可以完成指定区域内完全扫描的规划. 即对于给定区域, 要求找到一条最短扫描路径, 使得按照该路径扫描可以覆盖全区域. 通常区域内含有若干障碍物子区域和若干重点扫描子区域, 这就要求规划算法可以自动避开障碍物和穿越重点扫描区域.

10.2.2.1.2 区域覆盖规划算法

10.2.2.2 几何法

10.2.2.2.1 初始规划

10.2.2.2.2 障碍物躲避

人工设置多个圆形障碍区域, 要求算法改变初始规划的路线, 使得新规划的路线自动避开障碍区域.

设有待规划区域 \mathcal{R} , N 个原始路径有序点集 $\mathcal{P} = \{P_n\}_{n=1}^N$, M 个障碍圆区域 $\mathcal{O} = \{\mathcal{O}_m\}_{m=1}^M$, 其中, $P_n = (x_n, y_n)^T$, 障碍物是由方程 $\mathcal{O}_m : (x - x_{c_m})^2 + (y - y_{c_m})^2 \leq r_{c_m}^2$ 确定的圆形区域.

10.2.2.2.2.1 最短圆弧法

该方法的主要思想是将原始经过障碍圆区域的路径, 转变为与障碍圆圆周相距一安全距离 d 的一段最短圆弧, 如 图 10.2 所示, 蓝色为原始路径, 黑色圆表示障碍物, 红色圆弧为避障路径. 通过确定连续的两个点, 即入圆点 P_a 和出圆点 P_b , 对圆弧 $P_a P_b$ 进行离散化得到重新规划的避障路径.

详细的算法步骤陈述如下

注解: 最短圆弧法避障算法

输入: 原始路径有序点集构成的矩阵 $\mathbf{P} = [x_1, y_1 \ x_2, y_2 \ \cdots \ x_N, y_N] \in \mathbb{R}^{N \times 2}$, M 个障碍圆区域参数构成的矩阵 $\mathbf{C} = [x_{c_1}, y_{c_1}, r_{c_1} \ x_{c_2}, y_{c_2}, r_{c_2} \ \cdots \ x_{c_M}, y_{c_M}, r_{c_M}] \in \mathbb{R}^{M \times 3}$, 安全距离 d , 角度采样点数 N_θ .

输出: 规划后的路径有序点集构成的矩阵 \mathbf{R}

Step0: 初始化规划路径 $\mathbf{R} = \mathbf{P}$, 圆心 $\mathbf{O} = \mathbf{C}(N, 1 : 2)$

Step1: 计算每一个障碍圆圆心 \mathbf{O} 到原始路径中每一条线段 (共 $N - 1$, 并找出距离小于 r 的线段, 记为 $\mathcal{L} \in \mathbb{R}^{L \times 4}$.

Step2: 对这些线段进行离散化, 并判断离散化的点是否在圆内, 从而得到若干入圆点出圆点对 $\{P_a, P_b\}$

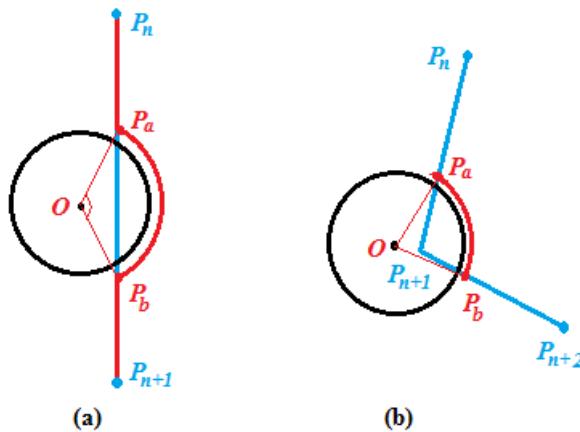


图 10.2: 圆形障碍物躲避示意图

Step3: 对每一个入圆点出圆点对 P_a, P_b , 计算夹角 $\theta_a = \langle \vec{OP}_a, \vec{e}_x \rangle$, $\theta_b = \langle \vec{OP}_b, \vec{e}_x \rangle$, 其中, $\vec{e}_x = [1, 0]^T$

Step4: 若 $|\theta_b - \theta_a| < \pi$, 则离散化角度区间 $\theta = [\theta_a, \theta_b]_{N_\theta}$; 否则离散化角度区间 $\theta = [[\theta_a, \theta_a/|\theta_a|]_{N_\theta/2}, [\theta_b/|\theta_b|, \theta_b]_{N_\theta/2}]$

Step5: 生成圆心为 O , 半径为 $r + d$, 一段圆弧 $\widehat{P_a P_b}$, 并插入到路径有序点集中 R 中.

Step6: 输出 R .

10.2.2.3 价值区域穿越

人工设置多个凸多边形高价值区域, 要求算法根据设置的区域, 改变初始规划的路线, 使得新规划的路线经过高价值区域.

设有待规划区域 \mathcal{R} , N 个原始路径有序点集 $\mathcal{P} = \{P_n\}_{n=1}^N$, M 个高价值区域 $\mathcal{V} = \{\mathcal{V}_m\}_{m=1}^M$, 其中, $P_n = (x_n, y_n)^T$, 价值区域是由有序点集 $\mathcal{V}_m : \{(x_1^{v_m}, y_1^{v_m}), (x_2^{v_m}, y_2^{v_m}), \dots, (x_K^{v_m}, y_K^{v_m})\}$ 围成的多边形区域.

10.2.2.3.1 矩形插入法

算法描述: 对于每一个价值区域, 计算重心, 找到原始路径线段中与重心相距最近的线段 $L_s = \vec{P_0 P_1}$, 接着计算过重心且与该线段 L_s 平行的线与凸多边形区域的交点 Q_1, Q_2 , 分别计算点 Q_1, Q_2 在线段 L_s 上的垂直投影点, 记为 U_1, U_2 , 若 $\|P_0 - U_1\|_2^2 \leq \|P_0 - U_2\|_2^2$, 则更新路径为 $U_1 \rightarrow Q_1 \rightarrow Q_2 \rightarrow U_2$, 反之则更新路径为 $U_2 \rightarrow Q_2 \rightarrow Q_1 \rightarrow U_1$.

10.2.2.2.4 实验与分析

10.2.2.2.4.1 实验 1

10.2.2.2.4.2 实验代码

具体代码可以在作者 GitHub 仓库获取, 地址为 [mpathplanning](https://github.com/antsfamily/mpathplanning) (<https://github.com/antsfamily/mpathplanning>).

代码 10.1: demo_PathPlanning_Geometric.m

```

1 clc
2 clear all
3 close all
4
5 safedist = 1;
6 NA = 6;
7 dedge = 3;
8 dsep = 20;
9
10 PlanningRegion = [0 0; 200 0; 200 100; 0 100];
11
12 ValueRegions = {
13     [10 40; 15 40; 18 50; 13 60; 5 50; 8 40]
14     [40 20; 45 20; 48 25; 43 30; 35 24; 38 20]
15     [50 40; 55 40; 58 45; 53 50; 45 44; 48 40]
16     [40 60; 45 60; 48 65; 43 70; 35 64; 38 60]
17     [80 40; 85 40; 88 50; 83 60; 75 50; 78 40]
18 };
19
20 ObstacleRegions = [
21     10, 10, 2;
22     46, 10, 4;
23     50, 80, 10;
24     88, 30, 5;
25     92, 15, 5;
26     90, 3, 2;
27     90, 94, 6;
28     165, 50, 5;
29 ];
30
31 %% Generates Original Path
32 xmin = min(PlanningRegion(:, 1));
33 xmax = max(PlanningRegion(:, 1));
34 ymin = min(PlanningRegion(:, 2));
35 ymax = max(PlanningRegion(:, 2));
36
37 Nx = uint16(((xmax-dedge) - (xmin+dedge)) / dsep);
38 x1 = linspace(xmin+dedge, xmax-dedge, Nx)';

```

(下页继续)

(续上页)

```

39 x2 = linspace(xmin+dedge, xmax-dedge, Nx)';
40 y1 = ones(Nx, 1) * (ymin + dedge);
41 y2 = ones(Nx, 1) * (ymax - dedge);
42
43 xy1 = [x1 y1];
44 xy2 = [x2 y2];
45
46 OriginalPath = zeros(2*Nx, 2);
47
48 idx = 1:2:Nx;
49 OriginalPath(idx*2-1, :) = xy1(idx, :);
50 idx = 2:2:Nx;
51 OriginalPath(idx*2, :) = xy1(idx, :);
52 idx = 1:2:Nx;
53 OriginalPath(idx*2, :) = xy2(idx, :);
54 idx = 2:2:Nx;
55 OriginalPath(idx*2-1, :) = xy2(idx, :);
56
57 figure
58 title('Path planning based on geometric analysis')
59 xlabel('x')
60 ylabel('y')
61 hold on
62 % original path
63 opplot(OriginalPath, '-b', 'linewidth', 1);
64 ReplanedPath = OriginalPath;
65
66 %% Valuable Regions Path Planning
67 % show polygonal valuable regions
68 cpplot(ValueRegions, '-g', 'linewidth', 1)
69
70 % polygonal valuable region passing program based on geometric method
71 ReplanedPath = pvp_geometry(ReplanedPath, ValueRegions);
72
73 %% Obstacle Path Planning
74 % show circular obstacles
75 circleplot(ObstacleRegions, '-k', 'linewidth', 1)
76
77 % circular obstacle avoidance program based on geometric method
78 ReplanedPath = coa_geometry(ReplanedPath, ObstacleRegions, safedist, NA);
79
80 % smooth
81 % P = smooth_path(P);
82
83 % display planning path dynamically, 0.3s/point
84 dynshow(ReplanedPath, 0.3, 10, '.r', 'linewidth', 2);
85

```

(下页继续)

(续上页)

```

86 % show final planning path
87 opplot(ReplanedPath, '-m', 'linewidth', 2);

```

10.2.2.2.4.3 实验结果

设置障碍圆与价值区域, 如图所示, 使用本节算法的结果如图所示.

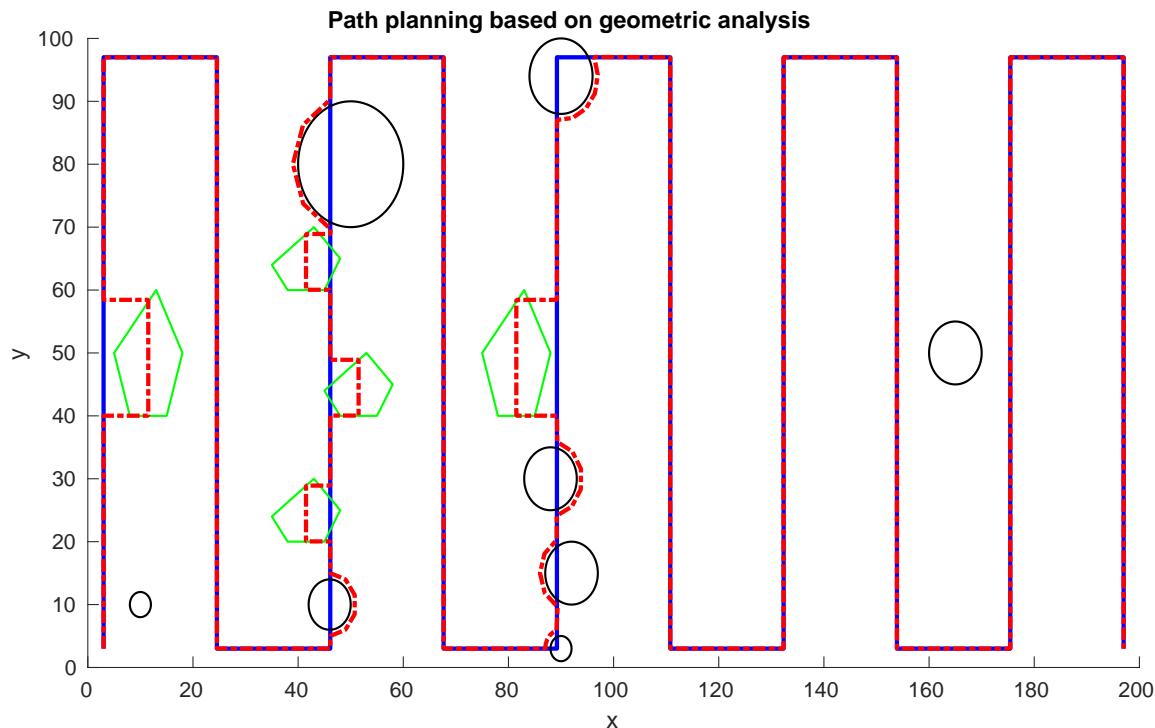


图 10.3: 最短圆弧避障算法与矩形插入价值区域穿越法结果示意图

10.2.3 动态规划

10.2.4 参考文献

10.2.5 名词术语

Configuration Space 配置空间 (Configuration Space)

Free Space 自由空间 (Free Space) 是指无障碍, 无碰撞可自由通过的空间, 常记为 C_{free} , 与障碍空间 $C_{obstacle}$ 互补.

