

84.

*****問題84*****

```

import numpy as np

# (m, k) の定義
def m(x):
    return 0

def k(x, y):
    return np.exp(-(x-y)**2 / 2)

# 関数 gp_sample の定義
def gp_sample(x, m, k):
    n = len(x)
    m_x = m(x)
    k_xx = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            k_xx[i, j] = k(x[i], x[j])
    R = np.linalg.cholesky(k_xx) # lower triangular matrix
    u = np.random.randn(n)
    return R.dot(u) + m_x

if __name__ == "__main__":
    from pathlib import Path

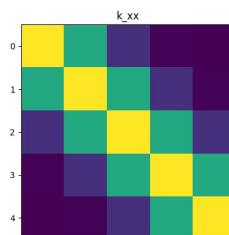
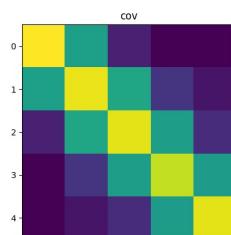
    import matplotlib.pyplot as plt
    # 亂数を発生して、共分散行列を生成し、k_xx と比較
    x = np.arange(-2, 3, 1)
    n = len(x)
    r = 1000
    z = np.zeros((n, r))
    for i in range(r):
        z[:, i] = gp_sample(x, m, k)
    k_xx = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            k_xx[i, j] = k(x[i], x[j])

    print("cov(z):\n", np.cov(z), "\n")
    print("k_xx:\n", k_xx)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(np.cov(z))
    plt.title("cov")
    plt.subplot(1, 2, 2)
    plt.imshow(k_xx)
    plt.title("k_xx")

    filename = Path("src/exercise/Chap6/out/problem84/result.png")
    filename.parent.mkdir(parents=True, exist_ok=True)
    plt.savefig(filename)

```



85. Y, f_2 の同時分布は、

$$\begin{bmatrix} Y \\ f_2 \end{bmatrix} \sim N \left(\begin{bmatrix} m_x \\ m_z \end{bmatrix}, \begin{bmatrix} R_{xx} + \sigma^2 I & R_{xz} \\ R_{zx} & R_{zz} \end{bmatrix} \right) \quad \text{である。}$$

部題 61 から、 Y の t と f_2 の事後確率は、

$$m' = m_z + R_{zx}(R_{xx} + \sigma^2 I)^{-1}(Y - m_x)$$

$$\mathcal{L}' = R_{zz} - R_{zx}(R_{xx} + \sigma^2 I)^{-1}R_{xz} \quad \text{且し } N(\mu', \mathcal{L}') \text{ である。}$$

すなはち、 $n=1, z_i = x$ を用いて、上記の式は f_2 と $f(x)$ の分布は、

$$m'(x) := m(x) + R_{xx}(R_{xx} + \sigma^2 I)^{-1}(Y - m_x) \quad (6.3)$$

$$f'(x, x) := f(x, x) - R_{xx}(R_{xx} + \sigma^2 I)^{-1}R_{xx} \quad (6.4)$$

となる。

(3)

86

```

1 def gp_2(x_pred):
2     h = np.zeros(n)
3     for i in range(n):
4         h[i] = k(x_pred, x[i])
5     L = np.linalg.cholesky(K + sigma_2 * np.identity(n))
6     alpha = np.linalg.solve(L, np.linalg.solve(L.T, (y - mu(x)))) → O(N³)
7     mm = mu(x_pred) + np.sum(np.dot(h.T, alpha))
8     gamma = np.linalg.solve(L.T, h) → O(N²)
9     ss = k(x_pred, x_pred) - np.sum(gamma**2)
10    return {"mm": mm, "ss": ss}

