1. (証明)

- (a)⇒(b):

$A = B^\top B$ を満たす $B \in \mathbb{R}^{n\times n}$ が存在すると仮定する。

このとき、任意の $x \in \mathbb{R}^n$ に対して、

$$x^\top A x \underset{A=B^\top B\,(\text{仮定})}{=} x^\top (B^\top B) x = (Bx)^\top (Bx) = \|Bx\|^2 \geq 0$$

- (b)⇒(c):

任意の $x \in \mathbb{R}^n$ に対して、$x^\top A x \geq 0$ が成立すると仮定する。

このとき、Aの固有値 $\lambda \in \mathbb{R}$ と対応する固有ベクトル $v \in \mathbb{R}^n$

($Av = \lambda v$ が成り立つ) に対して、

$$0 \underset{\text{仮定}}{\leq} v^\top A v \underset{Av=\lambda v}{=} v^\top \lambda v \underset{\lambda:\text{スカラー}}{=} \lambda v^\top v = \lambda \|v\|^2$$

が成り立つ。$\|v\|^2 \geq 0$ であるので、これは $\lambda \geq 0$ を意味する。

- (c)⇒(a):

A は実対称行列より、対角化可能であり、

$A = PDP^\top$ ($P \in \mathbb{R}^{n\times n}, D = \mathrm{diag}([\lambda_1, \dots, \lambda_n])$) と表せる。

今、Aのすべての固有値 $\lambda_1, \dots, \lambda_n \geq 0$ と仮定すると、

$\sqrt{D} = \mathrm{diag}([\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}])$ を定義できる。このとき、

$$A \underset{\text{対角化}}{=} P^\top D P \underset{\sqrt{D}\sqrt{D}=D,\ \sqrt{D}^\top=\sqrt{D}}{=} (P^\top \sqrt{D}^\top)(\sqrt{D}P) = (\sqrt{D}P)^\top (\sqrt{D}P)$$ が成り立つ。

よって、$A = B^\top B$ なる行列 $B = \sqrt{D}P \in \mathbb{R}^{n\times n}$ が存在する。

以上より、

$$\begin{array}{ccc} (a) & \Rightarrow & (b) \\ & \Uparrow \quad \Downarrow & \\ & (c) & \end{array}$$

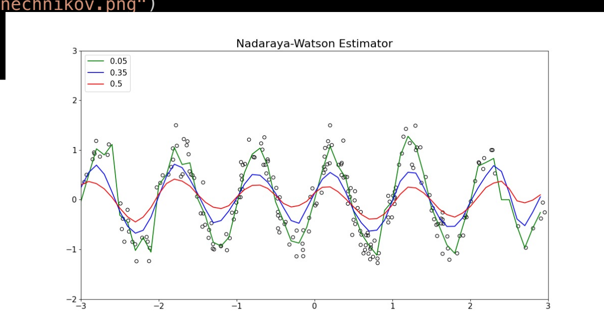
であるので、(a),(b),(c) はすべて同値。∎

(プログラム)

```python
"""問題1の後半（プログラムによる確認）"""
import numpy as np

rs = np.random.RandomState(42)
n = 4
B = rs.rand(n, n)
print(B)
"""
[[0.37454012 0.95071431 0.73199394 0.59865848]
 [0.15601864 0.15599452 0.05808361 0.86617615]
 [0.60111501 0.70807258 0.02058449 0.96990985]
 [0.83244264 0.21233911 0.18182497 0.18340451]]
"""


A = np.dot(B.T, B)
print(A)
"""
[[1.21892212 0.98281189 0.44695573 1.09506235]
 [0.98281189 1.47464666 0.75816171 1.42998244]
 [0.44695573 0.75816171 0.57267288 0.54183765]
 [1.09506235 1.42998244 0.54183765 2.08301543]]
"""



for _ in range(5):
    x = rs.rand(n)
    xTAx = np.dot(x, np.dot(A, x))
    is_nonnegative = (xTAx >= 0)
    print(f"{xTAx=}, {is_nonnegative=}")
    """
    xTAx=2.3448553824012173, is_nonnegative=True
    xTAx=1.9559046849520527, is_nonnegative=True
    xTAx=4.538994617211748, is_nonnegative=True
    xTAx=1.4777079672301598, is_nonnegative=True
    xTAx=7.949924623318303, is_nonnegative=True
    """
```