Obstacle Space 障碍空间 (Obstacle Space) 是指不能经过的区域, 常记为 $C_{obstacle}$, 与自由空间 C_{free} 互补.

Path Planning 路径规划 (Path Planning (http://en.volupedia.org/wiki/Path_planning))

Target Space 目标空间 (Target Space) 是自由空间的线性子空间, 指代我们期望物体移动的空间.

10.3 评价方法

10.3.1 经典评价方法

10.3.1.1 信息检索

10.3.1.1.1 常用数据检索指标

10.3.1.1.1.1 基本概念

信息检索是指从一些相关 (relevant) 和不相关 (irrelevant) 的信息中检索出相关信息, 为便于论述, 记检索出的为正 (positive) 响应, 未检索出的为负 (negative) 响应. 下面论述的这些评价指标也通常被用在目标检测, 图像分割, 变化检测等应用领域. 主要包括以下指标

- 真正 (True Positive, TP): 实际上是正样本, 被检索为正样本, 真相关, 也称正确检出数
- 真负 (True Negative, TN): 实际上是负样本, 被检索为负样本, 真不相关
- 假正 (False Positive, FP): 实际上是负样本, 被检索为正样本, 假相关, 也称为虚警
- 假负 (False Negative, FN): 实际上是正样本, 被检索为负样本, 假不相关, 也称为漏检
- 查准率 (Precision): 检索到的真相关样本占检索到的样本的比例
- 查全率 (Recall): 检索到的真相关样本占总体相关样本的比例
- 相关样本总体: 检索到的真相关样本 (正确检出数, TP) 与未检索到的相关样本 (FN) 之和
- 不相关样本总体: 检索到的假相关样本 (虚警, FP) 与未检索到的不相关样本 (TN) 之和
- 检索到的样本: 检索到的真相关样本 (正确检出数, TP) 与检索到的假相关样本 (FN) 之和

10.3.1.1.1.2 查准率

查准率 (*Precision*) 指检索到的真相关样本占检索到的样本的比例, 即在所有检索结果中, 真正相关样本所占比例, 亦即所有正响应中, 真正相关的样本所占比例, 在分类问题中也称正预测值 (Positive Predictive Value, PPV), 可以表示为

$$\text{PPV} = P = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (10.1)$$

10.3.1.1.1.3 查全率

查全率 (*Recall*), 指检索到的真相关样本占总体相关样本的比例; 即在总体相关样本中, 检出的真正相关样本所占比例; 亦即所有正样本中, 响应为正的样本所占比例, 也称 召回率. 在其它领域也称灵敏度 (Sensitivity), 真正率 (True Positive Rate, TPR) 可以表示为

$$\text{TPR} = R = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10.2)$$

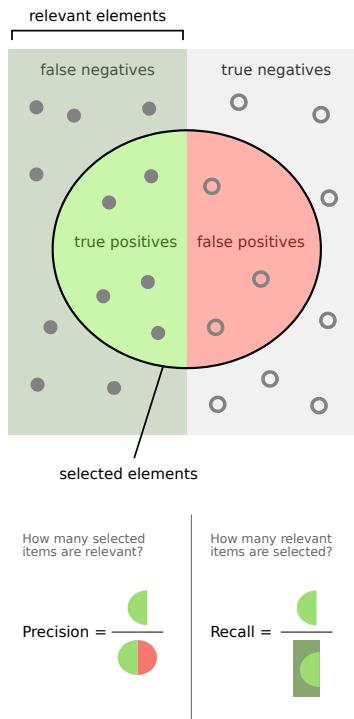


图 10.4: Precision and recall (来自维基百科)

Precision and recall (来自维基百科)

10.3.1.1.4 F-measure

查准率 (P) 与查全率 (R) 只能从一个方面衡量模型性能, 当查准率与查全率均比较高时, 模型效果越好. *F-measure* 是一种融合了查准率与查全率的综合评价指标, 可以表示为

$$F_{\beta} = \frac{(1 + \beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R} \quad (10.3)$$

其中, β 用于调整查准率与查全率所占比重, 通常取 $\beta = 0.5, 1, 2$. 当 $\beta = 0.5$ 时, 更强调查准率; 当 $\beta = 2$ 时, 更强调查全率.

警告: 举例: 假设有 n 对预测与真值数据对 $\{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^n$, 其中前 $n - 1$ 对的值全为零, 第 n 对, 仅有一个正例样本点且被正确预测/检索.

若将所有数据对看作总体, 有 $TP = 1, FP = 0, FN = 0$, 从而有查准率 $P = 1$, 查全率 $R = 1$, 继而有 F 值 $F_{all} = 1$

若将每一个数据对看作总体, 对于前 $n - 1$ 个数据, 有 $TP = 0, FP = 0, FN = 0$, 从而有查准率 $P = 0$, 查全率 $R = 0, F^{(i)} = 0, i = 1, 2, \dots, n - 1$; 对于第 n 个数据对有 $TP = 1, FP = 0, FN = 0$, 从而有查准率 $P = 1$, 查全率 $R = 1, F^{(n)} = 1$, 故平均 F 值为 $F_{avg} = (F^{(1)} + F^{(2)} + \dots + F^{(n-1)} + F^{(n)})/n = 1/n$.

10.3.1.1.5 虚警率

虚警率 (False Alarm Rate) 也称错误发现率 (False Discovery Rate, FDR), 是指在检索出的样本中, 不相关的样本所占比例, 与查准率互补, 可以表示为

$$FDR = \frac{FP}{TP + FP} = 1 - P \quad (10.4)$$

10.3.1.1.6 漏检率

漏检率 (Miss Alarm Rate) 也称假负率 (False Negative Rate, FNR), 是指未检索出的正类样本占总正类样本的比例, 与召回率互补, 可以表示为

$$FNR = \frac{FN}{FN + TP} = 1 - R \quad (10.5)$$

10.3.1.1.7 总结

- sensitivity, recall, hit rate, or true positive rate (TPR)

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$$

- specificity, selectivity or true negative rate (TNR)

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = 1 - FPR$$

- precision or positive predictive value (PPV)

$$PPV = \frac{TP}{TP + FP} = 1 - FDR$$

- negative predictive value (NPV)

$$NPV = \frac{TN}{TN + FN} = 1 - FOR$$

- miss rate or false negative rate (FNR)

$$FNR = \frac{FN}{P} = \frac{FN}{FN + TP} = 1 - TPR$$

- fall-out or false positive rate (FPR)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TNR$$

- false discovery rate (FDR)

$$FDR = \frac{FP}{FP + TP} = 1 - PPV$$

- false omission rate (FOR)

$$FOR = \frac{FN}{FN + TN} = 1 - NPV$$

- Threat score (TS) or Critical Success Index (CSI)

$$TS = \frac{TP}{TP + FN + FP}$$

- accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

- F1 score: is the harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

- Matthews correlation coefficient (MCC)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- Informedness or Bookmaker Informedness (BM)

$$BM = TPR + TNR - 1$$

- Markedness (MK)

$$MK = PPV + NPV - 1$$

10.3.1.2 数据分类

10.3.1.2.1 混淆矩阵

混淆矩阵 (*Confusion Matrix*)

10.3.1.2.2 Kappa 系数

10.3.1.3 相似性度量指标

10.3.1.3.1 集合方法

10.3.1.3.1.1 Jaccard

10.3.1.3.1.2 Jaccard 指数与距离

Jaccard 指数 (index), 也称为并上交 (Intersection over Union, IoU) 或 Jaccard 相似性系数 (*Jaccard Similarity Coefficient*), 用于测量样本集的相似性与多样性. Jaccard 系数用于衡量有限样本集间的相似性, 被定义为集合交的势除并的势.

$$J(\mathbb{A}, \mathbb{B}) = \frac{|\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A} \cup \mathbb{B}|} = \frac{|\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A}| + |\mathbb{B}| - |\mathbb{A} \cap \mathbb{B}|} \quad (10.6)$$

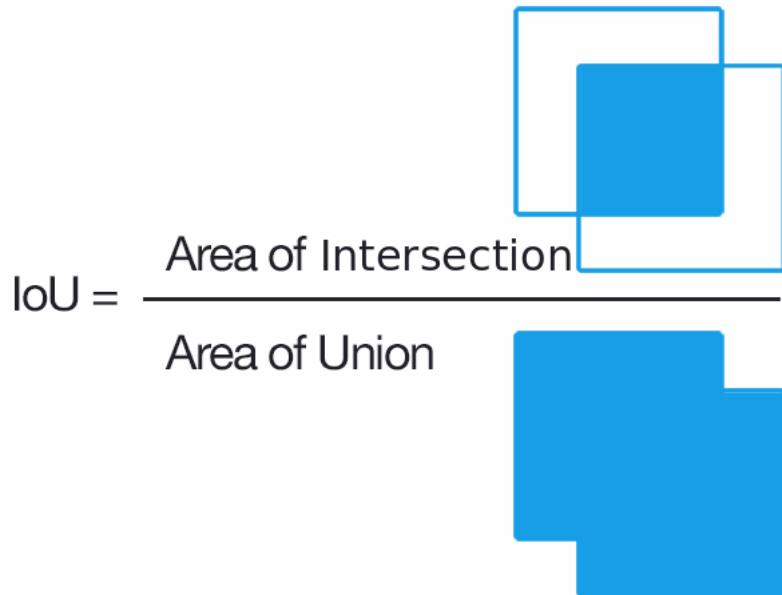


图 10.5: Intersection over Union

Intersection over Union

其中, $|\mathbb{A}|$ 表示集合 \mathbb{A} 的势¹, $0 < J(\mathbb{A}, \mathbb{B}) < 1$, 若 \mathbb{A}, \mathbb{B} 为空集, 定义 $J(\mathbb{A}, \mathbb{B}) = 1$.

Jaccard 距离用于衡量样本集合间的差异, 常定义如下:

$$d_J(\mathbb{A}, \mathbb{B}) = 1 - J(\mathbb{A}, \mathbb{B}) = \frac{|\mathbb{A} \cup \mathbb{B}| - |\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A} \cup \mathbb{B}|}$$

10.3.1.3.1.3 加权 Jaccard 指数与距离

10.3.1.3.1.4 概率 Jaccard 指数与距离

10.3.1.3.1.5 Dice 系数

Sørensen–Dice coefficient 简称为 *Dice Coefficient* 或 Dice Similarity Coefficient, 衡量两个样本集合的相似性, 给定集合 \mathbb{A}, \mathbb{B} , Dice 系数定义为

$$S = \text{DSC} = \frac{2|\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A}| + |\mathbb{B}|}. \quad (10.7)$$

对于二值数据, 采用 true positive (TP), false positive (FP), 和 false negative (FN) 的定义, 则 Dice 系数可以表示为

$$S = \text{DSC} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \quad (10.8)$$

容易得出 Dice 系数与 Jaccard 系数间存在如下关系

$$\begin{aligned} J &= S/(2 - S) \\ S &= 2J/(1 + J) \end{aligned} \quad . \quad (10.9)$$

¹ 集合的势即即集合中元素的个数, 参见集合的势与基数 (页 12).

且此时有 $DSC = F_1$.

提示: 将查准率式.10.1 与查全率式.10.2 代入式.10.1 中, 有

$$\begin{aligned} F_1 &= \frac{(1+1^2) \cdot P \cdot R}{1^2 \cdot P + R} = \frac{(1+1^2) \cdot \frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{1^2 \cdot \frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \\ &= \frac{2TP^2}{(TP+FP)(TP+FN)} \cdot \frac{(TP+FP)(TP+FN)}{TP(2TP+FP+FN)} \\ &= \frac{2TP}{2TP+FP+FN} = DSC \end{aligned}$$

10.3.1.3.2 结构相似性度量

有关图像领域中的结构相似性度量指标 (SSIM) 参见 [Section-EvaluationMethodImageQuality](#) 小节.