```

$O(N^3)$ の計算が必要で、これはコレスキーフィルフの他の実装
 では、 $O(N^2)$ の計算が必要な部分は、水色で省記入。

$$87. P(Y=1|x) = \frac{1}{1+\exp(-f(x))},$$

$$P(Y=-1|x) = \frac{\exp(-f(x))}{1+\exp(-f(x))} = \frac{1}{1+\exp(f(x))}$$

である。 $y \in \{-1, +1\}$ とす。

$$P(Y=y|x) = \left\{ 1 + \exp(-y f(x)) \right\}^{-1} \text{である。}$$

52. N 個の (x_i, y_i) のペアが与えられており、尤度関数を

$$\prod_{i=1}^N P(Y=y_i|x_i) = \prod_{i=1}^N \left\{ 1 + \exp(-y_i f(x_i)) \right\}^{-1}$$

とする。この最大化はこの対数

$$\log \prod_{i=1}^N \left\{ 1 + \exp(-y_i f(x_i)) \right\}^{-1} = - \sum_{i=1}^N \log [1 + \exp(-y_i f(x_i))]$$

の最大化と等価である。

また、 2^N は $\sum_{i=1}^N \log [1 + \exp(-y_i f(x_i))]$ の最小化と等価である。

88. 28行目

$$I + W^{\frac{1}{2}} R_{xx} W^{\frac{1}{2}} = LL^T \quad E満たす L を求めよ$$

29行目

$$\gamma := Wf_x + u$$

30行目

$$L\beta = W^{\frac{1}{2}} R_{xx} \gamma \quad E満たす \beta を求めよ$$

31行目

$$L^T W^{-\frac{1}{2}} \alpha = \beta \quad E満たす \alpha を求めよ$$

32行目

$$f_x := R_{xx}(\gamma - \alpha)$$

E満たす

22行目

$$\begin{aligned} f_x &:= (W + R_{xx}^{-1})^{-1} (Wf_x + u) \quad \stackrel{\text{Woodbury}}{\sim} \\ &= [R_{xx} - R_{xx} W^{\frac{1}{2}} (I + W^{\frac{1}{2}} R_{xx} W^{\frac{1}{2}})^{-1} W^{\frac{1}{2}} R_{xx}] \gamma \\ &= R_{xx} \gamma - R_{xx} W^{\frac{1}{2}} (LL^T)^{-1} L \beta \quad \because LB = W^{\frac{1}{2}} R_{xx} \gamma \\ &= R_{xx} (\gamma - W^{\frac{1}{2}} (LL^T)^{-1} L L^T W^{\frac{1}{2}} \alpha) \quad \because \alpha \in PのPDAK \\ &= R_{xx} (\gamma - \alpha) \end{aligned}$$

である(?) 28行目と 32行目は、

$$f_x \leftarrow (W + R_{xx}^{-1})^{-1} (Wf_x + u) \quad \rightarrow \text{更新した} Z_{c1} \text{を}$$

89.

```
*****問題89*****
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris

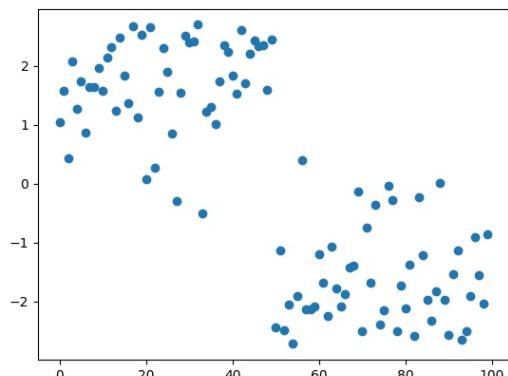
df = load_iris()           # Iris データ
x = df.data[50:150, 0:4]
y = np.array([1]*50 + [-1]*50)
n = len(y)

# 4 個の共変量でカーネルを計算
def k(x, y):
    return np.exp(np.sum(-(x - y)**2) / 2)

K = np.zeros((n, n))
for i in range(n):
    for j in range(n):
        K[i, j] = k(x[i, :], x[j, :])
eps = 0.00001
f = [0] * n
g = [0.1] * n

while np.sum((np.array(f) - np.array(g))**2) > eps:
    i = i + 1
    g = f           # 比較のため、更新前の値を保存する
    f = np.array(f)
    y = np.array(y)
    v = np.exp(-y * f)
    u = y * v / (1 + v)
    w = (v / (1 + v)**2)
    W = np.diag(w)
    W_p = np.diag(w**0.5)
    W_m = np.diag(w**(-0.5))
    L = np.linalg.cholesky(np.identity(n) + np.dot(np.dot(W_p, K), W_p))
    gamma = W.dot(f) + u
    beta = np.linalg.solve(L, np.dot(np.dot(W_p, K), gamma))
    alpha = np.linalg.solve(np.dot(L.T, W_m), beta)
    f = np.dot(K, (gamma - alpha))
print(list(f))

plt.scatter(range(len(f)), list(f))
filename = Path("src/exercise/Chap6/out/problem89/result.png")
filename.parent.mkdir(parents=True, exist_ok=True)
plt.savefig(filename)
```