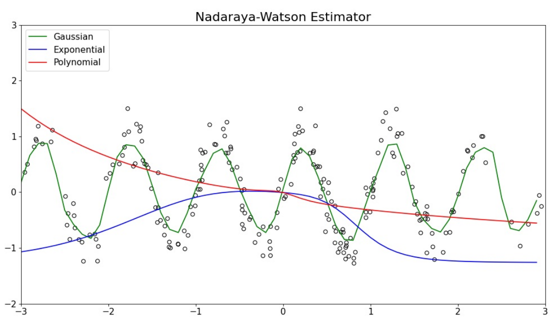
2. 前半：Dとfの定義と動作の確認

```python
"""問題2の前半（D, fの動作の確認(Epanechnikov)）"""
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np


def D(t: float) -> float:
    """Function D for Epanechnikov kernel."""
    if np.abs(t) <= 1.:
        return 3 * (1 - t ** 2) / 4
    return 0.

def k(x: float, y: float, lam: float) -> float:
    """Epanechnikov kernel."""
    return D(np.abs((x - y) / lam))


def f(observed_data: list[tuple[float, float]], x: float, lam: float) -> float:
    """Nadaraya-Watson Estimator."""
    numerator = np.sum([k(x, x_i, lam) * y_i for x_i, y_i in observed_data])
    denominator = np.sum([k(x, x_i, lam) for x_i, _ in observed_data])
    if denominator == 0:
        return 0.
    return numerator / denominator


rs = np.random.RandomState(42)
n = 250
x = 2 * rs.normal(size=n)
y = np.sin(2 * np.pi * x) + rs.normal(size=n) / 4
observed_data = [(x[i], y[i]) for i in range(n)]

plt.figure(num=1, figsize=(15, 8), dpi=80)
plt.xlim(-3, 3)
plt.ylim(-2, 3)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.scatter(x, y, facecolors="none", edgecolors="k", marker="o")

xx = np.arange(-3, 3, 0.1)
yy: list = [[] for _ in range(3)]
lam = [0.05, 0.35, 0.50]
color = ["g", "b", "r"]
for i in range(3):
    for zz in xx:
        yy[i].append(f(observed_data, zz, lam[i]))
    plt.plot(xx, yy[i], c=color[i], label=lam[i])

plt.legend(loc="upper left", frameon=True, prop={"size": 14})
plt.title("Nadaraya-Watson Estimator", fontsize=20)

# Save.
out_dir = Path("src/exercise/Chap1/out/problem2")
out_dir.mkdir(parents=True, exist_ok=True)
filename = Path(out_dir, "Epanechnikov.png")
plt.savefig(filename)
```