10.3.1.4 图像质量评价

10.3.1.4.1 误差类

10.3.1.4.1.1 均方误差

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=0}^N [|\mathbf{I}(i,j)|, |\hat{\mathbf{I}}(i,j)|]^2$$

10.3.1.4.2 信噪比类

$$PSNR = 10 \log 10 \left(\frac{V_{peak}^2}{MSE} \right)$$

10.3.1.4.3 结构相似性度量

10.3.1.4.3.1 SSIM

结构相似性指数 (Structure SIMilarity index, SSIM) 由周等人于 2004 年提出 [?], 设有数据 x 和参考数据 y , μ_x, μ_y 分别为其均值, σ_x, σ_y 分别为标准差, σ_x^2, σ_y^2 为其方差, σ_{xy} 为数据 x 和参考数据 y 的协方差, 用 l, c, s 分别表示亮度 (luminance), 对比度 (contrast) 和结构 (structure) 相似性, 则

$$\begin{aligned} l(x, y) &= \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \\ c(x, y) &= \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \\ s(x, y) &= \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \end{aligned}$$

其中, $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, $c_3 = c_2/2$, L 是数据的动态范围 (dynamic range) (典型的 $L = 2^{\# \text{ bits per pixel}} - 1$). 结构则相似性指标可表示为

$$\text{SSIM}(x, y) = [l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma].$$

当 $\alpha = \beta = \gamma = 1$, SSIM 等价于

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

10.3.1.4.3.2 MSSSIM

10.3.1.4.3.3 GSSIM

基于梯度的结构相似性 (Gradient-based Structure SIMilarity index, GSSIM) 度量方法 [?], 对 SSIM 中的对比度和结构度量部分做了更改, 使用数据的梯度而不是数据来计算. 对于二维图像数据, 梯度的计算可以通过 sobel 算子滤波实现. sobel 算子在水平和垂直方向上可表示为

$$G_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G_h = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

[link text](#)

10.3.1.4.4 实验与分析

10.3.1.4.4.1 核心代码

代码 10.2: demo_python.m

```

1 def ssim(X, Y, win=None, winsize=11, L=None, k1=0.01, k2=0.03, alpha=1, beta=1,
2     gamma=1, isavg=True, full=False):
3     """Structural similarity index
4
5     Parameters
6     -----
7     X : {ndarray}
8         reconstructed
9     Y : {ndarray}
10        referenced
11     win : {[type]}, optional
12         [description] (the default is None, which [default_description])
13     winsize : {number}, optional
14         [description] (the default is 11, which [default_description])

```

(下页继续)

(续上页)

```

14     L : {integer}, optional
15         the dynamic range of the pixel-values (typically this is :math:`2^{\lfloor \# \backslash
16 \rightarrow \text{text} \{ \text{bits per pixel} \} - 1 \rfloor}`. (the default is 255)
17     k1 : {number}, optional
18         [description] (the default is 0.01, which [default_description])
19     k2 : {number}, optional
20         [description] (the default is 0.03, which [default_description])
21     sizeavg : {bool}, optional
22         whether to average (the default is True, which average the result)
23     alpha : {number}, optional
24         luminance weight (the default is 1)
25     beta : {number}, optional
26         contrast weight (the default is 1)
27     gamma : {number}, optional
28         structure weight (the default is 1)
29     isavg : {bool}, optional
30         IF True, return the average SSIM index of the whole iamge,
31     full : {bool}, optional
32         IF True, return SSIM, luminance, contrast and structure index (the default
33         is False, which only return SSIM)
34     """
35
36
37     if L is None:
38         _, L = get_drange(Y.dtype)
39
40
41     if win is None and type(winsize) is not int:
42         winsize = 11
43         win = _SSIM_GAUSSIAN_KERNEL_11X11
44     if win is None and type(winsize) is int:
45         win = create_window(winsize, 1)
46
47     muX = convolve(X, win)
48     muY = convolve(Y, win)
49     muXsq = muX * muX
50     muYsq = muY * muY
51
52     sigmaXsq = np.abs(convolve(X * X, win) - muXsq)
53     sigmaYsq = np.abs(convolve(Y * Y, win) - muYsq)
54     sigmaXY = convolve(X * Y, win) - muX * muY
55
56     sigmaX = np.sqrt(sigmaXsq)
57     sigmaY = np.sqrt(sigmaYsq)

```

(下页继续)

(续上页)

```

59     luminance = (2. * muX * muY + C1) / (muX * muX + muY * muY + C1)
60     contrast = (2 * sigmaX * sigmaY + C2) / (sigmaXsq + sigmaYsq + C2)
61     structure = (sigmaXY + C3) / (sigmaX * sigmaY + C3)
62
63     ssim_map = (luminance**alpha) * (contrast**beta) * (structure**gamma)
64
65     if isavg:
66         ssim_map = np.mean(ssim_map)
67         luminance = np.mean(luminance)
68         contrast = np.mean(contrast)
69         structure = np.mean(structure)
70
71     if full:
72         return ssim_map, luminance, contrast, structure
73     else:
74         return ssim_map

```

10.3.1.5 计算复杂度/性能

10.3.1.5.1 操作数相关指标

操作数或运算次数通常用于衡量处理器的性能, 也可以用于衡量算法的复杂度.

当用于衡量处理器性能时, 通常用 **操作数/每秒** (Operations Per Second, OPS) 表示处理器每秒所能执行的操作次数, 用 **浮点运算次数/每秒** (Floating-point Operations Per Second, FLOPS) 表示处理器每秒所能执行的浮点运算次数. 有时也用 OPS@FPx 来表示处理器的 x 位浮点数处理能力, 如 1TOPS@FP16 表示半精度浮点数下的浮点数运算能力为 1TOPS, 其中, FP 表示浮点数 (Float Point) $1\text{TOPS} = 10^{12}\text{OPS}$, 不加说明时, $\text{FLOPS} = \text{OPS}@FP32$, 下表 10.1 给出了其它单位的值.

表 10.1: 常用数量单位及字母表示

单位	中文	英文	值
M	一百万	Mega	10^6
G	十亿	Giga	10^9
T	一万亿	Tera	10^{12}
P	一千万亿	Peta	10^{15}
E		Exa	10^{18}
Z		Zetta	10^{21}
Y		Yotta	10^{24}

提示: 通常用 FP64 表示 64 位 (bit) 双精度浮点数, FP32 表示 32 位 (bit) 单精度浮点数, FP16 表示 16 位 (bit) 半精度浮点数, FP8 表示 8 位 (bit) 浮点数, INT64 表示 64 位 (bit) 整数等等.

一般地, $1\text{OPS}@FP64 = 2\text{OPS}@FP32 = 4\text{OPS}@FP16 = 8\text{OPS}@FP8$, 不过不是总能成立, 有的处理器略有差

异。此外，一个 1OPS@FP16 的处理器的 16 位整数运算性能略大于 1OPS@INT16。

当用于衡量算法的复杂度时，通常用 **操作数** (OPerations, OPs) 表示算法所需要执行的操作次数，用 **浮点运算次数** (FLoating-point OPerations, FLOPs) 表示算法所需要执行的浮点运算次数。

10.3.1.5.2 处理器的衡量指标

10.3.1.5.2.1 传统指标

- 操作数 (Operations Per Second, OPS): 即每秒执行的操作次数
- 浮点运算次数 (FLoating-point Operations Per Second, FLOPS): 即每秒执行的浮点运算次数
- 单位: 一百万 (Mega, 10^6), 十亿 (Giga, 10^9), 一万亿 (Tera, 10^{12}), 一千万亿 (Peta, 10^{15})

10.3.1.5.3 计算平台与模型性能指标

衡量一个计算平台或模型的性能指标可以通过计算量 (算力/运算量) π 与访存量 (内存读写带宽/内存读写量) β 以及计算强度 \mathcal{I} 来分析。通常地，计算强度定义为

$$\mathcal{I} = \frac{\pi}{\beta}$$

当评估计算平台性能时， π 表示计算平台的算力，即每秒可执行的运算次数 (Operations Per Second, OPS)， β 表示计算平台内存带宽，单位通常为 bps(bit per second)；当评估模型性能时， π 表示模型计算量，即需要执行的运算次数 (OPerations, OPs)， β 表示内存读写量，单位通常为字节 B (Byte)。

举个例子，图 10.6 给出了 SSD300 网络的各层计算量与内存访问量分析结果，包括输入输出大小，参数量 (params)，乘累加量 (Fused multiply-add, Madd)，浮点数运算量 (FLOPs) 以及内存读写量 (MemR+W)。

由 图 10.6 知，SSD300 网络含有 26,285,486 个参数，内存占用为 207.65MB，总乘累加次数为 62.78GB，总浮点运算次数为 31.44GFLOPs@FP32，见 表 10.2。即对于每张 300×300 大小的图片，前向推理一次需要的浮点数运算量约为 31.44GFLOPs 访存量为 542.79MB，故 SSD300 模型的计算强度为 $\mathcal{I}_{ssd} = 31.44\text{GFLOPs}/542.79\text{MB} = 4 \times 31.44e9/542.79e6 = 231.69\text{FLOPs/Byte}$ 。

表 10.2: SSD300 网络参数与计算量

Total params	Total memory	Total MAdd	Total FLOPs	Total MemR+W
26,285,486	207.65MB	62.78G	31.44G	542.79MB

【腾处理器 Ascend 310 的开发套件的内存为 8GB，远大于 SSD300 网络的内存占用量 (207.65MB)，内存规格为 LPDDR4x，带宽为 3200Mbps。】腾处理器 Ascend 310 的浮点数处理能力为 8TOPS@FP16，即 8TFLOPS@FP16，亦即 4TFLOPS@FP32。Ascend 310 处理器的计算强度为 $\mathcal{I}_{Ascend310} = (4 \times 4e12)/(3200e6/8) = 40000\text{FLOPs/Byte}$ 。由此可知，Ascend 310 处理器的计算强度约为 SSD300 模型的计算强度的 172 倍。然而，计算平台的带宽为 3200Mbps，即每秒读写数据量最多为 400MB，而 SSD 网络的访存量为 542.79MB。