うまく正負が分かれています

90. $f_2 \in M_{f_2|Y}$ に置き換えて $M(x)$ は LZ の理由.

$Y \rightarrow f_2 \rightarrow f(x)$ の順で Markov 遠似が成り立つ。

$$E[f_2|Y] = M_{f_2|Y} \quad \text{である。}$$

独立性

$f(x)$ の分布は f_2 は条件付で $f_2(z|x)$ で決まる。

f_2 の分布は Y

\rightarrow Z , Y が与えられ $f_2(z|Y)$, f_2 は Z の変動と $f(x)$ は子変動は独立である。

91. gp-1 における R の計算 ($K^+ \alpha^2 I$ の並行部分の計算) を
説明せよ。

92. 命題 66 を 証明 すれば 完了。

$f(w, x) - m(x)$ の 平均 分 散 は それ ぞれ 0, $\beta(x, x) = \beta$,

$f(w, x) - m(x) \in f(w, y) - m(y)$ の 支 分 散 は $\beta(x, y)$ である。

m, β が 連続 $\Rightarrow \lim_{n \rightarrow \infty} \mathbb{E}[|f(w, x_n) - f(w, x)|^2] = 0^{(6.30)}$ で ある。

$$\mathbb{E}[|f(w, x) - f(w, y)|^2]$$

$$= \mathbb{E}\left[\left((f(w, x) - m(x)) - (f(w, y) - m(y)) + (m(x) - m(y))\right)^2\right]$$

$$= \mathbb{E}[(f(w, x) - m(x))^2] + \mathbb{E}[(f(w, y) - m(y))^2] + \mathbb{E}[(m(x) - m(y))^2]$$

$$- 2 \mathbb{E}[(f(w, x) - m(x))(f(w, y) - m(y))]$$

$$- 2 \mathbb{E}[(f(w, y) - m(y))(m(x) - m(y))]$$

$$- 2 \mathbb{E}[(m(x) - m(y))(f(w, x) - m(x))]$$

$$= \beta(x, x) + \beta(y, y) - 2\beta(x, y) + (m(x) - m(y))^2$$

であるで、 m と β が 連続 で ある なら す、 (6.30) の 成り立つ。

(6.30) \Rightarrow m, k が連続 E が成り立つ。

一般に既定 $k(x, y) = f(x)$, $m \equiv 0$ のとき、

$$k(x, y) - k(x', y') = (k(x, y) - f(x, y)) + (f(x, y) - f(x', y'))$$

の各括弧内は \mathbb{E} 。

$$\begin{aligned} |k(x, y) - k(x', y')| &= |\mathbb{E}[f(w, x)f(w, y)] - \mathbb{E}[f(w, x)f(w, y')]| \\ &\leq \mathbb{E}[f(w, y)^2]^{\frac{1}{2}} \mathbb{E}[|f(w, x) - f(w, x')|^2]^{\frac{1}{2}} \\ &= \sqrt{k(y, y)} \sqrt{\mathbb{E}[|f(w, x) - f(w, x')|^2]} \end{aligned}$$

同様に、

$$|k(x', y) - k(x, y)| \leq \sqrt{k(x', x')} \sqrt{\mathbb{E}[|f(w, y) - f(w, y')|^2]}$$

と上から k が連続の式、(6.30) が成り立つ。

□

93. Karhunen-Loeve の定理の証明

命題67の(6.32)式).

