

46. $\alpha_i \in \mathbb{R}$, $\forall i = 1, \dots, N$ の最適化が $f \in H$ の最適化と等価である。なぜならば, RKHS である H は,
 $M := \text{span}(\{K(x_i, \cdot)\}_{i=1}^N)$ とその直交補空間

$$M^\perp = \{f \in H \mid \langle f, K(x_i, \cdot) \rangle_H = 0, i = 1, \dots, N\}$$

の和を書け, $f_1 \in M, f_2 \in M^\perp$ とし, $f = f_1 + f_2$ であれば,

$$\begin{aligned} \sum_{i=1}^N (y_i - f(x_i))^2 &= \sum_{i=1}^N (y_i - f_1(x_i))^2 \quad (\because f_2 \in M^\perp) \\ &= \sum_{i=1}^N \left(y_i - \sum_{j=1}^N \alpha_j K(x_j, x_i) \right)^2 \quad (\because f_1 \in M), \end{aligned} \quad \cdots (1)$$

となる。よしに, $i = 1, \dots, N$ は勾配,

$$\begin{aligned} f(x_i) &= \langle f_1(\cdot) + f_2(\cdot), K(x_i, \cdot) \rangle_H \\ &= \langle f_1(\cdot), K(x_i, \cdot) \rangle_H \quad (\because f_2 \in M^\perp) \\ &= f_1(x_i) \end{aligned} \quad \cdots (2)$$

が成り立つ。よしに,

$$\begin{aligned} \|f\|_H^2 &= \|f_1\|_H^2 + \|f_2\|_H^2 + 2\langle f_1, f_2 \rangle_H \\ &= \|f_1\|_H^2 + \|f_2\|_H^2 \quad (\because f_1 \in M, f_2 \in M^\perp) \\ &\geq \|f_1\|_H^2 \end{aligned}$$

が成り立つ。

$$\begin{aligned} &\sum_{i=1}^N |y_i - f(x_i)|^2 + \lambda \|f\|^2 \\ &\geq \sum_{i=1}^N \left(y_i - \sum_{j=1}^N \alpha_j K(x_j, x_i) \right)^2 + \lambda \|f\|^2 \quad (\because (1), (2)), \\ &= \sum_{i=1}^N \left(y_i - \sum_{j=1}^N \alpha_j K(x_j, x_i) \right)^2 + \lambda \left\| \sum_{j=1}^N \alpha_j K(x_j, \cdot) \right\|^2 \end{aligned}$$

が成り立つ。よしである。

$$\text{SVM}, \alpha = [\alpha_1, \dots, \alpha_N]^T, K \in \mathbb{R}^{N \times N} \text{ が SVM 行列}, y = [y_1, \dots, y_N]^T \in \mathbb{R}^N,$$

$$\|f_\alpha\|_H^2 = \left\langle \sum_{i=1}^N \alpha_i k(x_i, \cdot), \sum_{j=1}^N \alpha_j k(x_j, \cdot) \right\rangle_H$$

$$= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \langle k(x_i, \cdot), k(x_j, \cdot) \rangle_H = \alpha^T K \alpha$$

とみなすと、目的関数の最小化問題.

$$\|y - K\alpha\|^2 + \lambda \alpha^T K \alpha \rightarrow \text{最小化問題}.$$

$\Rightarrow K \in \mathbb{R}^{n \times n}$ が対称正定, α を求める.

$$-K(y - K\alpha) + \lambda K\alpha = 0$$

つまり, $K \geq 0$ ならば, $K + \lambda I \geq 0$ す.

$\hat{\alpha} = (K + \lambda I)^{-1} y$ が最適解となり, 対応する誤差値は,

$$\|y - K(K + \lambda I)^{-1} y\|^2 + \lambda \|y^T (K + \lambda I)^{-1} K (K + \lambda I)^{-1} y\|.$$

47. $\alpha \in \mathbb{R}^N$ の最適化で, $f \in H$ の最適化が達成されるのは, (46)と同じ理由(表現定理)である。つまり,

$$f_1 \in M := \text{span}\left(\{\kappa(x_i, \cdot)\}_{i=1}^N\right), f_2 \in M^\perp \Leftrightarrow \forall f = f_1 + f_2 \in H, f^T \alpha = f_1^T \alpha$$

$$\sum_{i=1}^N f(x_i)^2 = \sum_{i=1}^N \langle f_1(\cdot) + f_2(\cdot), \kappa(x_i, \cdot) \rangle_H^2$$

$$= \sum_{i=1}^N \langle f_1(\cdot), \kappa(x_i, \cdot) \rangle_H^2 \quad (\because f_2 \in M^\perp)$$

$$= \sum_{i=1}^N f_1(x_i)^2$$

$$= \sum_{i=1}^N \left| \sum_{j=1}^N \alpha_j \kappa(x_j, x_i) \right|^2 \quad \alpha^T K^T K \alpha \quad (\because K = P)$$

$$= \sum_{i=1}^N \sum_{r=1}^N \sum_{s=1}^N \alpha_r \alpha_s \kappa(x_r, x_i) \kappa(x_s, x_i) = \alpha^T K^2 \alpha, \quad \text{⑨}$$