	module name	input shape	output shape	params	memory(MB)	MAdd	Flops	MemRead(B)	MemWrite(B)	duration[%]	MemR+W(B)
0	vgg.0	3 300 300	64 300 300	1792.0	21.97	311,040,000.0	161,280,000.0	1087168.0	23040000.0	9.40%	24127168.0
1	vgg.1	64 300 300	64 300 300	0.0	21.97	5,760,000.0	5,760,000.0	23040000.0	23040000.0	0.30%	46080000.0
2	vgg.2	64 300 300	64 300 300	36928.0	21.97	6,635,520,000.0	3,323,520,000.0	23187712.0	23040000.0	9.76%	46227712.0
3	vgg.3	64 300 300	64 300 300	0.0	21.97	5,760,000.0	5,760,000.0	23040000.0	23040000.0	0.26%	46080000.0
4	vgg.4	64 300 300	64 150 150	0.0	5.49	4,320,000.0	5,760,000.0	23040000.0	5760000.0	21.29%	28800000.0
5	vgg.5	64 150 150	128 150 150	73856.0	10.99	3,317,760,000.0	1,661,760,000.0	6055424.0	11520000.0	7.03%	17575424.0
6	vgg.6	128 150 150	128 150 150	0.0	10.99	2,880,000.0	2,880,000.0	11520000.0	11520000.0	0.16%	23040000.0
7	vgg.7	128 150 150	128 150 150	147584.0	10.99	6,635,520,000.0	3,320,640,000.0	12118336.0	11520000.0	6.33%	23630336.0
8	vgg.8	128 150 150	128 150 150	0.0	10.99	2,880,000.0	2,880,000.0	11520000.0	11520000.0	0.64%	23040000.0
9	vgg.9	128 150 150	128 75 75	0.0	2.75	2,160,000.0	2,880,000.0	11520000.0	2880000.0	4.44%	14400000.0
10	vgg.10	128 75 75	256 75 75	295168.0	5.49	3,317,760,000.0	1,660,320,000.0	4060672.0	5760000.0	2.32%	9820672.0
11	vgg.11	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0	5760000.0	5760000.0	0.04%	11520000.0
12	vgg.12	256 75 75	256 75 75	590080.0	5.49	6,635,520,000.0	3,319,200,000.0	8120320.0	5760000.0	3.26%	13880320.0
13	vgg.13	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0	5760000.0	5760000.0	0.02%	11520000.0
14	vgg.14	256 75 75	256 75 75	590080.0	5.49	6,635,520,000.0	3,319,200,000.0	8120320.0	5760000.0	3.01%	13880320.0
15	vgg.15	256 75 75	256 75 75	0.0	5.49	1,440,000.0	1,440,000.0	5760000.0	5760000.0	0.03%	11520000.0
16	vgg.16	256 75 75	256 38 38	0.0	1.41	1,108,992.0	1,440,000.0	5760000.0	1478656.0	1.74%	7238656.0
17	vgg.17	256 38 38	512 38 38	1180160.0	2.82	3,406,823,424.0	1,704,151,040.0	6199296.0	2957312.0	1.70%	9156608.0
18	vgg.18	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0	2957312.0	2957312.0	0.02%	5914624.0
19	vgg.19	512 38 38	512 38 38	2359880.0	2.82	6,813,646,848.0	3,407,562,752.0	12396544.0	2957312.0	3.86%	15535856.0
20	vgg.20	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0	2957312.0	2957312.0	0.02%	5914624.0
21	vgg.21	512 38 38	512 38 38	2359880.0	2.82	6,813,646,848.0	3,407,562,752.0	12396544.0	2957312.0	4.16%	15535856.0
22	vgg.22	512 38 38	512 38 38	0.0	2.82	739,328.0	739,328.0	2957312.0	2957312.0	0.02%	5914624.0
23	vgg.23	512 38 38	512 19 19	0.0	0.71	554,496.0	739,328.0	2957312.0	739328.0	0.99%	3696400.0
24	vgg.24	512 19 19	512 19 19	2359880.0	0.71	1,703,411,712.0	851,890,688.0	10178560.0	739328.0	1.98%	18917888.0
25	vgg.25	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0	739328.0	739328.0	0.02%	1478656.0
26	vgg.26	512 19 19	512 19 19	2359880.0	0.71	1,703,411,712.0	851,890,688.0	10178560.0	739328.0	1.83%	18917888.0
27	vgg.27	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0	739328.0	739328.0	0.02%	1478656.0
28	vgg.28	512 19 19	512 19 19	2359880.0	0.71	1,703,411,712.0	851,890,688.0	10178560.0	739328.0	1.00%	18917888.0
29	vgg.29	512 19 19	512 19 19	0.0	0.71	184,832.0	184,832.0	739328.0	739328.0	0.02%	1478656.0
30	vgg.30	512 19 19	512 19 19	0.0	0.71	1,478,656.0	184,832.0	739328.0	739328.0	1.40%	1478656.0
31	vgg.31	512 19 19	1024 19 19	4719616.0	1.41	3,406,823,424.0	1,703,781,376.0	19617792.0	1478656.0	6.17%	21096448.0
32	vgg.32	1024 19 19	1024 19 19	0.0	1.41	369,664.0	369,664.0	1478656.0	1478656.0	0.04%	2957312.0
33	vgg.33	1024 19 19	1024 19 19	1049600.0	1.41	757,071,872.0	378,995,600.0	5677056.0	1478656.0	2.00%	7155712.0
34	vgg.34	1024 19 19	1024 19 19	0.0	1.41	369,664.0	369,664.0	1478656.0	1478656.0	0.02%	2957312.0
35	L2Norm	512 38 38	512 38 38	512.0	2.82	0.0	0.0	0.0	0.0	0.22%	0.0
36	extras.0	1024 19 19	256 19 19	262400.0	0.35	189,267,968.0	94,726,400.0	2528256.0	369664.0	0.28%	2897920.0
37	extras.1	256 19 19	512 18 10	1189160.0	0.20	235,929,600.0	118,016,000.0	5099384.0	204800.0	0.37%	5295104.0
38	extras.2	512 10 10	128 18 10	65664.0	0.05	13,107,200.0	6,566,400.0	467456.0	51200.0	0.18%	518656.0
39	extras.3	128 10 10	256 5 5	295168.0	0.02	14,745,600.0	7,379,200.0	1231872.0	25600.0	0.39%	1257472.0
40	extras.4	256 5 5	128 5 5	32896.0	0.01	1,638,400.0	822,400.0	157184.0	12800.0	0.18%	169984.0
41	extras.5	128 5 5	256 3 3	295168.0	0.01	5,308,416.0	2,656,512.0	1193472.0	9216.0	1.05%	1202688.0
42	extras.6	256 3 3	128 3 3	32896.0	0.00	589,824.0	296,064.0	140880.0	4608.0	0.13%	145408.0
43	extras.7	128 3 3	256 1 1	295168.0	0.00	589,824.0	295,168.0	1185288.0	1024.0	0.12%	1186304.0
44	loc.0	512 38 38	16 38 38	73744.0	0.09	212,926,464.0	106,486,336.0	3252288.0	92416.0	0.31%	3344704.0
45	loc.1	1024 19 19	24 19 19	221208.0	0.03	159,694,848.0	79,856,088.0	2363488.0	34656.0	0.23%	2398144.0
46	loc.2	512 10 10	24 10 10	119616.0	0.01	22,118,400.0	11,061,600.0	647264.0	9600.0	0.51%	656864.0
47	loc.3	256 5 5	24 5 5	55320.0	0.00	2,764,800.0	1,383,000.0	246880.0	2400.0	0.13%	249280.0
48	loc.4	256 3 3	16 3 3	36880.0	0.00	663,552.0	331,920.0	156736.0	576.0	0.11%	157312.0
49	loc.5	256 1 1	16 1 1	36880.0	0.00	73,728.0	36,880.0	148544.0	64.0	0.09%	148608.0
50	conf.0	512 38 38	84 38 38	387156.0	0.46	1,117,863,936.0	559,053,264.0	4505936.0	485184.0	0.70%	4991120.0
51	conf.1	1024 19 19	126 19 19	1161342.0	0.17	838,397,952.0	419,244,462.0	6124024.0	181944.0	0.57%	6305968.0
52	conf.2	512 10 10	126 10 10	580734.0	0.05	116,121,600.0	58,073,400.0	2527736.0	50400.0	0.23%	2578136.0
53	conf.3	256 5 5	126 5 5	298430.0	0.01	14,515,200.0	7,260,750.0	1187320.0	12600.0	0.14%	1199929.0
54	conf.4	256 3 3	84 3 3	193620.0	0.00	3,483,649.0	1,742,580.0	783696.0	3024.0	0.11%	786720.0
55	conf.5	256 1 1	84 1 1	193620.0	0.00	387,072.0	193,620.0	775584.0	336.0	0.09%	775840.0
56	softmax	8732 21	8732 21	0.0	0.70	550,115.0	0.0	0.0	0.0	0.06%	0.0
total				26285486.0	207.65	62,782,359,651.0	31,435,153,596.0	0.0	0.0	100.00%	54278664.0

图 10.6: SSD 300 网络计算量与内存访问量分析

10.3.2 名词术语

Confusion Matrix 混淆矩阵 ([Confusion Matrix](http://en.volupedia.org/wiki/Confusion_matrix) (http://en.volupedia.org/wiki/Confusion_matrix)), 参见[数据分类](#) (页 613) 小节.

Dice Coefficient Dice 系数 ([Dice Coefficient](http://en.volupedia.org/wiki/Dice_coefficient) (http://en.volupedia.org/wiki/Dice_coefficient), DC), 参见[信息检索](#) (页 610) 小节.

F-measure F-measure ([F-measure](http://en.volupedia.org/wiki/Precision_and_recall) (http://en.volupedia.org/wiki/Precision_and_recall), 参见[信息检索](#) (页 610) 小节.

Jaccard Similarity Coefficient Jaccard 相似性系数 (Jaccard Similarity Coefficient (http://en.volupedia.org/wiki/Jaccard_similarity_coefficient), JSC), 即 IoU, 参见[信息检索](#) (页 610) 小节.

Precision 查准率 ([Precision](http://en.volupedia.org/wiki/Precision_and_recall) (http://en.volupedia.org/wiki/Precision_and_recall), 参见[信息检索](#) (页 610) 小节. 节 10.3.1.1

Predicted Positive Condition 预测阳性条件 (Predicted Positive Condition (http://en.volupedia.org/wiki/Precision_and_recall), PPC), 参见[信息检索](#) (页 610) 小节.

Recall 查全率 ([Recall](http://en.volupedia.org/wiki/Precision_and_recall) (http://en.volupedia.org/wiki/Precision_and_recall), 参见[信息检索](#) (页 610) 小节.

10.4 补充内容

10.4.1 数据集

10.4.1.1 数据集概览

10.4.1.1.1 混合

- By Department of Computer Science 5 (<http://www5.cs.fau.de/research/data/>) : Computational Two-Illuminant, Carotid Artery TOF MRA, DaLiAc - Daily Life Activities, Digital Brain Perfusion Phantom, Multi-Illuminant, Gold Standard Database for Evaluation of Fundus Image Segmentation Algorithms, Supervised Multispectral Image Segmentation, Image Manipulation, CAVAREV: CArdiac VAsculature Reconstruction EVAluation, RabbitCT: Benchmark for High-speed C-arm CT, An Annotated Image Database for Evaluation of Cell Detection Algorithms, 3-D Satellite Camera Frames, Multi-Sensor Super-Resolution, Bamboo Scroll

10.4.1.1.2 遥感影像

10.4.1.1.2.1 综合

- 武汉大学遥感图像标注数据库: 含分类, 检测, 变化检测
 - [homepage](http://captain.whu.edu.cn/WUDA-RSImg/index.html) (<http://captain.whu.edu.cn/WUDA-RSImg/index.html>)
 - DOTA: 目标检测, 2806 幅图像, 180k 物体, 15 类

- AID: 场景分类, 10k 幅图像, 30 类
 - AID++: 场景分类, 1000k 幅图像, 46 类
 - GID: 陆地使用分类, 150 幅图像, 5 类
 - SGID: 语义变化检测, 555 幅图像对, 3 类
 - SAR-building: 建筑检测, 2000 幅图像
 - SAR-ship: 舰船检测, 2000 幅图像
- **Awesome Satellite Imagery Datasets:** 各种卫星影像数据集, 分割, 分类, 检测, 融合
 - [homepage](https://github.com/chrieke/awesome-satellite-imagery-datasets) (<https://github.com/chrieke/awesome-satellite-imagery-datasets>)
 - **Satellite Image and Deep Learning: Resources for performing deep learning on satellite imagery**
 - [homepage](https://github.com/robmarkcole/satellite-image-deep-learning) (<https://github.com/robmarkcole/satellite-image-deep-learning>)
 - **遥感数据集** (<https://zhangbin0917.github.io/2018/06/12/%E9%81%A5%E6%84%9F%E6%95%B0%E6%8D%AE%E9%9B%86/>)
图像分类, 目标检测, 语义分割, 建筑道路检测, 变化检测, 超分辨; 光学, 多光谱, SAR, 光学雷达, 无人机数据等

10.4.1.2 变化检测数据集

10.4.1.2.1 遥感影像

- [Inria Aerial Image Labeling Dataset](https://project.inria.fr/aerialimagelabeling/files/) (<https://project.inria.fr/aerialimagelabeling/files/>)
- [Massachusetts Buildings Dataset](https://www.cs.toronto.edu/~vmnih/data/) (<https://www.cs.toronto.edu/~vmnih/data/>) Building Extraction
- [Building change detection dataset](http://study.rsgis.whu.edu.cn/pages/download/building_dataset.html) (http://study.rsgis.whu.edu.cn/pages/download/building_dataset.html)
- [MtS-WH](http://sigma.whu.edu.cn/resource.php) (<http://sigma.whu.edu.cn/resource.php>) : 武汉多时相场景变化检测数据集, 数据集主要用于进行场景变化检测的方法理论研究与验证. MtS-WH 主要包含两幅 7200 x 6000 像素大小的高分辨率遥感影像, 覆盖范围为中国武汉市汉阳区. 影像分别获取于 2002 年 2 月和 2009 年 6 月, 分辨率为 1m, 包含 4 个波段 (蓝, 绿, 红和近红外波段). 每个时相训练集包括 190 张影像, 测试集包括 1920 张影像. [?]
- [OSCD](https://rcdaudt.github.io/oscd/) (<https://rcdaudt.github.io/oscd/>) : Onera Satellite Change Detection Dataset, 多光谱 (13 波段), 可从 [这里](https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download) (<https://partage.mines-telecom.fr/index.php/s/G93tRIAgLs1sVBM/download>)
- [SZTAKI](http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) (http://web.eee.sztaki.hu/remotesensing/airchange_benchmark.html) : A Ground truth collection for change detection in optical aerial images taken with several years time differences, 用户名: sztaki, 密码: deva
- [Grupo Hiperespectral](https://gitlab.citius.usc.es/hiperespectral) (<https://gitlab.citius.usc.es/hiperespectral>) : 高光谱变化检测数据
- [ABCDdataset](https://github.com/gistairc/ABCDdataset) (<https://github.com/gistairc/ABCDdataset>)

10.4.1.2.2 自然场景

- [changedetection 2012] (<http://www.changedetection.net/>) [?]
- [changedetection 2014] (<http://www.changedetection.net/>) [?]