$$\begin{aligned} \mathbb{E}[f_n(w, x)^2] &= \mathbb{E}\left[\left(\sum_{j=1}^n I_f(w, e_j) e_j(x)\right)^2\right] \\ &= \sum_{i=1}^n \sum_{j=1}^n \mathbb{E}[I_f(w, e_i) I_f(w, e_j)] e_i(x) e_j(x) \\ &= \sum_{j=1}^n \lambda_j e_j^2(x) \end{aligned}$$

が成り立つ。よって (6.31) と 命題67の(2) 式,

$$\begin{aligned} \mathbb{E}[f_n(w, x) f(w, x)] &= \mathbb{E}\left[\sum_{j=1}^n I_f(w, e_j) e_j(x) f(w, x)\right] \\ &= \sum_{j=1}^n e_j(x) \int_E R(x, y) e_j(y) d\mu(y) \\ &= \sum_{j=1}^n \lambda_j e_j^2(x) \int_E e_j^2(y) d\mu(y) \\ &= \sum_{j=1}^n \lambda_j e_j^2(x) \end{aligned}$$

が成り立つ。

$$\begin{aligned} \mathbb{E}[|f_n(w, x) - f(w, x)|^2] &= \mathbb{E}[f_n(w, x)^2] - 2\mathbb{E}[f_n(w, x) f(w, x)] + \mathbb{E}[f(w, x)^2] \\ &= \sum_{j=1}^n \lambda_j e_j^2(x) - 2 \sum_{j=1}^n \lambda_j e_j^2(x) + R(x, x) \\ &= R(x, x) - \sum_{j=1}^n \lambda_j e_j^2(x) \end{aligned}$$

である。

$$\lim_{n \rightarrow \infty} \sup_{x \in E} \mathbb{E}[|f(w, x) - f_n(w, x)|] = 0$$

③

Brown運動の図示

```
"""問題93"""
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np

def lam(j):          # 固有値
    return 4 / ((2 * j - 1) * np.pi)**2

def ee(j, x):        # 固有関数の定義
    return np.sqrt(2) * np.sin((2 * j - 1) * np.pi / 2 * x)

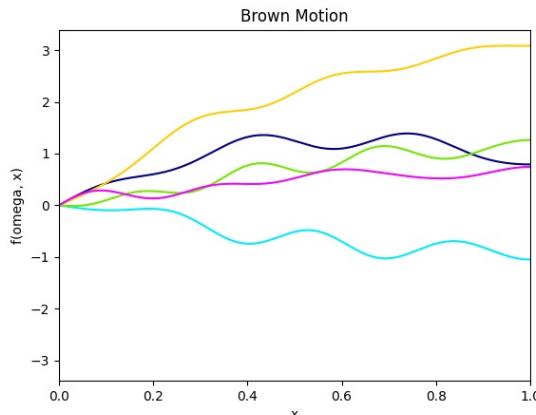
n = 10
m = 5

# Gauss 過程の定義
def f(z, x):
    n = len(z)
    S = 0
    for i in range(n):
        S = S + z[i] * ee(i, x) * np.sqrt(lam(i))
    return S

plt.figure()
plt.xlim(0, 1)
plt.xlabel("x")
plt.ylabel("f(omega, x)")
colormap = plt.cm.gist_ncar # nipy_spectral, Set1, Paired
colors = [colormap(i) for i in np.linspace(0, 0.8, m)]

for j in range(m):
    z = np.random.randn(n)
    x_seq = np.arange(0, 3.001, 0.001)
    y_seq = []
    for x in x_seq:
        y_seq.append(f(z, x))
    plt.plot(x_seq, y_seq, c=colors[j])

plt.title("Brown Motion")
filename = Path("src/exercise/Chap6/out/problem93/result.png")
filename.parent.mkdir(parents=True, exist_ok=True)
plt.savefig(filename)
```