2. 後半: Gauss, 指数型, 多項式 カーネルの実装と確認

```python
"""問題2の後半（Epanechnikovカーネル以外）"""
from pathlib import Path
from typing import Callable

import matplotlib.pyplot as plt
import numpy as np


def f(observed_data: list[tuple[float, float]], x: float, k: Callable[[float, float],
float]) -> float:
    """Nadaraya-Watson Estimator."""
    numerator = np.sum([k(x, x_i) * y_i for x_i, y_i in observed_data])
    denominator = np.sum([k(x, x_i) for x_i, _ in observed_data])
    if denominator == 0:
        return 0.
    return numerator / denominator

methods = ["Gaussian", "Exponential", "Polynomial"]

rs = np.random.RandomState(42)
n = 250
x = 2 * rs.normal(size=n)
y = np.sin(2 * np.pi * x) + rs.normal(size=n) / 4
observed_data = [(x[i], y[i]) for i in range(n)]

plt.figure(num=1, figsize=(15, 8), dpi=80)
plt.xlim(-3, 3)
plt.ylim(-2, 3)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.scatter(x, y, facecolors="none", edgecolors="k", marker="o")

xx = np.arange(-3, 3, 0.1)
yy: list = [[] for _ in range(3)]
color = ["g", "b", "r"]

for i, method in enumerate(methods):
    match method:
        case "Gaussian":
            def k(x: float, y: float, sigma: float = 0.1) -> float:
                """Gaussian Kernel."""
                return np.exp(-np.linalg.norm(x - y)**2 / (2 * sigma**2))
        case "Exponential":
            def k(x: float, y: float, beta: float = 1.) -> float:
                """Exponential Kernel."""
                return np.exp(beta * np.dot(x, y))
        case "Polynomial":
            def k(x: float, y: float, m: int = 3) -> float:
                """Polynomial Kernel."""
                return (np.dot(x, y) + 1) ** m
        case _:
            msg = f"ValueError: Unknown method: {method}"

    for zz in xx:
        yy[i].append(f(observed_data, zz, k))
    plt.plot(xx, yy[i], c=color[i], label=method)

plt.legend(loc="upper left", frameon=True, prop={"size": 14})
plt.title("Nadaraya-Watson Estimator", fontsize=20)

# Save.
out_dir = Path("src/exercise/Chap1/out/problem2")
out_dir.mkdir(parents=True, exist_ok=True)
filename = Path(out_dir, "other_kernels.png")
plt.savefig(filename)
```



3. $A \in \mathbb{R}^{3\times3}$ の固有値を $\lambda_1, \lambda_2, \lambda_3$ とする。このとき、
任意の $t \in \mathbb{R}$ について、$I_3 \in \mathbb{R}^{3\times3}$ を単位行列として

$$\det(tI_3 - A) = (t-\lambda_1)(t-\lambda_2)(t-\lambda_3)$$

が成立する。(固有多項式より)
この式について $t=0$ を代入すると、

$$\det(-A) = (-1)^3 \lambda_1 \lambda_2 \lambda_3$$

となる。
今、$A \in \mathbb{R}^{3\times3}$ より、$\det(-A) = (-1)^3 \det(A)$ であるので、

$$\det(A) = \lambda_1 \lambda_2 \lambda_3 \quad \cdots (*)$$

が成り立つ。 ▨

また $(*)$ より、$\det(A) < 0$ のとき、$\lambda_1, \lambda_2, \lambda_3$ のうち、
少なくとも1つは負となることが分かる。よって、
非負定値行列の固有値は全て非負であるので、
$\det(A) < 0$ のとき、A は非負定値でない。 ▨

4. (前半)

$A, B \in \mathbb{R}^{n\times n}$ を非負定値行列とする。

まず, $A\circ B$ が対称行列であることは, アダマール積の定義より明らか。

A は非負定値行列より, ある直交行列 $U \in \mathbb{R}^{n\times n}$ が存在して,

$A = U\Lambda U^T$ と書ける。ここで, $\Lambda \in \mathbb{R}^{n\times n}$ は, A の固有値を対角成分に並べた対角行列である。同様に, $B = VMV^T$ と書ける。このとき,

U, V の第 i 列をそれぞれ u_i, v_i, A, B の第 i 固有値を λ_i, μ_i とすると,

$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T$, $B = \sum_{i=1}^{n} \mu_i v_i v_i^T$ が成立する。また, 行列

$(u_i u_i^T)\circ(v_j v_j^T)$ の (k, l) 成分について

$$((u_i u_i^T)\circ(v_j v_j^T))_{kl} = u_{ik} u_{il} v_{jk} v_{jl} = ((u_i\circ v_j)(u_i\circ v_j)^T)_{kl}$$

が成り立つので, $\underline{(u_i u_i^T)\circ(v_j v_j^T) = (u_i\circ v_j)(u_i\circ v_j)^T}$ である。

よって,

$$\begin{aligned} A\circ B &= \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \mu_i (u_i u_i^T)\circ(v_j v_j^T) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \mu_i (u_i\circ v_j)(u_i\circ v_j)^T \\ &\succeq O \end{aligned}$$

(*)