10.4.1.3 图像分割数据集

10.4.1.3.1 视网膜图像

- [DRIVE](http://www.isi.uu.nl/Research/Databases/DRIVE/index.php) (<http://www.isi.uu.nl/Research/Databases/DRIVE/index.php>)
- [STARE](http://cecas.clemson.edu/~ahoover/stare/) (<http://cecas.clemson.edu/~ahoover/stare/>) : STructured Analysis of the Retina
- [Tortuosity](http://bioimlab.dei.unipd.it/Retinal%20Vessel%20Tortuosity.htm) (<http://bioimlab.dei.unipd.it/Retinal%20Vessel%20Tortuosity.htm>) : Retinal Vessel Tortuosity Data Set
- [HRF](http://www5.cs.fau.de/research/data/fundus-images/) (<http://www5.cs.fau.de/research/data/fundus-images/>) : automatic segmentation on retinal fundus images
- [retina dataset](https://github.com/yiweichen04/retina_dataset) (https://github.com/yiweichen04/retina_dataset) : Retina dataset containing 1) normal 2) cataract 3) glaucoma 4) retina disease

10.4.1.4 雷达数据获取

10.4.1.4.1 公开数据

- [Alaska Satellite Facility \(ASF\)](https://www.asf.alaska.edu/) (<https://www.asf.alaska.edu/>) : 注册后可下载公开的数据, 含 ALOS PAL-SAR, Sentinel-1, UAVSAR (Airborne SAR), ERS-1, ERS-2, Seasat, SMAP, JERS-1, RADARSAT-1 等数据
- [地理空间数据云](http://www.gscloud.cn/) (<http://www.gscloud.cn/>) : 含 LANDSAT, Sentinel, MODIS, 高分系列数据, 环境系列, EO-1, 注册可下载, 分免费与商业数据, Level1 数据.
- [欧洲航天局 European Space Agency \(ESA\)](https://www.esa.int/) (<https://www.esa.int/>) : 从 [The home of ESA Earth Online data](https://earth.esa.int/web/guest/data-access) (<https://earth.esa.int/web/guest/data-access>) 下载
- [Copernicus Open Access Hub](https://scihub.copernicus.eu/) (<https://scihub.copernicus.eu/>) : Sentinel1-5 系列, 注册可下载 Level0 数据.

第十一卷补充内容

11.1 引言

涉及写作等.

11.2 写作与排版

11.2.1 LaTeX 简明教程

11.2.1.1 LaTeX 基础

- [latex writing](https://www.andy-roberts.net/writing/latex) (<https://www.andy-roberts.net/writing/latex>)

11.2.1.1.1 数学相关

11.2.1.1.1.1 定义定理证明等

导言区加入:

- `\newtheorem{环境名}{自定义标题}[主计数器]`
 - `\newtheorem{theorem}{myTheorem} [Chapter]`
 - `\newtheorem{theorem}{\hspace{2em} 定理} [Chapter]`
- `\newtheorem{环境名}{自定义标题}[主计数器]`

正文中引用:

```
\begin{theorem}  
This is a theorem  
\end{theorem}
```

11.2.1.2 LaTeX 进阶

11.2.1.3 问题解决

11.2.1.3.1 安装

11.2.1.3.2 使用

11.2.1.3.2.1 Extra alignment tab has been changed to cr

提示如下错误:

Extra alignment tab has been changed to \cr. [\end{array}] \right]\end{split}]

查看报错行源码发现，矩阵实际为 4 列，然而只指定了 3 列的参数 (ccc)：

将 ccc 改为 $cccc$ 即可.

11.2.2 文献管理与引用

11.2.2.1 简介

11.2.2.2 文献管理

11.2.2.2.1 常见文献管理软件

- [Citavi](https://www.citavi.com/) (<https://www.citavi.com/>)：文献管理, 做笔记, 记录 idea 等等超方便;

软件名称	性质	支持的操作系统	优点	缺点	说明
EndNote (http://endnote.com/)	商业	Windows, Mac	专业, 基本不影响 Word 打开速度	从 PDF 中自动抽取题录效果差, 笔记功能和 PDF 阅读功能不太好, 支持平台少, 比较贵 (X7: \$249.95)	网上有将其安装在 Linux 上的方法
Mendeley (https://www.mendeley.com/)	免费	Windows, Mac, iOS, Linux	免费, 界面美观, 基本不影响 Word 打开速度, 笔记功能好 (不保存在 PDF 中, 可同步分享), PDF 阅读功能比 EndNote 好, 自动从 PDF 中提取题录的效果好, 可以在线更新 (Google Scholar) 题录	引文 Style 格式不如 EndNote 全	需要注册免费帐号
NoteExpress (http://www.inoteexpress.com/CompanyWeb/)	商业	Windows	可在线更新题录	比较贵, 影响 Word 打开速度	•
Bibus (http://bibus-biblio.sourceforge.net/wiki/index.php/Main_Page)	免费	Windows, Mac, Linux	免费, 支持 OpenOffice 和 MS Word	引文 Style 格式由网友提供	•
Zotero (https://www.zotero.org/)	免费	Windows、Mac、Linux			•
Citavi (https://www.citavi.com/)	免费	Windows	功能丰富强大、界面友好, 特别是笔记功能强大、插入文献方便	免费版仅支持单个库 <100 个文献, 仅支持 Windows	含免费版与付费版

11.2.2.2.2 推荐的管理软件

- [Citavi](https://www.citavi.com/) (<https://www.citavi.com/>) : 文献管理, 做笔记, 记录 idea 等等超方便, [官网教程](https://www.softhead-citavi.com/blog/2328) (<https://www.softhead-citavi.com/blog/2328>)
- [JabRef](#) : 参考文献排版时使用.

11.2.2.2.3 文献引用格式

11.2.2.2.3.1 常见格式介绍

常见的有

- bibtex
- ris
- Refworks

11.2.2.2.3.2 注意事项

通过百度学术或其它文献库下载的参考文献引用格式可能会存在小问题, 如下为百度学术与 IEEE 中的同一篇论文的 bibtex 引用条目.

```
1 % by IEEE
2 @ARTICLE{7103337,
3   author={J. {Tang} and C. {Deng} and G. {Huang}},
4   journal={IEEE Transactions on Neural Networks and Learning Systems},
5   title={Extreme Learning Machine for Multilayer Perceptron},
6   year={2016},
7   volume={27},
8   number={4},
9   pages={809–821},
10  doi={10.1109/TNNLS.2015.2424995},
11  ISSN={2162-237X},
12  month={April},}

13
14 % by baiduqueshu
15 @article{Tang2017Extreme,
16   author={Tang, J. and Deng, C. and Huang, G. B.},
17   journal={IEEE Transactions on Neural Networks & Learning Systems},
18   title={Extreme Learning Machine for Multilayer Perceptron},
19   volume={27},
20   number={4},
21   pages={809–821},
22   year={2017},
23 }
```

又如

```

1 % by IEEE
2 @ARTICLE{6205396,
3   author={Z. {Xu} and X. {Chang} and F. {Xu} and H. {Zhang}},
4   journal={IEEE Transactions on Neural Networks and Learning Systems},
5   title={{L1/2} Regularization: A Thresholding Representation Theory and a Fast ↩
6   ↪Solver},
7   year={2012},
8   volume={23},
9   number={7},
10  pages={1013–1027},
11  doi={10.1109/TNNLS.2012.2197412},
12  ISSN={2162-237X},
13  month={July}, }

14 % by baiduxueshu
15 @article{Xu2012L1,
16   author={Xu, Z. and Chang, X. and Xu, F. and Zhang, H.},
17   journal={IEEE Transactions on Neural Networks & Learning Systems},
18   title={L1/2 regularization: a thresholding representation theory and a fast ↩
19   ↪solver},
20   volume={23},
21   number={7},
22   pages={1013–1027},
23   year={2012},
}

```

仔细观察可以发现，期刊名 *IEEE Transactions on Neural Networks and Learning Systems* 在百度学术中被转成 *IEEE Transactions on Neural Networks & Learning Systems*，这将导致使用 JabRef 软件转换缩写格式失败，这会使得在文章中引用效果不同，下图所示

Tang, J.; Deng, C.; Huang, G. Extreme Learning Machine for Multilayer Perceptron. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, 27, 809–821. doi:10.1109/TNNLS.2015.2424995.
Tang, J.; Deng, C.; Huang, G.B. Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks & Learning Systems* **2017**, 27, 809–821.
Xu, Z.; Chang, X.; Xu, F.; Zhang, H. $L_{1/2}$ Regularization: A Thresholding Representation Theory and a Fast Solver. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, 23, 1013–1027. doi:10.1109/TNNLS.2012.2197412.
Xu, Z.; Chang, X.; Xu, F.; Zhang, H. $L_{1/2}$ regularization: a thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks & Learning Systems* **2012**, 23, 1013–1027.

图 11.1: 文献引用效果对比

文献引用效果对比。IEEE 给出的 bibtex 格式，期刊缩写正常，且特殊符号正常显示。

11.2.2.3 文献引用

11.2.2.3.1 在文章中引用文献

11.2.2.3.1.1 引用样式

- Citation Style Language (<https://citationstyles.org/>)

11.2.2.3.1.2 期刊缩写

有些杂志中要求参考文献在引用时期刊名采用缩写形式 (Periodical Abbreviation) , 常见的缩写规范有两种, 下面举例说明:

- Full: IEEE Transactions on Geoscience and Remote Sensing
- ISO: IEEE Trans. Geosci. Remote Sens.
- MEDLINE: IEEE Trans Geosci Remote Sens

常见的期刊缩写可以在下面的列表中找到:

- CASSI (<https://cassi.cas.org/search.jsp>) : CAS Source Index (CASSI) Search Tool
- abbrv jabref (<http://abbrv.jabref.org/>) : JabRef 软件可以随意切换缩写格
- abbrv jabref github (<https://github.com/antsfamily/abbrv.jabref.org>)
- abbrv jabref github journals (<https://github.com/JabRef/abbrv.jabref.org/tree/master/journals>)
- abbrv CSL github (<https://github.com/citation-style-language/abbreviations>)

文献管理软件 JabRef 提供了丰富的文献缩写库, 并且可以很方便的在缩写格式间切换, 可在 Tools 菜单中找到, 如下动图所示:

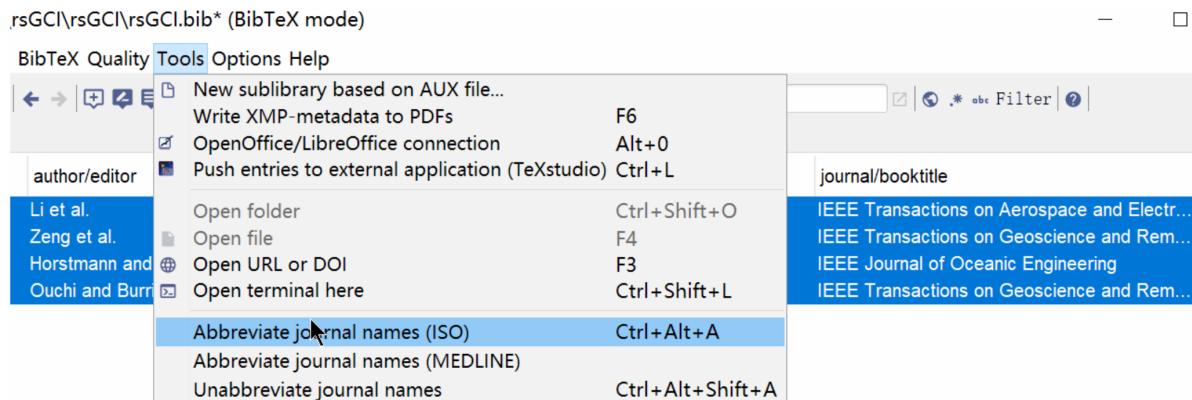


图 11.2: Periodical Abbreviation Switch with JabRef

Periodical Abbreviation Switch with JabRef

提示:

1. 切换完直接保存即可对 bib 文献库中的文献期刊缩写进行转换.

2. 使用百度学术等一些软件下载的 bibtex 文献格式可能会不正确, 会遇到缩写转换不成功的情况, 参见文献引用格式 (页 628).