94. $\varphi_{5/2}$ ($P=2$)

$$\begin{aligned}\varphi_{5/2}(z) &= \exp\left(-\frac{\sqrt{5}z}{\ell}\right) \frac{\Gamma(3)}{\Gamma(5)} \sum_{i=0}^2 \frac{(2+i)!}{i!(2-i)!} \left(\frac{\sqrt{20}z}{\ell}\right)^{2-i} \\ &= \exp\left(-\frac{\sqrt{5}z}{\ell}\right) \frac{2!}{4!} \left(\left(\frac{\sqrt{20}z}{\ell}\right)^2 + 3! \frac{\sqrt{20}z}{\ell} + \frac{4!}{2!}\right) \\ &= \left(1 + \frac{\sqrt{5}z}{\ell} + \frac{5z^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}z}{\ell}\right)\end{aligned}$$

$\varphi_{3/2}$ ($P=1$)

$$\begin{aligned}\varphi_{3/2}(z) &= \exp\left(-\frac{\sqrt{3}z}{\ell}\right) \frac{\Gamma(2)}{\Gamma(3)} \sum_{i=0}^1 \frac{(1+i)!}{i!(1-i)!} \left(\frac{\sqrt{12}z}{\ell}\right)^{1-i} \\ &= \exp\left(-\frac{\sqrt{3}z}{\ell}\right) \frac{1}{2} \left(\frac{\sqrt{12}z}{\ell} + 2\right) \\ &= \left(1 + \frac{\sqrt{3}z}{\ell}\right) \exp\left(-\frac{\sqrt{3}z}{\ell}\right)\end{aligned}$$

$\ell = 0.05$ のときの Matérn カーネル ($\nu = 1, \dots, 10$)

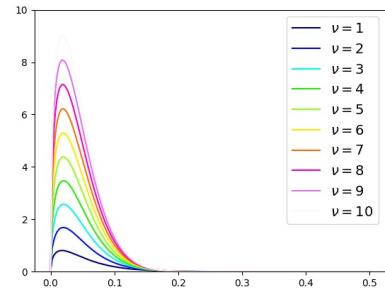
```
"""問題94"""
from pathlib import Path

import numpy as np
from matplotlib import pyplot as plt
from scipy.special import gamma

def matern(nu, l, r):
    p = nu - 1 / 2
    S = 0
    for i in range(int(p+1)):
        S += gamma(p + i + 1) / gamma(i + 1) / gamma(p - i + 1) \
            * (np.sqrt(8 * nu) * r / l)**(p - i)
    S *= gamma(p + 2) / gamma(2 * p + 1) * np.exp(-np.sqrt(2 * nu) * r / l)
    return S

if __name__ == "__main__":
    m = 10
    l = 0.05
    colormap = plt.cm.gist_ncar # nipy_spectral, Set1, Paired
    color = [colormap(i) for i in np.linspace(0, 1, len(range(m)))]
    x = np.linspace(0, 0.5, 200)
    plt.ylim(0, 10)
    for i in range(1, m + 1):
        plt.plot(x, matern(i, l, x), c=color[i - 1], label=r"\nu=%d" % i)

plt.legend(loc="upper right", frameon=True, prop={"size": 14})
filename = Path("src/exercise/Chap6/out/problem94/result.png")
filename.parent.mkdir(parents=True, exist_ok=True)
plt.savefig(filename)
```



95.