$$\|f\|_H^2 = \|f_1 + f_2\|_H^2 = \|f_1\|_H^2 + \|f_2\|_H^2 \quad (\because f_1 \in M, f_2 \in M^\perp),$$

$$\geq \|f_1\|_H^2 = \left\| \sum_{j=1}^N \alpha_j \kappa(x_j, \cdot) \right\|_H^2$$

$$= \sum_{r=1}^N \sum_{s=1}^N \alpha_r \alpha_s \kappa(x_r, x_s)$$

$$= \alpha^T K \alpha$$

①, ② すなはち (4.8) の最大化因。

$\alpha^T K^T K \alpha - \mu(\alpha^T K \alpha - 1)$ の最大化に帰着できる。

もし R , $\beta = K^{\frac{1}{2}} \alpha$ とするとき, 固有方程式 $K\beta = \lambda \beta$ の固有値・固有ベクトルを

$\lambda_1, \dots, \lambda_N, u_1, \dots, u_N$ とする。

$$\alpha = K^{-\frac{1}{2}} \beta = \frac{1}{\sqrt{\lambda}} \beta = \frac{u_1}{\sqrt{\lambda_1}}, \dots, \frac{u_N}{\sqrt{\lambda_N}}$$

となる。

問題の意図が読み取れる。

```

****問題48****
from typing import Callable
import numpy as np

def kernel_pca_train(X: np.ndarray, k: Callable[[np.ndarray, np.ndarray], float]) -> np.ndarray:
    """(4.9)のように中心化されたGram行列に対して最適なalphaを求める

    Args:
        X (np.ndarray): データ(N, p)
        k (Callable[[np.ndarray, np.ndarray], float]): カーネル

    Returns:
        np.ndarray: (4.9)のように中心化されたGram行列に対して最適なalpha(N, N)
    """
    # データXとカーネルkからGram行列を求める
    N = X.shape[0]
    K = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            K[i, j] = k(X[i], X[j])
    # (4.9)に従って中心化
    S = [np.sum(K[i]) for i in range(N)]
    T = [np.sum(K[:, j]) for j in range(N)]
    U = np.sum(K)
    for i in range(N):
        for j in range(N):
            K[i, j] = K[i, j] - S[i] / N - T[j] / N + U / N**2
    # 因縁と回帰ベクトルを用いて最適なalphaを求める
    val, vec = np.linalg.eigh(K)
    idx = val.argsort()[-1:-1]
    val = val[idx]
    vec = vec[idx]
    alpha = np.zeros_like(K)
    for i in range(N):
        alpha[:, i] = vec[:, i] / np.sqrt(np.maximum(val[i], 1e-6))
    return alpha

def kernel_pca_test(
    X: np.ndarray,
    k: Callable[[np.ndarray, np.ndarray], float],
    alpha: np.ndarray,
    m: int,
    z: np.ndarray,
) -> np.ndarray:
    """X, k, alpha(kernel_pca_train(X, k)), m, zからm次までのスコアpcaを求める

    Args:
        X (np.ndarray): データ(N, p)
        k (Callable[[np.ndarray, np.ndarray], float]): カーネル
        alpha (np.ndarray): kernel_pca_train(X, k)
        m (int): 1<= m <= p
        z (np.ndarray): Xのある行x_i (i in [N])(p)

    Returns:
        np.ndarray: X, k, alpha(kernel_pca_train(X, k)), m, zからm次までのスコアpca
    """
    N = X.shape[0]
    pca = np.zeros(m)
    for i in range(N):
        pca = pca + alpha[i, 0:m] * k(X[i], z)
    return pca

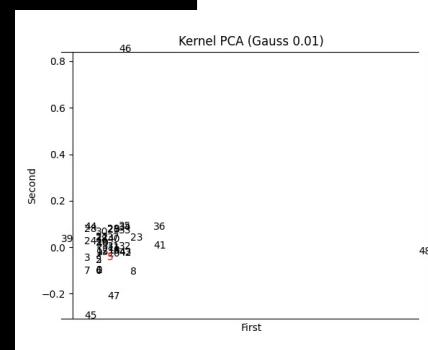
if __name__ == "__main__":
    """動作確認"""
    from pathlib import Path

    import matplotlib.pyplot as plt
    import pandas as pd

    sigma2 = 1e-2
    def k(x: np.ndarray, y: np.ndarray) -> float:
        """Gaussian kernel."""
        return np.exp(-np.linalg.norm(x - y)**2 / 2 / sigma2)
    X = pd.read_csv("https://raw.githubusercontent.com/selva86/datasets/master/USArrests.csv")
    x = X.to_numpy()[:, :-1]
    n = x.shape[0]
    p = x.shape[1]
    alpha = kernel_pca_train(x, k)
    z = np.zeros((n, 2))
    for i in range(n):
        z[i, :] = kernel_pca_test(x, k, alpha, 2, x[i, :])

    min1 = np.min(z[:, 0])
    min2 = np.min(z[:, 1])
    max1 = np.max(z[:, 0])
    max2 = np.max(z[:, 1])
    plt.xlim(min1, max1)
    plt.ylim(min2, max2)
    plt.xlabel("First")
    plt.ylabel("Second")
    plt.title("Kernel PCA (Gauss 0.01)")
    for i in range(n):
        if i != 4: # no ga: PLR2004
            plt.text(x=z[i, 0], y=z[i, 1], s=i) # type: ignore[arg-type]
    plt.text(z[4, 0], z[4, 1], 5, c="r") # type: ignore[arg-type]
    filename = Path("src/exercise/Chap4/out/problem48/result.png")
    filename.parent.mkdir(parents=True, exist_ok=True)
    plt.savefig(filename)
    plt.close()

```



49.