である。ここで最後の (*) については,

$\lambda_i \geq 0$ かつ, $\mu_i \geq 0$ と,

$(u_i\circ v_j)(u_i\circ v_j)^T \succeq O$ ($\because$ 命題1(1)より, 非負定値) と,

$c_1 \geq 0, c_2 \geq 0, A_1 \succeq O, A_2 \succeq O \Rightarrow c_1 A_1 + c_2 A_2 \succeq O$ を用いた。

$\forall x,\ x^T(c_1A_1 + c_2A_2)x = \underset{\geq 0}{c_1}\,\underset{\geq 0}{x^T A_1 x} + \underset{\geq 0}{c_2}\,\underset{\geq 0}{x^T A_2 x} \geq 0$

($A \succeq O$ は, A が非負定値行列であることを示す記号)

4.（後半）

k_1, k_2 を正定値カーネルとして，$k = k_1 k_2$ について考える。

この k の n 次 Gram 行列 G は，k_1, k_2 の n 次 Gram 行列をそれぞれ G_1, G_2 として，

$$G = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

$$= \begin{bmatrix} k_1(x_1, x_1) k_2(x_1, x_1) & \cdots & k_1(x_1, x_n) k_2(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k_1(x_n, x_1) k_2(x_n, x_1) & \cdots & k_1(x_n, x_n) k_2(x_n, x_n) \end{bmatrix}$$

$$= G_1 \circ G_2$$

が成り立つ。k_1, k_2 は正定値カーネルであるので，G_1, G_2 は非負定値である。さらに，非負定値行列のアダマール積はまた非負定値行列であるので，G は非負定値である。

よって，$k = k_1 k_2$ は正定値カーネルである。 ■

5. すべての成分が非負実数 $a \geq 0$ である正方行列 $A \in \mathbb{R}^{n \times n}$ は，

$A = a \mathbb{1}\mathbb{1}^T = (\sqrt{a}\,\mathbb{1})(\sqrt{a}\,\mathbb{1})^T$ と表せるため，

命題1より，A は非負定値である。 $\boxtimes$

$B = (\sqrt{a}\,\mathbb{1})^T \in \mathbb{R}^{1 \times n}$ とみる。

また，非負実定数 a を出力するカーネルのグラム行列は

$a\mathbb{1}\mathbb{1}^T \succeq O$ であるので，このようなカーネルは正定値カーネルである。

$\boxtimes$

6. $\Psi_{3,2}(x_1, x_2)$

$= [1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{6}x_1x_2, \sqrt{3}x_2^2, x_1^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, x_2^3]$

とすると、

$\Psi_{3,2}(x_1, x_2)^T \Psi(y_1, y_2)$

$= 1 + 3x_1y_1 + 3x_2y_2 + 3x_1^2y_1^2 + 6x_1x_2y_1y_2 + 3x_2^2x_2^2$

$\quad + x_1^3y_1^3 + 3x_1^2x_2y_1^2y_2 + 3x_1x_2^2y_1y_2^2 + x_2^3y_2^3$

$= (x^Ty)^3 + 3(x^Ty)^2 + 3x^Ty + 1$

$= (1 + x^Ty)^3 \quad (= k_{3,2}(x, y))$

7. (1). 指数型, Gauss カーネル, 多項式カーネルが正定値カーネルとなることを示す。

指数型

まず, $k(x,y) = x^T y$ は, 命題 2 の $\Phi(x) = x$, $H = \mathbb{R}^d$ の場合より, $\mathbb{R}^d$ 上の正定値カーネルである。 … ①

$$k_m(x,y) := 1 + \beta x^T y + \frac{\beta^2}{2}(x^T y)^2 + \cdots + \frac{\beta^m}{m!}(x^T y)^m$$

$(\beta > 0,\ m \geq 0)$ は, 各係数が非負であるような正定値カーネル $k(x,y) = x^T y$ の多項式であるので, 命題 4(1), (2) より正定値カーネルである。