*****問題95*****

```

from pathlib import Path

import numpy as np
from matplotlib import pyplot as plt
from scipy.special import gamma

colormap = plt.cm.gist_ncar # nipy_spectral, Set1, Paired
colors = [colormap(i) for i in np.linspace(0, 0.8, 5)]

def matern(nu, l, r):
    p = nu - 1 / 2
    S = 0
    for i in range(int(p+1)):
        S = S + gamma(p + i + 1) / gamma(i + 1) / gamma(p - i + 1) \
            * (np.sqrt(8 * nu) * r / l)**(p - i)
    S = S * gamma(p + 2) / gamma(2 * p + 1) * np.exp(-np.sqrt(2 * nu) * r / l)
    return S

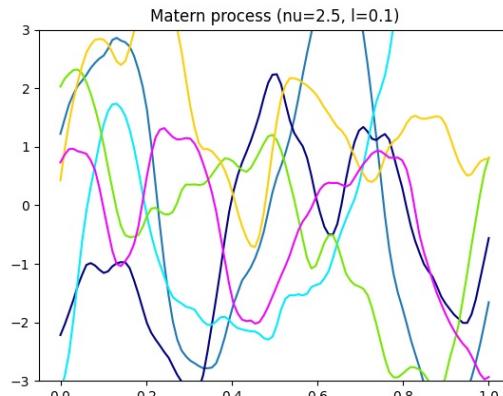
def rand_100(Sigma):
    L = np.linalg.cholesky(Sigma) # 共分散行列を Cholesky 分解
    u = np.random.randn(100)
    y = L.dot(u) # 平均 0 の共分散行列 Sigma の乱数を 1 組発生
    return y

x = np.linspace(0, 1, 100)
z = np.abs(np.subtract.outer(x, x)) # 距離行列を計算
l = 0.1
nu = 5 / 2

plt.figure()
Sigma_M = matern(nu, l, z) # 行列
y = rand_100(Sigma_M)
plt.plot(x, y)
plt.ylim(-3, 3)
for i in range(5):
    y = rand_100(Sigma_M)
    plt.plot(x, y, c=colors[i])
plt.title(f"Matern process ({nu=}, {l=})")

filename = Path("src/exercise/Chap6/out/problem95/result.png")
filename.parent.mkdir(parents=True, exist_ok=True)
plt.savefig(filename)

```



96. ランダム要素であるが、確率過程ではない例

時間の構造が存在しない確率変数、たとえば、サンドイッチを1回振って得られる目は、ランダム要素であるが、確率変数ではない。

確率過程であるが、ランダム要素ではない例

不明

97

```
*****問題97*****
import numpy as np
import skfda
X, y = skfda.datasets.fetch_weather(return_X_y=True, as_frame=True)
df = X.iloc[:, 0].values

def g(j, x):
    # 基底を p 個用意する
    if j == 0:
        return 1 / np.sqrt(2 * np.pi)
    if j % 1 == 0:
        return np.cos((j // 2) * x) / np.sqrt(np.pi)
    else:
        return np.sin((j // 2) * x) / np.sqrt(np.pi)

def eta(x):
    return np.array([g(i, x) for i in range(p)])

def beta(x, y):
    # 関数の p 個の基底の係数を計算
    X = np.zeros((N, p))
    for i in range(N):
        for j in range(p):
            X[i, j] = g(j, x[i])
    beta = np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)),
                         + 0.0001 * np.identity(p)), X.T), y)
    return np.squeeze(beta)

def calc_K_N(x_array, C, d):
    """(6.40)の計算"""
    K_N = np.zeros((p, p))
    for i in range(p):
        x = x_array[i]
        eta_x = eta(x)
        for j in range(p):
            y = x_array[j]