すすめのため、 $X \in \mathbb{R}^{N \times P}$ は正規化されてゐる。

- カ-ズレを用ひる場合、

$$X = \bigcup \Sigma V^T \quad (\Sigma \in \mathbb{R}^{N \times P}, \Sigma \in \mathbb{R}^{P \times P}, V \in \mathbb{R}^{P \times P}) \text{ と } SVD_{\text{SVD}},$$

Σ, V は直交行列, Σ は対角行列である。

$$\frac{1}{N-1} X^T X = \frac{1}{N-1} V \Sigma^2 V^T \text{ となる}, \text{ から } V \Sigma \text{ が } 3D,$$

$$\frac{1}{N-1} X^T X V = \frac{1}{N-1} V \Sigma^2 V^T V \text{ となる}, V \text{ の各列が主成分ベクトル},$$

主成分

$\Rightarrow \exists I, X_1, \dots, X_N \in \mathbb{R}^P$ のとき

$$X V = \bigcup \Sigma V^T \cdot V = \bigcup \Sigma \text{ の最初の } m \text{ 列である}.$$

- 線形カ-ズレの場合

Gram 行列は $K = X X^T = \bigcup \Sigma^2 U^T$ と書け、右側の Σ が $3D$ 。

$K U = X X^T U = \bigcup \Sigma^2$ となる。すなはち、 U の各列が P_1, \dots, P_N であり、 $K^{-\frac{1}{2}} U$ の各列

$\alpha_1, \dots, \alpha_N$ が主成分ベクトルとなる。 $\exists I, X_1, \dots, X_N \in \mathbb{R}^P$ のとき

$$K K^{-\frac{1}{2}} U = \bigcup \Sigma^2 U^T (\bigcup \Sigma^2 U^T)^{-\frac{1}{2}} U = \bigcup \Sigma \text{ の最初の } m \text{ 列である}.$$

したがって、中心化は Σ と一致する。修正された Gram 行列の (i, j) 成分は、

$$x_i x_j - \frac{1}{N} \sum_{k=1}^N x_i x_k - \frac{1}{N} \sum_{k=1}^N x_j x_k + \frac{1}{N} \sum_{k=1}^N \sum_{l=1}^N x_k x_l = (x_i - \bar{x})(x_j - \bar{x})$$

と一致する。カ-ズレを用ひる場合の中心化と一致する。 \Rightarrow 得られた U は同じである。

$$50. f(x_i) = f_i(x_i) \quad (i=1, \dots, N) \text{ と, } \|f\|_H \geq \|f_i\|_H, f(\cdot) = \sum_{i=1}^N \gamma_i k(x_i, \cdot)$$

から、(4.12) の最小化問題は次の問題と等価である：

$$\begin{aligned} & \text{minimize}_{\beta} \frac{1}{2} \left\| \sum_{i=1}^N \gamma_i k(x_i, \cdot) \right\|_H^2 + C \sum_{i=1}^N \varepsilon_i \\ & \gamma \in \mathbb{R}^N, \beta \in \mathbb{R}, \varepsilon \in \mathbb{R}^N \\ & \text{subject to} \quad \gamma_i (f(x_i) + \beta_0) - (1 - \varepsilon_i) \geq 0, \quad \forall i \\ & \quad \varepsilon_i \geq 0, \quad \forall i \end{aligned}$$

命題43の対応

$f_0(\beta)$
($\beta = (\gamma, \beta_0, \varepsilon)$)
 $f_1(\beta) \geq 0$,
 $f_2(\beta) \geq 0$, $\forall i$

この問題のKKT条件は、 f_0, f_1, f_2 が凸であることに注意して。

" $(\gamma, \beta, \varepsilon)$ が 上記の問題の 最適解であること、

$$\gamma_i (f(x_i) + \beta_0) - (1 - \varepsilon_i) \geq 0 \quad \forall i, \quad f_1(\beta^*) \geq 0$$

$$\varepsilon_i \geq 0 \quad \forall i, \quad f_2(\beta^*) \geq 0$$

である、

$$d_i \nabla \gamma_i (f(x_i) + \beta_0) - (1 - \varepsilon_i) = 0 \quad \forall i, \quad d_i f_1(\beta^*) = 0$$

$$M_i \varepsilon_i = 0 \quad \forall i, \quad M_i f_2(\beta^*) = 0.$$

$$\sum_j \gamma_j k(x_i, x_j) - \sum_j d_j \gamma_j k(x_i, x_j) = 0 \quad \forall i, \quad \nabla_{\gamma_i} L = 0$$

$$\sum_i d_i \gamma_i = 0 \quad \forall i, \quad \nabla_{\beta_0} L = 0,$$

$$C - d_i - M_i = 0 \quad \forall i, \quad \nabla_{\varepsilon_i} L = 0.$$

$$d_i \geq 0 \quad \forall i, \quad \text{うがうねり系数}$$

$$M_i \geq 0 \quad \forall i$$

を満たすうがうねり系数 d_i, M_i が存在することは同値である。

この3つの式から, $M_i \geq 0 \Leftrightarrow C - d_i \geq 0 \Leftrightarrow 0 \leq d_i \leq C$ となり, 故特徴の式が一層す。