11.3 生活常识

11.3.1 电子电器篇

11.3.1.1 声音设备

11.3.1.1.1 耳机

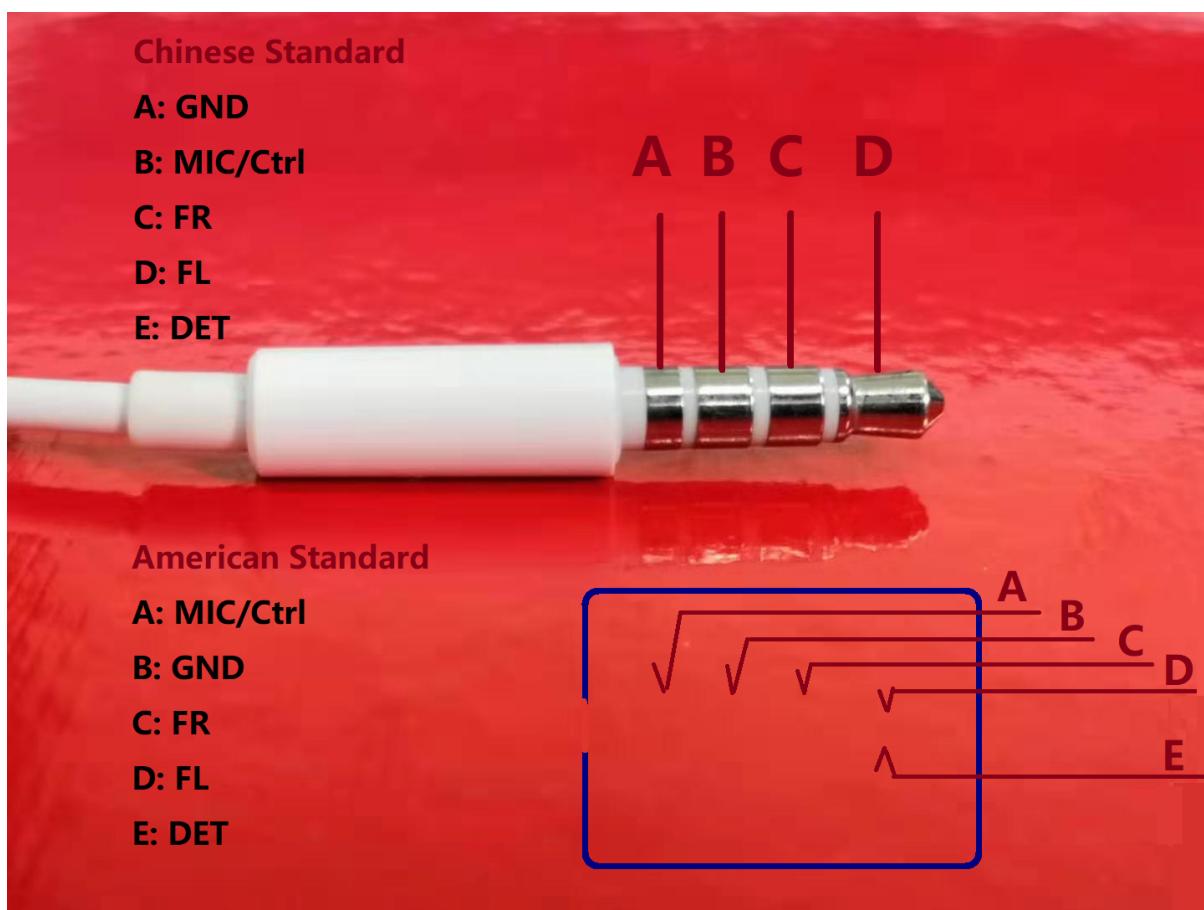


图 11.3: 耳机接口示意图

常见的四线式耳机接口一般有两种标准, 中国标准和美国标准, 不同的只是地线 (GND) 与麦克线 (MIC) 的顺序.

注解: 如何检测耳机是否插入

图中, DET 为耳机检测线, 当不插入耳机时, 该引脚为高电平; 当插入耳机时, 该引脚通过左声道耳机接地, 由于耳机电阻一般为几十欧姆, 故此时 DET 被拉为低电平, 此信号触发微处理器中断, 由此检测耳机插入.

11.3.1.2 红外遥控

11.3.1.2.1 红外遥控原理

11.3.1.2.1.1 红外发射

11.3.1.2.1.2 红外接收

11.3.1.2.2 手机方案

11.3.1.2.2.1 3.5mm 耳机无源方案

有关耳机的知识参见[耳机](#) (页 631) 小节.

11.3.1.2.2.2 3.5mm 耳机有源方案

11.3.1.2.2.3 USB 有源方案

11.3.1.2.2.4 应用软件

CHAPTER 12

名词术语

Architecture 体系结构

Cross Compiling 简单地说, 就是在一个平台上生成另一个平台上的可执行代码. 包含两个概念: 体系结构 (Architecture) 和操作系统 (Operating System). 同一体系结构可以运行不同的操作系统; 同样, 同一操作系统也可以在不同的体系结构上运行. 举例来说, 我们常说的 x86 Linux 平台实际上是 Intel x86 体系结构和 Linux for x86 操作系统的统称; 而 x86 WinNT 平台实际上是 Intel x86 体系结构和 Windows NT for x86 操作系统的简称. 对于嵌入式 Linux 开发中的交叉编译, 一般是指, 在 Linux PC 机上, 利用交叉编译器 *arm-linux-gcc*, 编译生成 Linux ARM 上的可执行程序.

DMIPS 即 Dhrystone Million Instructions executed Per Second, Dhrystone 是一种整数运算测试程序, DMIPS 表示了在 Dhrystone 这样一种测试方法下的 MIPS. 举例来说, 如果某处理器性能为: 2DMIPS/MHZ, 那么对于主频为 1000MHz 即 1GHz 的处理器, 其计算能力为 2000DMIPS.

CPU 性能评估采用综合测试程序, 较流行的有 Whetstone 和 Dhrystone 两种. Dhrystone 主要用于测整数计算能力, 计算单位就是 DMIPS. 采用 Whetstone 主要用于测浮点计算能力, 计算单位就是 MFLOPS. DMIPS 只适宜于评估标量机, 不能用于评估向量机. 而 MFLOPS 则比较适用于衡量向量机的性能.

Embedded Operating System 嵌入式操作系统 (Embedded Operating System, 简称: EOS) 是指用于嵌入式硬件系统的操作系统. 嵌入式操作系统是一种用途广泛的系统软件, 通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等. 嵌入式操作系统负责嵌入式系统的全部软、硬件资源的分配、任务调度, 控制、协调并发活动. 它必须体现其所在系统的特征, 能够通过装卸某些模块来达到系统所要求的功能. 目前在嵌入式领域广泛使用的操作系统有: 嵌入式实时操作系统 μC/OS-II、嵌入式 Linux、Windows Embedded、VxWorks 等, 以及应用在智能手机和平板电脑的 Android、iOS 等.

Embedded Real-time Operation System 嵌入式实时操作系统 (Embedded Real-time Operation System, EROS), 具备实时性的嵌入式操作系统.

GPU 图形处理单元 (Graphic Processing Unit, GPU)

GUI 图形用户界面 (Graphical User Interface, 简称 GUI, 又称图形用户接口) 是指采用图形方式显示的计算机操作用户界面.

Independent Component Analysis 独立成分分析 (Independent Component Analysis (http://en.volupedia.org/wiki/Independent_component_analysis), ICA), 在信号处理中, 独立分量分析是一种将多变量信号分离为加性子成分的计算方法. 通过假设子成分是非高斯信号, 并且它们在统计上彼此独立来实现的. ICA 是盲源分离的一种特殊情况, 其基本思想是从一组混合的观测信号中分离出独立信号, 比如在一个大房间里, 很多人同时在说话, 样本是这个房间里各个位置的一段录音, ICA 可以从这些混合的录音中分离出每个人独立的说话的声音. ICA 认为观测信号是若干个统计独立的分量的线性组合, ICA 要做的是一个解混过程.

Markup Language 标记语言 (也称置标语言、标记语言、标志语言、标识语言) 是一种将文本 (Text) 以及文本相关的其他信息结合起来, 展现出关于文档结构和数据处理细节的计算机文字编码. 与文本相关的其他信息 (包括例如文本的结构和表示信息等) 与原来的文本结合在一起, 但是使用标记 (markup) 进行标识. 当今广泛使用的标记语言是超文本标记语言 (HyperText Markup Language, HTML) 和可扩展标记语言 (eXtensible Markup Language, XML). 标记语言广泛应用于网页和网络应用程序. 标记最早用于出版业, 是作者、编辑以及出版商之间用于描述出版作品的排版格式所使用的.

MIPS 每秒百万条指令 (Million Instructions executed Per Second, MIPS), 用来计算同一秒内系统的处理能力, 即每秒执行了多少百万条指令.

OpenCL 开放计算语言 (Open Computing Language, OpenCL), 是一个为异构平台编写程序的框架, 此异构平台可由 CPU、GPU、DSP、FPGA 或其他类型的处理器与硬件加速器所组成. OpenCL 由一门用于编写 kernels (在 OpenCL 设备上运行的函数) 的语言 (基于 C99) 和一组用于定义并控制平台的 API 组成. OpenCL 提供了基于任务分区和数据分区的并行计算机制.

OpenCL 类似于另外两个开放的工业标准 OpenGL 和 OpenAL, 这两个标准分别用于三维图形和计算机音频方面. OpenCL 扩充了 GPU 图形生成之外的能力. OpenCL 由非盈利性技术组织 Khronos Group 掌管.

Principal Component Analysis 主成分分析 (Principal Component Analysis (http://en.volupedia.org/wiki/Principal_component_analysis), PCA) 是一种统计过程, 它使用正交变换将一组可能相关的变量观测值 (每个实体具有不同的数值) 转换为一组线性不相关变量的值, 称为主成分. 这种转换的定义方式是, 第一个主成分具有最大的可能方差 (即, 尽可能多地解释数据中的可变性), 并且每个后续成分在其与前面的成分正交的约束下依次具有最大的可能方差. 结果向量 (每个向量都是变量的线性组合, 包含 n 个观测值) 是不相关的正交基集. PCA 对原始变量的相对尺度很敏感.

Processing System 处理系统 (Processing System, PS), 在 Xilinx Zynq 系列设备中, SOC 被分成 PS (ARM 部分) 与 PL (FPGA 部分).

Programmable Logic 可编程逻辑 (Programmable Logic, PL), 在 Xilinx Zynq 系列设备中, SOC 被分成 PS (ARM 部分) 与 PL (FPGA 部分).

Proper Subset 真子集

Real Time multi-tasking Operation System 实时多任务操作系统 (Real Time multi-tasking Operation System, RTOS). 在嵌入式应用中通常注重其实时性.

Set 集合 (Set) 是指具有某种特定性质的具体的或抽象的对象构成的总体.

Stream Multiprocessor 流处理器 (Stream Multiprocessor, SM)

Stream Processor 流处理器 (Stream Processor, SP)

Subset 子集

Thread Processor Cluster 线程处理器簇 (Thread Processor Cluster, TPC)

XADC 赛灵思模拟到数字转换器 (Xilinx Analogue to Digital Converter, XADC)

xdc 赛灵思设计约束 (Xilinx Design Constraints, XDC)

Xilinx Software Development Kit 赛灵思软件开发套件 (Xilinx Software Development Kit, SDK)

CHAPTER 13

Indices and tables

- genindex
- search

Bibliography

- [1] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.
- [2] Jerome H. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, pages 817–823, 1981.
- [3] Jerome Friedman. Classification and multiple regression through projection pursuit. Technical Report, Stanford Linear Accelerator Center and Department of Statistics, 01 1985.
- [4] Peter Hall. On projection pursuit regression. *The Annals of Statistics*, 17(2):573–588, 1989. URL: <http://www.jstor.org/stable/2241571>.
- [5] M. Maechler, D. Martin, J. Schimert, M. Csoppenszky, and J. N. Hwang. Projection pursuit learning networks for regression. In [1990] *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, volume, 350–358. Nov 1990. doi:[10.1109/TAI.1990.130362](https://doi.org/10.1109/TAI.1990.130362) (<https://doi.org/10.1109/TAI.1990.130362>).
- [6] Hwang, J N, Lay, S R, . Maechler, M., Martin, R D, and . Schimert, J. Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on Neural Networks*, 5(3):342–353, May 1994. doi:[10.1109/72.286906](https://doi.org/10.1109/72.286906) (<https://doi.org/10.1109/72.286906>).
- [7] Y. Zhao and C. G. Atkeson. Implementing projection pursuit learning. *IEEE Transactions on Neural Networks*, 7(2):362–373, March 1996. doi:[10.1109/72.485672](https://doi.org/10.1109/72.485672) (<https://doi.org/10.1109/72.485672>).
- [8] Amith Singhee and Rob A. Rutenbar. *SiLVR: Projection Pursuit for Response Surface Modeling*, pages 1–57. Springer Netherlands, Dordrecht, 2009. URL: https://doi.org/10.1007/978-90-481-3100-6_1, doi:[10.1007/978-90-481-3100-6_1](https://doi.org/10.1007/978-90-481-3100-6_1) (https://doi.org/10.1007/978-90-481-3100-6_1).
- [9] Héctor Corrada Bravo. Neural networks. 2017. URL: https://github.com/hcorrada/dscert-mldm/blob/3b3695c5ce02cc8b177e9b01e4ba8cb55f589561/materials/lectures/neural_nets/neural_nets.pdf.
- [10] Pavel Komarov. The gory details of projection pursuit. 2018. URL: <https://github.com/pavelkomarov/projection-pursuit/blob/master/doc/math.pdf>.

