信号処理特論 第13回課題

視覚認知システム研究室
学籍番号:2233730006　千畑 颯也

2024年7月31日

課題3

平均0，分散1のガウス乱数$x(n)(n = 0, 1, \cdots, 999)$を入力信号として，出力を

$$y(n) = \sum_{m=0}^{N-1} g(n-m)x(m) + \epsilon(n) = \sum_{m=0}^{N-1} g(m)x(n-m) + \epsilon(n)$$

作る．ただし，$N = 5$であり，$(g(0), g(1), g(2), g(3), g(4)) = (1, -0.9, 0.8, -0.7, 0.6)$，$x(m) = 0(m < 0)$とする．また，$\epsilon(n)$は$x(n)$とは独立なガウス乱数を1/20倍したものである．$x(n)$と$y(n)$を用いて最小2乗法により$(g(0), g(1), g(2), g(3), g(4))$を求めよ．

以下の式をガウスザイゼル法を用いて解くことで$g(0) \sim g(4)$を求める．

$$\begin{pmatrix} <x_n x_n> & <x_n x_{n-1}> & \cdots & <x_n x_{n-4}> \\ <x_{n-1} x_n> & <x_{n-1} x_{n-1}> & \cdots & <x_{n-1} x_{n-4}> \\ <x_{n-2} x_n> & <x_{n-2} x_{n-1}> & \cdots & <x_{n-2} x_{n-4}> \\ <x_{n-3} x_n> & <x_{n-3} x_{n-1}> & \cdots & <x_{n-3} x_{n-4}> \\ <x_{n-4} x_n> & <x_{n-4} x_{n-1}> & \cdots & <x_{n-4} x_{n-4}> \end{pmatrix} \begin{pmatrix} g(0) \\ g(1) \\ g(2) \\ g(3) \\ g(4) \end{pmatrix} = \begin{pmatrix} <x_n y_n> \\ <x_{n-1} y_n> \\ <x_{n-2} y_n> \\ <x_{n-3} y_n> \\ <x_{n-4} y_n> \end{pmatrix}$$

算出した結果を以下に示す．表1より，最小2乗法を用いて正しく$g(0) \sim$

表1: 真値と算出した値

	真値	算出した値
$g(0)$	1.0	1.0006358
$g(1)$	-0.9	-0.89831411
$g(2)$	0.8	0.79984317
$g(3)$	-0.7	-0.70179211
$g(4)$	0.6	0.60227514

$g(4)$ を求めることができていることが確認できた．今回の用いたプログラムを以下に示す．

```
import numpy as np

def gaussSeidel( A, b, tol ):
  xOld = np.empty_like(b)
  error = 1e12

  L = np.tril(A)
  U = A - L
  LInv = np.linalg.inv(L)

  while error > tol:
    x = np.dot( LInv, b-np.dot( U, xOld ) )
    error = np.linalg.norm( x - xOld )/np.linalg.norm(x)
    xOld = x
  return x

N = 1000
m = 5
g = [1.0,-0.9,0.8,-0.7,0.6]
uniform_rand = np.zeros((2,N))
gauss_rand = np.zeros((2,N))
y = np.zeros(N)

for i in range(N):
  uniform_rand[0][i] = np.random.rand()
  uniform_rand[1][i] = np.random.rand()
  gauss_rand[0][i] = np.sqrt(-2*np.log(uniform_rand[0][i]))*np.cos(2*np
  gauss_rand[1][i] = np.sqrt(-2*np.log(uniform_rand[0][i]))*np.sin(2*np
  for j in range(m):
    y[i] += g[j]*gauss_rand[0][i-j]
  y[i] = y[i] + ((gauss_rand[1][i])/20)

x = np.zeros(m)
x = np.append(x,gauss_rand[0])
cov = np.zeros((m,m))
xy_cov = np.zeros(m,)

for i in range(m):
  for j in range(m):
    cov[i][j] = cov[j][i] = np.cov(x[m-i:N+m-i],x[m-j:N+m-j],bias=True)
    cov[i][i] = np.cov(x[m-i:N+m-i],x[m-i:N+m-i],bias=True)[0,1]

for i in range(5):
  xy_cov[i] = np.cov(x[m-i:N+m-i],y[0:1000],bias=True)[0,1]

x = gaussSeidel(cov,xy_cov,1e-20)

print(x)
```