51.

```
*****問題51*****
from typing import Callable

import cvxopt
import matplotlib.pyplot as plt
import numpy as np
from cvxopt import matrix

def K_linear(x: np.ndarray, y: np.ndarray) -> float:
    """線形カーネル"""
    return x.T @ y

def K_poly(x: np.ndarray, y: np.ndarray) -> float:
    """多项式カーネル"""
    return (1 + x.T @ y)**2

def create_K_Gaussian(sigma2: float) -> Callable[[np.ndarray, np.ndarray], float]:
    """ガウジアンカーネルのグラム行列を返す関数を作成"""
    def K_Gaussian(x: np.ndarray, y: np.ndarray) -> float:
        return np.exp(-np.linalg.norm(x - y)**2 / 2 / sigma2)
    return K_Gaussian

def svm_2(
    X: np.ndarray,
    y: np.ndarray,
    C: float,
    K: Callable[[np.ndarray, np.ndarray], float],
    )-> dict[str, np.ndarray]:
    """SVMの最適解を出力"""
    eps = 0.0001
    n = X.shape[0]
    P = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            P[i, j] = K(X[i, :], X[j, :]) * y[i] * y[j]
    # パッケージにある matrix 関数を使って指定する必要がある
    P = matrix(P + np.eye(n) * eps)
    A = matrix(-y.T.astype(float))
    b = matrix(np.array([0]).astype(float))
    h = matrix(np.array([C] * n + [0] * n).reshape(-1, 1).astype(float))
    G = matrix(np.concatenate([np.diag(np.ones(n)), np.diag(-np.ones(n))]))
    q = matrix(np.array([-1] * n).astype(float))
    res = cvxopt.solvers.qp(P, q, A=A, b=b, G=G, h=h)
    alpha = np.array(res["x"])
    # x が本文中の alpha に対応
    beta = ((alpha * y).T @ X).reshape(2, 1)
    index = (eps < alpha[:, 0]) & (alpha[:, 0] < C - eps)
    beta_0 = np.mean(y[index] - X[index, :] @ beta)
    return {"alpha": alpha, "beta": beta, "beta_0": beta_0}

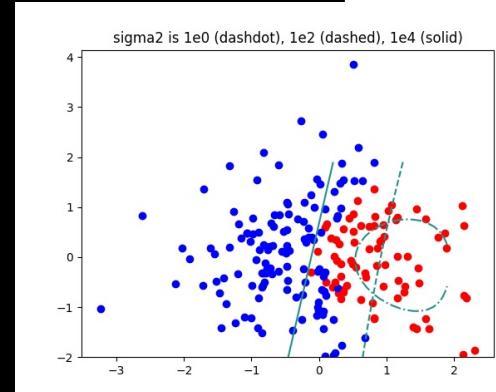
def plot_kernel(X: np.ndarray, y: np.ndarray, K: Callable[[np.ndarray, np.ndarray], float], line: str) -> None:
    """境界線を図示"""
    res = svm_2(X, y, 1, K)
    alpha = res["alpha"][:, 0]
    beta_0 = res["beta_0"]

    def f(u: float, v: float) -> float:
        """Function f"""
        S = beta_0
        for i in range(X.shape[0]):
            S = S + alpha[i] * y[i] * K(X[i, :], np.array([u, v]))
        return S

    # ww は f(x,y) の高さ。これから輪郭を求めることができる
    uu = np.arange(-2, 2, 0.1)
    vv = np.arange(-2, 2, 0.1)
    ww = []
    for v in vv:
        w = [f(u, v) for u in uu]
        ww.append(w)
    plt.contour(uu, vv, ww, levels=0, linestyles=line)

if __name__ == "__main__":
    """動作確認"""
    from pathlib import Path
    rs = np.random.RandomState(42) # 亂数生成器を固定
    a = 3
    b = -1
    n = 200
    X = rs.randn(n, 2)
    y = np.sign(a * X[:, 0] + b * X[:, 1]**2 + 0.3 * rs.randn(n))
    y = y.reshape(-1, 1)
    for i in range(n):
        if y[i] == 1:
            plt.scatter(X[i, 0], X[i, 1], c="red")
        else:
            plt.scatter(X[i, 0], X[i, 1], c="blue")
    plot_kernel(X, y, create_K_Gaussian(sigma2=1e0), line="dashdot")
    plot_kernel(X, y, create_K_Gaussian(sigma2=1e2), line="dashed")
    plot_kernel(X, y, create_K_Gaussian(sigma2=1e4), line="solid")
    plt.title("sigma2 is 1e0 (dashdot), 1e2 (dashed), 1e4 (solid)")

filename = Path("src/exercise/Chap4/out/problem51/result.png")
filename.parent.mkdir(parents=True, exist_ok=True)
plt.savefig(filename)
plt.close()
```