            eta_y = eta(y)
            K_N[i, j] = 1 / N * eta_x.T @ (C - d).T @ (C - d) @ eta_y
    return K_N

N = 365
n = 35
p = 100
df = df_coordinates[0].data_matrix
C = np.zeros((n, p))
x = np.arange(1, N+1) * (2 * np.pi / N) - np.pi
for i in range(n):
    y = df[i]
    C[i, :] = beta(x, y)

d = C / N
K_N = calc_K_N(x, C, d)

print(f'{C.shape=}')
print("C=n", C)

print(f'{d.shape=}')
print("d=n", d)

print(f'{K_N.shape=}')
print("K_N=n", K_N)

"""
C.shape=(35, 100)
C=
[[ 3.91857883e+00  5.54170723e+00  7.51345140e+00 ... -2.50833038e-02
   9.09669951e-03  9.09669949e-03]
 [ 1.3847856e+00  7.26689475e+00  9.78004069e+00 ...  4.22972179e-02
  -6.87705525e-02 -6.87705525e-02]
 [ 4.60349458e+00  6.51032447e+00  8.94517048e+00 ...  4.16597541e-02
  -4.58403972e-02 -4.58403972e-02]

[-8.6515770e+00 -1.14958554e+01  1.37821772e+01 ...  2.89353305e-02
  -9.08034699e-02 -9.08034700e-02]
[-7.72041067e+00 -1.09183094e+01  1.86918496e+01 ...  5.59748757e-02
  6.36540111e-02  6.36540110e-02]
[-1.38017850e+01 -1.95186715e+01  1.52525593e+01 ... -3.26473727e-02
  -1.47387978e-02 -1.47387978e-02]
d.shape=(35, 100)
d=
[[ 1.07358324e-02  1.51827598e-02  2.05847983e-02 ... -6.87213802e-05
   2.49060261e-05  2.49060260e-05]
 [ 4.0780015e-02  1.99993007e-02  2.67946320e-02 ...  1.15882789e-04
  -1.88412473e-04 -1.88412473e-04]
 [-2.115348139e-02  1.78365054e-02  2.45073164e-02 ...  1.14136312e-04
  -1.25590129e-04 -1.25590129e-04]
 ...
 [-2.20963225e-02 -3.12489189e-02  3.77593896e-02 ...  7.92748781e-05
  -2.48776630e-04 -2.48776630e-04]
 [-2.11518100e-02 -2.99131766e-02  5.12105468e-02 ...  1.53355824e-04
  -1.74394551e-04 -1.74394551e-04]
 [-3.78131095e-05 -5.34758122e-02  4.17878337e-02 ... -8.94448566e-05
  -4.03802679e-05 -4.03802680e-05]
K_N.shape=(100, 100)
K_N=
[[ 23.25480918 23.38947594 23.55997927 ... -0.35058581 -0.63956022
   -0.96739058]
 [23.38947594 23.52866019 23.70421461 ... -0.3476852 -0.63948188
   -0.97051358]
 [23.55997927 23.70421461 23.8856201 ... -0.32817788 -0.62354911
   -0.9586173 ]
 ...
 [-0.35058581 -0.3476852 -0.32817788 ...  4.93659181  4.93224433
  -4.94683031]
 [-0.62056032 -0.63948188 -0.62354911 ...  4.93224433  4.93313199
  -4.94683031]
 [-0.96739058 -0.97051358 -0.9586173 ...  4.94028509  4.94683031
  -4.96720162]]
"""