また,

$$k_\infty(x,y) := \exp(\beta x^T y) = \lim_{m \to \infty} k_m(x,y)$$ は,

$\{k_m\}$ の極限である(テイラー展開)ため, 命題 4(3) より正定値カーネルである。 … ② ∎

Gauss カーネル

$$k(x,y) := \exp\left(-\frac{1}{2\sigma^2}\|x-y\|_2^2\right) \quad (\sigma > 0)$$ は,

$$k(x,y) = \underbrace{\exp\left(-\frac{\|x\|_2^2}{2\sigma^2}\right)}_{f(x)} \underbrace{\exp\left(\frac{x^T y}{\sigma^2}\right)}_{②\left(\beta = \frac{1}{\sigma^2}\right)} \underbrace{\exp\left(-\frac{\|y\|_2^2}{2\sigma^2}\right)}_{f(y)}$$

と書ける。よって, 命題 4 (5) と, ② の $\beta = \frac{1}{\sigma^2}$ の場合より, $\exp\left(\frac{x^T y}{\sigma^2}\right)$ が正定値カーネルであることから,

$k(x,y)$ は正定値カーネルである。

7 (1) (続き)

多項式カーネル

$k_{m,d}(x,y) := (x^\top y+1)^m$ は正定値カーネル $k(x,y)=x^\top y$ ($\because$ ①) の多項式であり、その各係数は非負であるので、命題 4 (1), (2) より正定値カーネルである。

(2) 正定値カーネルを正規化して得られるカーネルが正定値カーネルとなることを示す。

$k(x,y)$ を正定値カーネルとして、それを正規化したカーネルを $\bar{k}(x,y)$ とすると、

$$\bar{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)\,k(y,y)}}$$ となる。

$f(x) := \left(k(x,x)\right)^{-\frac{1}{2}}$, $f(y) := \left(k(y,y)\right)^{-\frac{1}{2}}$

とすれば、命題 4 (5) より、$\bar{k}(x,y)$ は正定値カーネルとなることが分かる。 ∎

7.(3). Gaussカーネル, 指数型を正規化する。

Gaussカーネル

$$\frac{\exp\left(-\frac{1}{2\sigma^2}\|x-y\|_2^2\right)}{\sqrt{\exp\left(-\frac{1}{2\sigma^2}\|x-x\|_2^2\right)\exp\left(-\frac{1}{2\sigma^2}\|y-y\|_2^2\right)}} = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|_2^2\right)$$

$(\because \exp(0)=1)$

よってGaussカーネルは正規化しても変化なし。

指数型

$$\frac{\exp(\beta x^\top y)}{\sqrt{\exp(\beta x^\top x)\exp(\beta y^\top y)}}$$

$$= \exp\left(-\frac{\beta}{2}(x^\top x + y^\top y - 2x^\top y)\right)$$

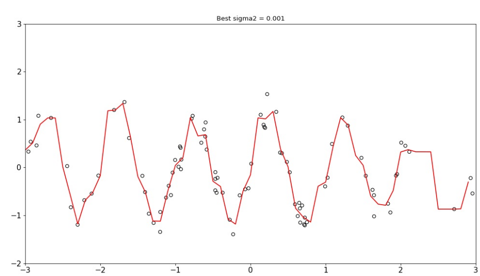
$$= \exp\left(-\frac{\beta}{2}\|x-y\|_2^2\right)$$