52. (4, 21) 8).

$$g''(\xi_1) = 6\beta_4 = 0 \quad \text{or} \quad g''(\xi_1) = 2\beta_3 + 6\beta_4\xi_1 = 0$$
$$\Leftrightarrow \beta_3 = \beta_4 = 0.$$

(4, 22) 8).

$$g''(\xi_J) = 6\beta_4 + 6 \sum_{j=1}^J \beta_{j+4} = 0 \quad \text{or} \quad g''(\xi_J) = 2\beta_3 + 6\beta_4\xi_J + 6 \sum_{j=1}^J \beta_{j+4}(\xi_J - \xi_j) = 0$$

由 $\beta_3 = \beta_4 = 0$ 可得 $\beta_3 = \beta_4 = 0$.

$$\sum_{j=1}^J \beta_{j+4} = 0. \quad \text{或} \quad \sum_{j=1}^J \beta_{j+4}(\xi_J - \xi_j) = 0$$

$$\Leftrightarrow \xi_J \sum_{j=1}^J \beta_{j+4} = \sum_{j=1}^J \beta_{j+4} \xi_j$$

$$\Leftrightarrow \sum_{j=1}^J \beta_{j+4} \xi_j = 0.$$

成真數。

53.

(a). r は最高次数が $2g-1$ の自然なアプローチである。

$$r^{(q)}(0) = \dots = r^{(2g-1)}(0) = r^{(q)}(1) = \dots = r^{(2g-1)}(1) = 0$$

である。

$$\begin{aligned} \int_0^1 r^{(q)}(x) S^{(q)}(x) dx &= [r^{(q)}(x) S^{(q-1)}(x)]_0^1 - \int_0^1 r^{(q+1)}(x) S^{(q-1)}(x) dx \\ &= - \int_0^1 r^{(q+1)}(x) S^{(q-1)}(x) dx \quad (\because r^{(q)}(1) = r^{(q)}(0) = 0) \\ &= \dots = (-1)^{q-1} \int_0^1 r^{(2g-1)}(x) S^{(1)}(x) dx \\ &= (-1)^{q-1} \sum_{j=1}^{N-1} r^{(2g-1)}(x_j^+) [S(x_{j+1}) - S(x_j)] = 0. \end{aligned}$$

である。□

($\because S(x_i) = 0, i=1, \dots, N, r^{(2g-1)}(x_j^+) \in r$ の $(2g-1)$ 次の微分係数)

(b). $r^{(2g-1)}(x_j^+)$ は区間 (x_j, x_{j+1}) で一定である。

$$\begin{aligned} \int_0^1 |g^{(q)}(x)|^2 dx &= \int_0^1 |r^{(q)}(x) + S^{(q)}(x)|^2 dx \quad 0 \quad (\because (a)) \\ &= \int_0^1 |r^{(q)}(x)|^2 dx + \int_0^1 |S^{(q)}(x)|^2 dx + 2 \underbrace{\int_0^1 r^{(q)}(x) S^{(q)}(x) dx}_{\text{0}} \\ &= \int_0^1 |r^{(q)}(x)|^2 dx + \int_0^1 |S^{(q)}(x)|^2 dx \\ &\geq \int_0^1 |r^{(q)}(x)|^2 dx \end{aligned}$$

である。□

(c). 一方, $g, r \in W_g[0, 1]$ である, $s \in W_g[0, 1]$ である。

$$S(x) = \sum_{i=0}^{g-1} \frac{S^{(i)}(0)}{i!} x^i + \int_0^1 \frac{(x-u)^{g-1}}{(g-1)!} S^{(g)}(u) du$$

が成立する。(7. および (b). の導き成り立つこと, ここで $S^{(g)}(x) = 0$ である) これは,

$$S(x) = \sum_{i=0}^{g-1} \frac{S^{(i)}(0)}{i!} x^i$$

を意味する。□

(d). (c) の結果から, $S(x_i) = 0$ ($i=1, \dots, N$) を満たす必要があるが,

N が g の次数 $g-1$ を上回れば, 全区間で $S(x) = 0$ である。□

54. $\hat{k}(x, y)$ の平均が $k(x, y)$ である ($\hat{k}(x, y)$ が $k(x, y)$ の不偏推定量である)ことを示す。

$$\begin{aligned} \mathbb{E}[\hat{k}(x, y)] &= \mathbb{E}\left[\frac{1}{m} \sum_{i=1}^m z_i(x) z_i(y)\right] \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{E}[z_i(x) z_i(y)] \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{E}[\cos(w_i^\top x + b_i) \cos(w_i^\top y + b_i)] \\ &= 2 \times \frac{1}{m} \sum_{i=1}^m \mathbb{E}[\cos(w_i^\top x + b_i) \cos(w_i^\top y + b_i)] \\ &= \frac{1}{m} \times m k(x, y) \quad (\because k(x, y) の定義) \\ &= k(x, y). \end{aligned}$$