```

$$98. \quad n_i(x) = \begin{cases} \frac{1}{\sqrt{2\pi}} & i=1 \\ \frac{\cos kx}{\sqrt{\pi}} & i=2k-1 \\ \frac{\sin kx}{\sqrt{\pi}} & i=2k \end{cases} \quad (k \in \mathbb{N})$$

と表せる：つまり、問題23 のように書くのが簡単だ。

$$w_{ij} := \int_{-\pi}^{\pi} n_i(x) n_j(x) dx$$

$$= \delta_{i,j}$$

$$\text{つまり } W = [w_{ij}] = I_p \quad i \text{ある}$$

99.

```
*****問題99*****
import numpy as np
import skfda
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA

X, y = skfda.datasets.fetch_weather(return_X_y=True, as_frame=True)
df = X.iloc[:, 0].values
```

```
def g(j, x): # 基底を p 個用意する
    if j == 0:
        return 1 / np.sqrt(2 * np.pi)
    if j % 1 == 0:
        return np.cos((j // 2) * x) / np.sqrt(np.pi)
    else:
        return np.sin((j // 2) * x) / np.sqrt(np.pi)

def beta(x, y): # 関数の p 個の基底の係数を計算
    X = np.zeros((N, p))
    for i in range(N):
        for j in range(p):
            X[i, j] = g(j, x[i])
    beta = np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)) + 0.0001 * np.identity(p)), X.T), y)
    return np.squeeze(beta)
```

```
N = 365
n = 35
m = 5
p = 100

# df.coordinates[0]: 気温, df.coordinates[1]: 降水量:
df = df.coordinates[1].data_matrix
C = np.zeros((n, p))
for i in range(n):
    x = np.arange(1, N+1) * (2 * np.pi / N) - np.pi
    y = df[i]
    C[i, :] = beta(x, y)
pca = PCA()
pca.fit(C)
B = pca.components_.T
xx = C.dot(B)

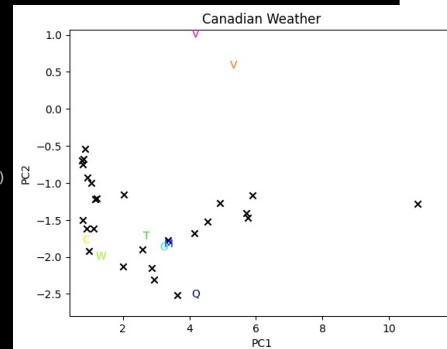
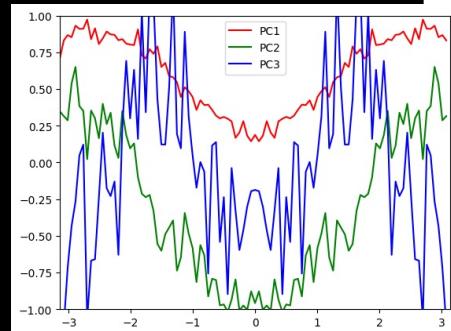
x_seq = np.arange(-np.pi, np.pi, 2 * np.pi / 100)
colors = ["red", "green", "blue"]
colormap = plt.cm.gist_ncar # nipy_spectral, Set1, Paired
color = [colormap(i) for i in np.linspace(0, 1, len(range(10)))]

def h(coef, x): # 係数を用いて関数を定義
    S = 0
    for j in range(p):
        S = S + coef[j] * g(j, x)
    return S
```

```
plt.figure()
plt.xlim(-np.pi, np.pi)
plt.ylim(-1, 1)
for j in range(3):
    plt.plot(x_seq, h(B[:, j], x_seq), c=colors[j], label="PC%d" % (j+1))
plt.legend(loc="best")
plt.show()
```

```
place = X.iloc[:, 1]
index = [9, 11, 12, 13, 16, 23, 25, 26]
others = [x for x in range(34) if x not in index]
first = [place[i][0] for i in index]
```

```
plt.figure()
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.xlim(min(xx[:, 0]) * 1.1, max(xx[:, 0]) * 1.1)
plt.ylim(min(xx[:, 1]) * 1.1, max(xx[:, 1]) * 1.1)
plt.title("Canadian Weather")
plt.scatter(xx[others, 0], xx[others, 1], marker="x", c="k")
for i in range(8):
    l = plt.text(xx[index[i], 0], xx[index[i], 1], s=first[i], c=color[i])
plt.show()
```



(100) (図6.8と図6.9)と(問題99)の結果を見れば分かる。

(問題の黄図は合ってます?)

気温と降水量を同時に扱うなら...



x_1 を気温、 x_2 を降水量とする基底関数を

$$k_1 = 0, \dots, k_2 = 0, \dots, n/2.$$

$\frac{1}{2\pi}$	$(k_1 = k_2 = 0)$
$\frac{1}{\sqrt{2}\pi} \cos(k_2 x_2)$	$(k_1 = 0, k_2 : \text{even})$
$\frac{1}{\sqrt{2}\pi} \sin(k_2 x_2)$	$(k_1 = 0, k_2 : \text{odd})$
$\frac{1}{\sqrt{2}\pi} \cos(k_1 x_1)$	$(k_1 : \text{even}, k_2 = 0)$
$\frac{1}{\sqrt{2}\pi} \sin(k_1 x_1)$	$(k_1 : \text{odd}, k_2 = 0)$
$\frac{1}{\pi} \cos(k_1 x_1) \cos(k_2 x_2)$	$(k_1 : \text{even}, k_2 : \text{even})$
$\frac{1}{\pi} \cos(k_1 x_1) \sin(k_2 x_2)$	$(k_1 : \text{even}, k_2 : \text{odd})$
$\frac{1}{\pi} \sin(k_1 x_1) \cos(k_2 x_2)$	$(k_1 : \text{odd}, k_2 : \text{even})$
$\frac{1}{\pi} \sin(k_1 x_1) \sin(k_2 x_2)$	$(k_1 : \text{odd}, k_2 : \text{odd})$

と定義する: ここで、気温と降水量を同時に扱う。