- [3] C. Petr. Online learning of neural takagi-sugeno fuzzy model. In *NAFIPS 2005 - 2005 Annual Meeting of the North American Fuzzy Information Processing Society*, 478–483. 2005. doi:[10.1109/NAFIPS.2005.1548582](https://doi.org/10.1109/NAFIPS.2005.1548582) (<https://doi.org/10.1109/NAFIPS.2005.1548582>).
- [2] Barnabas Bede. *Mathematics of Fuzzy Sets and Fuzzy Logic*. Springer, Cham, 2013. ISBN 978-3-642-35221-8. doi:[10.1007/978-3-642-35221-8](https://doi.org/10.1007/978-3-642-35221-8) (<https://doi.org/10.1007/978-3-642-35221-8>).
- [3] Enric Trillas and Luka Eciolaza. *Fuzzy Logic: An Introductory Course for Engineering Students*. Springer, Cham, 2015. ISBN 978-3-319-14203-6. doi:[10.1007/978-3-319-14203-6](https://doi.org/10.1007/978-3-319-14203-6) (<https://doi.org/10.1007/978-3-319-14203-6>).
- [4] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. URL: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>, doi:[10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X) ([https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X)).
- [5] Zdzisław Pawlak. Rough sets. *International Journal of Computer & Information Sciences*, 11(5):341–356, 1982. doi:[10.1007/BF01001956](https://doi.org/10.1007/BF01001956) (<https://doi.org/10.1007/BF01001956>).
- [5] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, 1985. doi:[10.1109/TSMC.1985.6313399](https://doi.org/10.1109/TSMC.1985.6313399) (<https://doi.org/10.1109/TSMC.1985.6313399>).
- [6] M. Sugeno and G.T Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28(1):15–33, 1988. doi:[10.1016/0165-0114\(88\)90113-3](https://doi.org/10.1016/0165-0114(88)90113-3) ([https://doi.org/10.1016/0165-0114\(88\)90113-3](https://doi.org/10.1016/0165-0114(88)90113-3)).
- [8] X. Gu and S. Wang. Bayesian takagi–sugeno–kang fuzzy model and its joint learning of structure identification and parameter estimation. *IEEE Transactions on Industrial Informatics*, 14(12):5327–5337, 2018. doi:[10.1109/TII.2018.2813977](https://doi.org/10.1109/TII.2018.2813977) (<https://doi.org/10.1109/TII.2018.2813977>).
- [1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. now, 2011. ISBN 9781601984609. URL: <https://ieeexplore.ieee.org/document/8186925>.
- [2] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17 –40, 1976. URL: <http://www.sciencedirect.com/science/article/pii/0898122176900031>, doi:[https://doi.org/10.1016/0898-1221\(76\)90003-1](https://doi.org/10.1016/0898-1221(76)90003-1) ([https://doi.org/10.1016/0898-1221\(76\)90003-1](https://doi.org/10.1016/0898-1221(76)90003-1)).
- [3] R. Glowinski and A. Marroco. Sur l’ approximation, par éléments finis d’ ordre un, et la résolution, par penalisation-dualité, d’ une classe de problèmes de dirichlet non linéaires. *Revue Française d’ Automatique, Informatique et Recherche Opérationnelle*, 9(2):41–76, 1975.
- [4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM JOURNAL ON IMAGING SCIENCES*, 2009.
- [5] Mario A. T. Figueiredo, Jose M. Bioucas-Dias, and Manya v. Afonso. Fast frame-based image deconvolution using variable splitting and constrained optimization. In *SSP ‘09*, 109–112. [Piscataway, N.J.], 2009. IEEE. doi:[10.1109/SSP.2009.5278628](https://doi.org/10.1109/SSP.2009.5278628) (<https://doi.org/10.1109/SSP.2009.5278628>).
- [6] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19(9):2345–2356, 2010. doi:[10.1109/TIP.2010.2047910](https://doi.org/10.1109/TIP.2010.2047910) (<https://doi.org/10.1109/TIP.2010.2047910>).

- [7] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. A fast algorithm for the constrained formulation of compressive image reconstruction and other linear inverse problems. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4034–4037. 2010. doi:[10.1109/ICASSP.2010.5495758](https://doi.org/10.1109/ICASSP.2010.5495758) (<https://doi.org/10.1109/ICASSP.2010.5495758>).
- [8] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011. doi:[10.1109/TIP.2010.2076294](https://doi.org/10.1109/TIP.2010.2076294) (<https://doi.org/10.1109/TIP.2010.2076294>).
- [1] R C Dewar. Maximum entropy production and the fluctuation theorem. *Journal of Physics A General Physics*, 38(21):L371, 2005.
- [1] Y. Wu and V. S. Batista. Matching-pursuit for simulations of quantum processes. *Jcp*, 118:6720–6724, April 2003. doi:[10.1063/1.1560636](https://doi.org/10.1063/1.1560636) (<https://doi.org/10.1063/1.1560636>).
- [2] Yan Wu, Mihaela Rosca, and Timothy Lillicrap. Deep compressed sensing. 2019. URL: <https://arxiv.org/abs/1905.06723>.
- [3] Richard G. Baraniuk and Michael B. Wakin. Random projections of smooth manifolds. *Foundations of Computational Mathematics*, 9(1):51–77, 2009. doi:[10.1007/s10208-007-9011-z](https://doi.org/10.1007/s10208-007-9011-z) (<https://doi.org/10.1007/s10208-007-9011-z>).
- [4] Marco F. Duarte, Mark A. Davenport, Michael B. Wakin, Jason N. Laska, Dharmpal Takhar, Kevin F. Kelly, and Richard G. Baraniuk. Multiscale random projections for compressive classification. In IEEE Staff, editor, *2007 IEEE International Conference on Image Processing*, VI –161–VI –164. Piscataway, Jan. 2007. IEEE. doi:[10.1109/ICIP.2007.4379546](https://doi.org/10.1109/ICIP.2007.4379546) (<https://doi.org/10.1109/ICIP.2007.4379546>).
- [5] Yonina C. Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, Cambridge and New York, 2012. ISBN 9780511794308. doi:[10.1017/CBO9780511794308](https://doi.org/10.1017/CBO9780511794308) (<https://doi.org/10.1017/CBO9780511794308>).
- [6] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer New York, New York, NY, 2013. ISBN 978-0-8176-4947-0. doi:[10.1007/978-0-8176-4948-7](https://doi.org/10.1007/978-0-8176-4948-7) (<https://doi.org/10.1007/978-0-8176-4948-7>).
- [7] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008. URL: <http://www.sciencedirect.com/science/article/pii/S1631073X08000964>, doi:[10.1016/j.crma.2008.03.014](https://doi.org/10.1016/j.crma.2008.03.014) (<https://doi.org/10.1016/j.crma.2008.03.014>).
- [8] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi:[10.1109/T-C.1974.223784](https://doi.org/10.1109/T-C.1974.223784) (<https://doi.org/10.1109/T-C.1974.223784>).
- [9] S. G. Mallat and. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, Dec 1993. doi:[10.1109/78.258082](https://doi.org/10.1109/78.258082) (<https://doi.org/10.1109/78.258082>).
- [10] *Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition*, volume, Nov 1993. doi:[10.1109/ACSSC.1993.342465](https://doi.org/10.1109/ACSSC.1993.342465) (<https://doi.org/10.1109/ACSSC.1993.342465>).
- [11] Joel A. Tropp and Anna C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007. doi:[10.1109/TIT.2007.909108](https://doi.org/10.1109/TIT.2007.909108) (<https://doi.org/10.1109/TIT.2007.909108>).

- [12] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005. doi:[10.1109/TIT.2005.858979](https://doi.org/10.1109/TIT.2005.858979) (<https://doi.org/10.1109/TIT.2005.858979>).
- [13] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
- [14] S. D. Babacan, R. Molina, and A. K. Katsaggelos. Bayesian compressive sensing using laplace priors. *IEEE Transactions on Image Processing*, 19(1):53–63, 2010. doi:[10.1109/TIP.2009.2032894](https://doi.org/10.1109/TIP.2009.2032894) (<https://doi.org/10.1109/TIP.2009.2032894>).
- [15] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008. doi:[10.1109/TSP.2007.914345](https://doi.org/10.1109/TSP.2007.914345) (<https://doi.org/10.1109/TSP.2007.914345>).
- [1] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. doi:[10.1109/TPAMI.2015.2389824](https://doi.org/10.1109/TPAMI.2015.2389824) (<https://doi.org/10.1109/TPAMI.2015.2389824>).
- [2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv e-prints*, 2016.
- [3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv e-prints*, 2017.
- [4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv e-prints*, 2018.
- [5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. doi:[10.1109/TPAMI.2017.2699184](https://doi.org/10.1109/TPAMI.2017.2699184) (<https://doi.org/10.1109/TPAMI.2017.2699184>).
- [1] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13(2):464–471, 2002. doi:[10.1109/72.991432](https://doi.org/10.1109/72.991432) (<https://doi.org/10.1109/72.991432>).
- [2] C.-fu Lin and S.-de Wang. Fuzzy support vector machines with automatic membership setting. In Lipo Wang, editor, *Support Vector Machines: Theory and Applications*, pages 233–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. doi:[10.1007/10984697_11](https://doi.org/10.1007/10984697_11) (https://doi.org/10.1007/10984697_11).
- [3] Xiufeng Jiang, Zhang Yi, and Jian Cheng Lv. Fuzzy svm with a new fuzzy membership function. *Neural Computing & Applications*, 15(3):268–276, 2006. doi:[10.1007/s00521-006-0028-z](https://doi.org/10.1007/s00521-006-0028-z) (<https://doi.org/10.1007/s00521-006-0028-z>).
- [9] Enhui Zheng, Jinyong Liu, Huijuan Lu, Ling Wang, and Le Chen. A new fuzzy extreme learning machine for regression problems with outliers or noises. In Hiroshi Motoda, Zhaojun Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, 524–534. Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi:[10.1007/BF00994018](https://doi.org/10.1007/BF00994018) (<https://doi.org/10.1007/BF00994018>).
- [4] H. Rong, G. Huang, N. Sundararajan, and P. Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, 2009. doi:[10.1109/TSMCB.2008.2010506](https://doi.org/10.1109/TSMCB.2008.2010506) (<https://doi.org/10.1109/TSMCB.2008.2010506>).