□

*****問題54*****

```
import numpy as np

m = 100
sigma = 10
sigma2 = sigma ** 2
rs = np.random.RandomState(42)

def k(x: np.ndarray, y: np.ndarray) -> np.ndarray:
    return np.exp(-(x - y) ** 2 / (2 * sigma2))

def z(x: np.ndarray, w: np.ndarray, b: np.ndarray) -> np.ndarray:
    """Function z."""
    return np.sqrt(2 / m) * np.cos(w * x + b)

def zz(x: np.ndarray, y: np.ndarray, w: np.ndarray, b: np.ndarray) -> float:
    """Function zz."""
    return np.sum(z(x, w, b) * z(y, w, b))

def estimate_hat_k(x: np.ndarray, y: np.ndarray) -> float:
    """hat_kを推定"""
    w = rs.randn(m) / sigma
    b = rs.rand(m) * 2 * np.pi
    return zz(x, y, w, b)

ITER = 1000
gap = np.zeros(ITER)
for i in range(ITER):
    x = rs.randn(1)
    y = rs.randn(1)
    gap_i = np.abs(estimate_hat_k(x, y) - k(x, y))
    gap[i] = gap_i

print("Mean of gap (absolute error) is ", np.mean(gap))
# Mean of gap (absolute error) is 0.056773402236984225
```

よく似定でている。

55. まず、補題 6 を示す。

X が非負の値を取る確率密度である。このとき、 $E[X]$ が

$$E[X] = E[X \cdot I(X \geq \varepsilon)] + E[X \cdot I(X < \varepsilon)]$$

$$\geq E[X \cdot I(X \geq \varepsilon)] \quad \text{区间分割}$$

$$\geq \varepsilon P[X \geq \varepsilon]$$

$$\therefore P[X \geq \varepsilon] \leq \frac{E[X]}{\varepsilon} \quad \text{が成り立つ。}$$

次に補題 6 を用いて、補題 5 を示す。

補題 6 は、任意の $s > 0$ で

lemma.6 ($\exp(sX), \exp(s\varepsilon) \geq 0$)
↓

$$P[X \geq \varepsilon] = P[sX \geq s\varepsilon] = P[\exp(sX) \geq \exp(s\varepsilon)] \leq e^{-s\varepsilon} E[e^{sX}]$$

が成り立つ。また、任意の $\varepsilon > 0$ に

$$P[X \geq \varepsilon] \leq \inf_{s>0} e^{-s\varepsilon} E[e^{sX}]$$

が成り立つ。

□

56. 命題46 から, (4.29) を導く.

命題46 (Hoeffdingの不等式) 独立に生起し, $[a_i, b_i]$ の範囲に値をとる X_i ($i = 1, \dots, n$)
および任意の $\epsilon > 0$ について

$$P(|\bar{X} - \mathbb{E}[\bar{X}]| \geq \epsilon) \leq 2 \exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \quad (4.30)$$

が成立する。ただし, \bar{X} で $(X_1 + \dots + X_n)/n$ をあらわすものとする。

$\hat{\beta}(x, y)$ は $\beta(x, y)$ の正偏確定量である, すなはち, $\mathbb{E}[\hat{\beta}(x, y)] = \beta(x, y)$ である。

また, $\hat{\beta}(x, y) := \frac{1}{m} \sum_{i=1}^m Z_i(x) Z_i(y)$,

$$Z_i(x) = \sqrt{2} \cos(w_i^\top x + b_i)$$

から, $-2 \leq Z_i(x) Z_i(y) \leq 2$ である。

$$\begin{cases} X : \beta(x, y) \\ n : m \\ a_i : -2 \\ b_i : 2 \end{cases} \quad \text{と対応付けるとして。}$$

$$\begin{aligned} P(|\beta(x, y) - \hat{\beta}(x, y)| \geq \epsilon) &\leq 2 \exp\left(-\frac{2m^2\epsilon^2}{16m}\right) \\ &= 2 \exp\left(-\frac{m\epsilon^2}{8}\right) \end{aligned}$$

となる。 \square

命題 45 の証明

加法定理より、

$$2 \cos(\omega^\top x + b) \cos(\omega^\top y + b) = \cos(w^\top(x - y)) + \cos(w^\top(x + y) + 2b)$$

が成立する。第 2 項は ω を固定して b で平均をとると 0 になるので、

$$\mathbb{E}_{\omega,b}[\sqrt{2} \cos(\omega^\top x + b) \cdot \sqrt{2} \cos(\omega^\top y + b)] = \mathbb{E}_\omega \cos(w^\top(x - y))$$

が成立する。また、Euler の公式 $e^{i\theta} = \cos \theta + i \sin \theta$ を命題 5 に適用すると $k(x, y)$ が実数値をとるので、 $\mathbb{E}[\sin(\omega^\top(x - y))] = 0$ となり、 $k(x, y)$ は

$$\mathbb{E}_\omega \exp(iw^\top(x - y)) = \mathbb{E}_\omega [\cos(\omega^\top(x - y)) + i \sin(\omega^\top(x - y))] = \mathbb{E}_\omega [\cos(\omega^\top(x - y))]$$