- [2] Guang Bin Huang, Qin Yu Zhu, and Chee Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
- [3] Weiwei Zong and Guang-Bin Huang. Face recognition based on extreme learning machine. *Neurocomputing*, 74:2541–2551, 2011. doi:[10.1016/j.neucom.2010.12.041](https://doi.org/10.1016/j.neucom.2010.12.041) (<https://doi.org/10.1016/j.neucom.2010.12.041>).
- [4] G. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, 10(2):18–29, 2015. doi:[10.1109/MCI.2015.2405316](https://doi.org/10.1109/MCI.2015.2405316) (<https://doi.org/10.1109/MCI.2015.2405316>).
- [5] J. Tang, C. Deng, and G. B. Huang. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks & Learning Systems*, 27(4):809–821, 2017.
- [6] C. Ouyang, T. Kao, Y. Cheng, C. Wu, C. Tsai, and M. Wu. An improved fuzzy extreme learning machine for classification and regression. In *2016 International Conference on Cybernetics, Robotics and Control (CRC)*, 91–94. 2016. doi:[10.1109/CRC.2016.028](https://doi.org/10.1109/CRC.2016.028) (<https://doi.org/10.1109/CRC.2016.028>).
- [7] W. B. Zhang and H. B. Ji. Fuzzy extreme learning machine for classification. *Electronics Letters*, 49(7):448–450, 2013. doi:[10.1049/el.2012.3642](https://doi.org/10.1049/el.2012.3642) (<https://doi.org/10.1049/el.2012.3642>).
- [8] Shihabudheen KV and G.N. Pillai. Regularized extreme learning adaptive neuro-fuzzy algorithm for regression and classification. *Knowledge-Based Systems*, 127:100–113, 2017. URL: <http://www.sciencedirect.com/science/article/pii/S0950705117301831>, doi:[10.1016/j.knosys.2017.04.007](https://doi.org/10.1016/j.knosys.2017.04.007) (<https://doi.org/10.1016/j.knosys.2017.04.007>).
- [9] Enhui Zheng, Jinyong Liu, Huijuan Lu, Ling Wang, and Le Chen. A new fuzzy extreme learning machine for regression problems with outliers or noises. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, 524–534. Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [1] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv e-prints*, 2016.
- [2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:[10.1109/5.726791](https://doi.org/10.1109/5.726791) (<https://doi.org/10.1109/5.726791>).
- [3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016. doi:[10.1109/TPAMI.2015.2439281](https://doi.org/10.1109/TPAMI.2015.2439281) (<https://doi.org/10.1109/TPAMI.2015.2439281>).
- [4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60:84–90, 2012.
- [5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.
- [6] C. Szegedy, and, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. 2015. doi:[10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594) (<https://doi.org/10.1109/CVPR.2015.7298594>).
- [7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3:2672–2680, 2014.

- [1] J. - R. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, 1993. doi:[10.1109/21.256541](https://doi.org/10.1109/21.256541) (<https://doi.org/10.1109/21.256541>).
 - [2] C. Isik and M. Farrokhi. Recurrent neuro-fuzzy systems. In Can Islk and Valerie Cross, editors, *1997 Annual Meeting of the North American Fuzzy Information Processing Society–NAFIPS*, 362–366. [New York] and Piscataway, N.J., 1997. Institute of Electrical and Electronics Engineers and IEEE Service Center. doi:[10.1109/NAFIPS.1997.624067](https://doi.org/10.1109/NAFIPS.1997.624067) (<https://doi.org/10.1109/NAFIPS.1997.624067>).
 - [3] C. Petr. Online learning of neural takagi-sugeno fuzzy model. In *NAFIPS 2005 - 2005 Annual Meeting of the North American Fuzzy Information Processing Society*, 478–483. 2005. doi:[10.1109/NAFIPS.2005.1548582](https://doi.org/10.1109/NAFIPS.2005.1548582) (<https://doi.org/10.1109/NAFIPS.2005.1548582>).
 - [4] H. Rong, G. Huang, N. Sundararajan, and P. Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, 2009. doi:[10.1109/TSMCB.2008.2010506](https://doi.org/10.1109/TSMCB.2008.2010506) (<https://doi.org/10.1109/TSMCB.2008.2010506>).
 - [5] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, 1985. doi:[10.1109/TSMC.1985.6313399](https://doi.org/10.1109/TSMC.1985.6313399) (<https://doi.org/10.1109/TSMC.1985.6313399>).
 - [6] M. Sugeno and G.T Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28(1):15–33, 1988. doi:[10.1016/0165-0114\(88\)90113-3](https://doi.org/10.1016/0165-0114(88)90113-3) ([https://doi.org/10.1016/0165-0114\(88\)90113-3](https://doi.org/10.1016/0165-0114(88)90113-3)).
-
- [1] M. P. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. Curran Associates, Inc, 2010. URL: <http://papers.nips.cc/paper/3923-self-paced-learning-for-latent-variable-models.pdf>.
 - [2] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander Hauptmann. Easy samples first: self-paced reranking for zero-example multimedia search. In *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*. 11 2014. doi:[10.1145/2647868.2654918](https://doi.org/10.1145/2647868.2654918) (<https://doi.org/10.1145/2647868.2654918>).
 - [3] E. Sangineto, M. Nabi, D. Culibrk, and N. Sebe. Self paced deep learning for weakly supervised object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3):712–725, 2019. doi:[10.1109/TPAMI.2018.2804907](https://doi.org/10.1109/TPAMI.2018.2804907) (<https://doi.org/10.1109/TPAMI.2018.2804907>).
 - [4] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2379–2386. 2013. doi:[10.1109/CVPR.2013.308](https://doi.org/10.1109/CVPR.2013.308) (<https://doi.org/10.1109/CVPR.2013.308>).
 - [5] J. Yang, X. Wu, J. Liang, X. Sun, M. Cheng, P. L. Rosin, and L. Wang. Self-paced balance learning for clinical skin disease recognition. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2019. doi:[10.1109/TNNLS.2019.2917524](https://doi.org/10.1109/TNNLS.2019.2917524) (<https://doi.org/10.1109/TNNLS.2019.2917524>).
 - [6] Te Pi, Xi Li, Zhongfei Zhang, Deyu Meng, Fei Wu, Jun Xiao, and Yueting Zhuang. Self-paced boost learning for classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI '16*, 1932–1938. AAAI Press [download], 2016. URL: <http://dl.acm.org/citation.cfm?id=3060832.3060891>.
-
- [1] 郭爽. 复杂环境中的辐射源信号分选技术研究. PhD thesis, 西安电子科技大学, 2017.
 - [2] 范亚敏. 雷达辐射源模拟技术研究. PhD thesis, 西安电子科技大学, 2013.

- [1] 同武勤 and 樊祥. 毫米波合成孔径雷达成像技术. 火力与指挥控制, 31(3):78–封三, 2006.
- [2] 魏志强, 李春化, 周子超, 苏小敏, 李雅梅, and 王乐. 毫米波安检成像雷达设计. 火控雷达技术, 42(3):5–10, 2013.
- [3] Anatoliy O. Borysenko, Dmitriy L. Sostanovsky, and Elen S. Borysenko. Portable imaging uwb radar system with two-element receiving array. In Carl E. Baum, Alexander P. Stone, and J. Scott Tyo, editors, *Ultra-Wideband Short-Pulse Electromagnetics 8*, pages 153–160. Springer New York, New York, NY, 2007. doi:[10.1007/978-0-387-73046-2_21](https://doi.org/10.1007/978-0-387-73046-2_21) (https://doi.org/10.1007/978-0-387-73046-2_21).
- [4] Margaret Cheney and Brett Borden. Synthetic aperture radar imaging. In Otmar Scherzer, editor, *Handbook of mathematical methods in imaging*, volume 5 of Springer reference, pages 655–690. Springer, New York, 2011. doi:[10.1007/978-0-387-92920-0_15](https://doi.org/10.1007/978-0-387-92920-0_15) (https://doi.org/10.1007/978-0-387-92920-0_15).
- [5] G. Brooker, J. Martinez, and R. Hennessey. Millimetre wave radar imaging of mining vehicles. In *The 7th European Radar Conference*, 284–287. 2010.
- [6] D. S. Goshi, Y. Liu, K. Mai, L. Bui, and Y. Shih. Recent advances in 94 ghz fmcw imaging radar development. In *2009 IEEE MTT-S International Microwave Symposium digest*, 77–80. [Piscataway, N.J.], 2009. IEEE. doi:[10.1109/MWSYM.2009.5165636](https://doi.org/10.1109/MWSYM.2009.5165636) (<https://doi.org/10.1109/MWSYM.2009.5165636>).
- [7] K. Jin, W. Chang, Z. Li, and J. Yang. Imaging of space targets in fmcw-isar. In *IET International Radar Conference 2013*, 0701. Stevenage, England, 2013. IET. doi:[10.1049/cp.2013.0214](https://doi.org/10.1049/cp.2013.0214) (<https://doi.org/10.1049/cp.2013.0214>).
- [8] D. Korneev, L. Bogdanov, A. Nalivkin, and S. Berezin. 3d imaging system based on fmcw millimeter wave radar. In Kenneth J. Button and G. R. Neil, editors, *IRMMW-THz 2005*, 367–368. Piscataway, N.J., 2005. IEEE. doi:[10.1109/ICIMW.2005.1572565](https://doi.org/10.1109/ICIMW.2005.1572565) (<https://doi.org/10.1109/ICIMW.2005.1572565>).
- [9] S. A. Kuznetsov, S. N. Makarov, V. N. Koshelenko, M. A. Astafev, and A. v. Arzhannikov. 140 ghz active imaging systems based on fmcw radar. In *39th International Conference on Infrared, Millimeter, and Terahertz Waves (IRMMW-THz), 2014*, 1–2. Piscataway, NJ, 2014. IEEE. doi:[10.1109/IRMMW-THz.2014.6956097](https://doi.org/10.1109/IRMMW-THz.2014.6956097) (<https://doi.org/10.1109/IRMMW-THz.2014.6956097>).
- [10] David Macfarlane and Duncan Robertson. A 94ghz real aperture 3d imaging radar. In *Proceedings of the 3rd European Radar Conference*, 154–157. London, 2006. Horizon House Publications. doi:[10.1109/EURAD.2006.280297](https://doi.org/10.1109/EURAD.2006.280297) (<https://doi.org/10.1109/EURAD.2006.280297>).
- [11] N. Pohl, T. Jaeschke, and M. Vogt. An sige-chip-based 80 ghz fmcw-radar system with 25 ghz bandwidth for high resolution imaging. In *2013 14th International Radar Symposium (IRS)*, volume 1, 239–244. 2013.
- [12] P. Alizadeh, C. Parini, and K. Z. Rajab. A low-cost fmcw radar front end for imaging at 24 ghz to 33 ghz. In *2015 Loughborough Antennas Propagation Conference (LAPC)*, 1–4. 2015. doi:[10.1109/LAPC.2015.7366007](https://doi.org/10.1109/LAPC.2015.7366007) (<https://doi.org/10.1109/LAPC.2015.7366007>).
- [13] James Schellenberg, Richard Chedester, and John McCoy. Multi-channel receiver for an e-band fmcw imaging radar. In *IEEE MTT-S International Microwave Symposium, 2007*, 1359–1362. Piscataway, NJ, 2007. IEEE Service Center. doi:[10.1109/MWSYM.2007.380465](https://doi.org/10.1109/MWSYM.2007.380465) (<https://doi.org/10.1109/MWSYM.2007.380465>).

- [14] S. Xu, J. Wang, and A. Yarovoy. Super resolution doa for fmcw automotive radar imaging. In *2018 IEEE Conference on Antenna Measurements Applications (CAMA)*, 1–4. 2018. doi:[10.1109/CAMA.2018.8530609](https://doi.org/10.1109/CAMA.2018.8530609) (<https://doi.org/10.1109/CAMA.2018.8530609>).
- [15] Z. Peng, J. Muñoz-Ferreras, R. Gómez-García, L. Ran, and C. Li. 24-ghz biomedical radar on flexible substrate for isar imaging. In *2016 IEEE MTT-S International Wireless Symposium (IWS)*, 1–4. 2016. doi:[10.1109/IEEE-IWS.2016.7585400](https://doi.org/10.1109/IEEE-IWS.2016.7585400) (<https://doi.org/10.1109/IEEE-IWS.2016.7585400>).
- [1] Lan G.Cumming Frank H.Wong. 合成孔径雷达成像: 算法与实现. 电子工业出版社, 2012. URL: <http://sar.ece.ubc.ca>.
- [2] R. Baraniuk and P. Steeghs. Compressive radar imaging. In *2007 IEEE Radar Conference*, volume, 128–133. April 2007. doi:[10.1109/RADAR.2007.374203](https://doi.org/10.1109/RADAR.2007.374203) (<https://doi.org/10.1109/RADAR.2007.374203>).
- [3] V. M. Patel, G. R. Easley, D. M. Healy, Jr., and R. Chellappa. Compressed synthetic aperture radar. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):244–254, April 2010. doi:[10.1109/JSTSP.2009.2039181](https://doi.org/10.1109/JSTSP.2009.2039181) (<https://doi.org/10.1109/JSTSP.2009.2039181>).
- [4] R. Kang and B. Wang G. Qu. Two effective strategies for complex domain compressive sensing. *Circuits Systems and Signal Processing*, 35(9):3380–3392, 2016.
- [1] Lan G.Cumming Frank H.Wong. 合成孔径雷达成像: 算法与实现. 电子工业出版社, 2012. URL: <http://sar.ece.ubc.ca>.
- [2] Cumming, Ian G., and and Frank Hay-chee Wong. *Digital processing of synthetic aperture radar data: algorithms and implementation*. Artech House, Boston, 2005.
- [1] D. P. Bertsekas. *Constrained optimization and Lagrange Multiplier methods*. Academic Press, Inc., 1982.
- [2] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv e-prints*, 2010.