と書ける。これと命題 5 より、(4.28) が成立する。 \square

ここで、命題 5 を用いてます。

命題 5 により、これらの性質が導かれています。

58. Woodbury の公式（後に証明する）から、

$$(Z^T Z + \lambda I_m) Z^T = Z^T (Z Z^T + \lambda I_N)$$

$$\Leftrightarrow Z^T (Z Z^T + \lambda I_N)^{-1} = (Z^T Z + \lambda I_m)^{-1} Z^T$$

($\because Z Z^T + \lambda I_N > 0, Z^T Z + \lambda I_m > 0$ の両辺)

が成り立つ。RFF における指定した式に用いた X_{train}, X_N は

別の $x \in E$ に対して、 $Z(x) = [Z_1(x), \dots, Z_m(x)]$ (行ベクトル)

とする。 $\hat{\beta} := (Z^T Z + \lambda I_m)^{-1} Z^T y$ とする。

$$\hat{f}(x) := \sum_{i=1}^N \hat{\alpha}_i \hat{r}(x, x_i) = \sum_{i=1}^N \hat{\alpha}_i Z(x) Z(x_i)^T$$

$$= Z(x) \sum_{i=1}^N Z(x_i)^T \hat{\alpha}_i = Z(x) \underbrace{Z^T \hat{\alpha}}_{\sim}$$

$$= Z(x) \underbrace{Z^T (Z Z^T + \lambda I_N)^{-1} y}_{\text{R}} \quad (Z Z^T + \lambda I_N) \hat{\alpha} = y \text{ が満たす。}$$

$$= Z(x) (Z^T Z + \lambda I_m)^{-1} Z^T y$$

$$= Z(x) \hat{\beta}$$

が成り立つ。

2. 2. Woodbury の公式:

- $\text{既約 } U \in \mathbb{R}^{r \times s}, V \in \mathbb{R}^{s \times r}, r, s \geq 1$ の時.

$$U(I_s + VU) = (I_r + UV)U \quad \text{が成り立つ。}$$

$$(I_r + UV) = U + UVU \quad (\because UI_s = U)$$

$$(I_r + UV) = U + UVU \quad (\because I_r U = U)$$

\Rightarrow 2式を比較して、

59. (4.33) を計算する

$$Z \in \mathbb{R}^{N \times m} \text{ の計算} = O(mN)$$

$$Z^T Z \quad \vdots \quad : O(m^2 N)$$

$$(Z^T Z + \lambda I_m)^{-1} \quad : O(m^3)$$

$$\underbrace{(Z^T Z + \lambda I_m)^{-1} Z^T y}_{2つめの積} : O(m^2)$$

(4.33) を得るまでに

高々 $O(m^2 N)$

の計算コスト

• 続いて $X \in E$ は? 1. $Z, f(x)$ を計算する

$$Z(x) \text{ の計算} = O(m)$$

$$Z(x) \hat{\beta} \quad \vdots \quad : O(m)$$

高々 $O(m)$

の計算コスト

6D. (4.34) の証明

Sherman-Morrison-Woodbury の公式 (4.35) は

$$r = m, s = N, A = \lambda I_N, U = R, C = I_r, V = R^T E \text{ とおぼて,}$$

$$(\lambda I_N + R I_r R^T)^{-1} = (\lambda I_N)^{-1} - (\lambda I_N)^{-1} R (I_m^{-1} + R^T (\lambda I_N)^{-1} R)^{-1} R^T (\lambda I_N)^{-1}$$

$$\Leftrightarrow (R R^T + \lambda I_N)^{-1} = \frac{1}{\lambda} \left\{ I_N - R \left(I_m + \frac{1}{\lambda} R^T R \right)^{-1} R^T \frac{1}{\lambda} \right\}$$

$$= \frac{1}{\lambda} \left\{ I_N - R (R^T R + \lambda I_m)^{-1} R^T \right\}$$

□

$K = R R^T$ の分解が でて左とす。左と左の計算量は
 $N \times N$ の正方行列の逆行列を求めるのに必要な $O(N^3)$ が,
右の計算量は $m \times m$ の正方行列の逆行列を求めるのに
必要な $O(m^3)$ と $N \times m$ と $m \times m$ の行列の積に必要な
 $O(Nm^2)$ であり, 総じて $O(Nm^2)$ である。

よって, 右の計算がよっぽど重い。

6 |

```
1 def alpha_m(K, x, y, m):
2     n = len(x)
3     U, D, V = np.linalg.svd(K[:, :m], :m])
4     u = np.zeros((n, m))
5     for i in range(m):
6         for j in range(n):
7             u[j, i] = np.sqrt(m/n) * np.sum(K[j, :m] * U[:, m, i] / D[i])
8     mu = D * n / m
9     R = np.zeros((n, m))
10    for i in range(m):
11        R[:, i] = np.sqrt(mu[i]) * u[:, i]
12    Z = np.linalg.inv(np.dot(R.T, R) + lam * np.eye(m))
13    alpha = np.dot((np.eye(n) - np.dot(np.dot(R, Z), R.T)), y) / lam
14    return(alpha)
```

この部分以下のよう修正する。

$\alpha = \text{np.linalg.inv}(\text{np.dot}(R, R.T) + \text{lam} * \text{np.eye}(n))$

62. $R_{k,k}^2$ は任意の k について非負であるので、

$$f(i) = B_{k,k} - \sum_{k=1}^{i-1} R_{k,k}^2 \quad \text{は単調減少である。}$$

一方、最も大きい i^* に対して、 R_k の選び方に $\underbrace{\dots}_{\text{成り立つ。}}$ $f(i^*) \geq 0$ が

$R_{k,k}^2$ が最大となる index であり、
 $R_{k,k}^2 \geq 0$.

63. リスト内法で示す

まず、 $i = 1$ のとき、

初期
 $j=1 : R_{1,1} = \sqrt{B_{1,1}} \quad \& \quad B_{1,1} = \sum_{h=1}^i R_{j,h} R_{i,h} E_{j,h}$

初期角
 $j=2, \dots, N$ のとき、 $R_{j,1} = \frac{1}{\sqrt{B_{1,1}}} B_{j,1} \quad (\because E_{j,1} > 0)$

$$\sum_{h=1}^i R_{j,h} R_{i,h} = R_{j,1} R_{1,1} = \frac{1}{\sqrt{B_{1,1}}} B_{j,1} \sqrt{B_{1,1}} = B_{j,1}$$

$$\therefore B_{j,i} = \sum_{h=1}^i R_{j,h} R_{i,h} \quad \text{を満たす。}$$

次に第*i*列が実現して $i+3$ を仮定するすれば第*i+1*列

初期
 $B_{j,i} = \sum_{h=1}^i R_{j,h} R_{i,h} \quad (i=i, j=i+1, \dots, N)$

が成り立つことを仮定する。このとき第*i+1*列を仮定する

まず、初期要素 $R_{i+1,i+1} \neq 0$?

$$R_{i+1,i+1} = \sqrt{B_{i+1,i+1} - \sum_{h=1}^i R_{i+1,h}^2}$$

$$\Leftrightarrow B_{i+1,i+1} = R_{i+1,i+1}^2 + \sum_{h=1}^i R_{i+1,h}^2$$

$$= \sum_{h=1}^{i+1} R_{i+1,h} R_{i+1,h} \quad \text{「なぜか?」等式成立。}$$

次に、非初期要素 $R_{j,i+1} \quad (j = i+2, \dots, N)$ が $\neq 0$?

$$R_{j,i+1} = \frac{1}{R_{i+1,i+1}} \left(B_{j,i+1} - \sum_{h=1}^i R_{j,h} R_{i+1,h} \right) \quad (\because E_{j,i+1} > 0)$$

$$\Leftrightarrow B_{j,i+1} = R_{j,i+1} R_{i+1,i+1} + \sum_{h=1}^i R_{j,h} R_{i+1,h}$$
$$= \sum_{h=1}^{i+1} R_{j,h} R_{i+1,h} \quad \text{「なぜか?」等式成立。}$$

したがって数学的帰納法で一度実現した列が再度更新されることはあり得ない。

帰納法の用い方として、
完成された列が更新されて、
それが体系的に言っている。
不要でも(元の問題のため)

64.

```

import numpy as np

def incomplete_cholesky(A, m):
    A_copy = A.copy() # Aのコピーを作成
    n = A_copy.shape[0]
    R = np.zeros((n, n))
    P = np.eye(n)
    for i in range(m):
        max_R = -np.inf
        for j in range(i, n):
            RR = A_copy[j, j]
            for h in range(i):
                RR = RR - R[j, h]**2
            if max_R < RR:
                k = j
                max_R = RR
        R[i, i] = np.sqrt(max_R)
        if k != i:
            for j in range(i):
                w = R[i, j]
                R[i, j] = R[k, j]
                R[k, j] = w
            for j in range(n):
                w = A_copy[j, k]
                A_copy[j, k] = A_copy[j, i]
                A_copy[j, i] = w
            for j in range(n):
                w = A_copy[k, j]
                A_copy[k, j] = A_copy[i, j]
                A_copy[i, j] = w
        Q = np.eye(n)
        Q[i, i] = 0
        Q[k, K] = 0
        Q[i, K] = 1
        Q[k, i] = 1
        P = np.dot(P, Q)
        if i < n - 1:
            for j in range(i + 1, n):
                S = A_copy[j, i]
                for h in range(i):
                    S = S - R[i, h] * R[j, h]
                R[j, i] = S / R[i, i]
    return np.dot(P, R)

if __name__ == "__main__":
    n = 5
    r = 3
    D = np.random.randint(-n, n, size=(n, n))
    A = np.dot(D, D.T) + n * np.eye(n) # A is symmetric positive definite
    L = incomplete_cholesky(A, r)
    print("A\n", A)
    print("L @ L.T\n", np.dot(L, L.T))

```

元々の incomplete_cholesky は一概に発生してしまったが、自己修正によって、r=5で問題なく A@L@L.T が得られた。