よって指数型を正規化すると、

$\sigma^2 = \beta^{-1}$ としたときのGaussカーネルとなる。

8.

```python
"""問題8"""
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np


def K(x: float, y: float, sigma2: float) -> float:
    """Gaussian Kernel."""
    return np.exp(-np.linalg.norm(x - y) ** 2 / (2 * sigma2))

rs = np.random.RandomState(42)
n = 100
x = 2 * rs.normal(size=n)
y = np.sin(2 * np.pi * x) + rs.normal(size=n) / 4

# 最適なsigma2の探索
sigma2_seq = np.arange(1e-3, 1e-2, 1e-3)
SS_min = np.inf
for sigma2 in sigma2_seq:
    SS = 0
    for k in range(n):
        test = [k]
        train = [x for x in range(n) if x not in test]
        for j in test:
            u, v = 0., 0.
            for i in train:
                kk = K(x[i], x[j], sigma2)
                u = u + kk * y[i]
                v = v + kk
            if v != 0:
                z = u / v
                SS = SS + (y[j] - z) ** 2
    if SS_min > SS:
        SS_min = SS
        sigma2_best = sigma2

print("Best sigma2 =", sigma2_best)

# 最適なsigma2を用いて曲線を表示
def f(observed_data: list[tuple[float, float]], x: float, sigma2: float) -> float:
    """Nadaraya-Watson Estimator."""
    numerator = np.sum([K(x, x_i, sigma2) * y_i for x_i, y_i in observed_data])
    denominator = np.sum([K(x, x_i, sigma2) for x_i, _ in observed_data])
    if denominator == 0:
        return 0.
    return numerator / denominator

xx = np.arange(-3, 3, 0.1)
observed_data = [(x[i], y[i]) for i in range(n)]
yy = [f(observed_data, zz, sigma2_best) for zz in xx]

# 表示
plt.figure(num=1, figsize=(15, 8), dpi=80)
plt.xlim(-3, 3)
plt.ylim(-2, 3)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.scatter(x, y, facecolors="none", edgecolors="k", marker="o")
plt.plot(xx, yy, c="r")
plt.title(f"Best sigma2 = {sigma2_best}")

# Save.
out_dir = Path("src/exercise/Chap1/out/problem8")
out_dir.mkdir(parents=True, exist_ok=True)
filename = Path(out_dir, "optimal_sigma2.png")
plt.savefig(filename)
```



leave-one-out で最良の $sigma^2$ を
用いたときの曲線
(オーバーフィッティングしている。)

9. Borel集合 $B=\{1\}$ について考える。このとき、

$$\{e\in E \mid X(e)\in B\}=\{1,3,5\}\notin \mathcal{F}=\{\{1,2,3\},\{4,5,6\},\{\},E\}$$

である。よって、$\mathcal{F}$ をこのように定義したとき、X は確率変数とならない。

10. 正規分布

$$f(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$ の特性関数 $\varphi(t)$ は,

$$\begin{aligned}
\varphi(t) &:= \mathbb{E}[\exp(itx)] \\
&= \int_{-\infty}^{\infty}\exp(itx)f(x)\,dx \\
&= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}+\frac{\mu x}{\sigma^2}+itx-\frac{\mu^2}{2\sigma^2}\right)dx \\
&= \underbrace{\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\{x-(\mu+it\sigma^2)\}^2}{2\sigma^2}\right)dx}_{(\text{平均 }\mu+it\sigma^2,\ \text{分散 }\sigma^2\text{ の正規分布の和})=1}\times\exp\left(i\mu t-\frac{t^2\sigma^2}{2}\right) \\
&= \exp\left(i\mu t-\frac{t^2\sigma^2}{2}\right)
\end{aligned}$$

である。よって, $\mu=0$ のとき, 任意の $t\in\mathbb{R}$ に対して, $\varphi(t)\in\mathbb{R}$ である。

Laplace 分布

$$f(x)=\frac{\alpha}{2}\exp(-\alpha|x|)$$ の特性関数 $\varphi(t)$ は,

$$\begin{aligned}
\varphi(t) &:= \int_{-\infty}^{\infty}\frac{\alpha}{2}\exp(itx)\exp(-\alpha|x|)\,dx \\
&= \frac{\alpha}{2}\left[\int_{-\infty}^{0}\exp\{(it+\alpha)x\}\,dx+\int_{0}^{\infty}\exp\{(it-\alpha)x\}\,dx\right] \\
&= \frac{\alpha}{2}\left\{\left[\frac{e^{(it+\alpha)x}}{it+\alpha}\right]_{-\infty}^{0}+\left[\frac{e^{(it-\alpha)x}}{it-\alpha}\right]_{0}^{\infty}\right\} \\
&= \frac{\alpha^2}{t^2+\alpha^2}
\end{aligned}$$

であるので, 任意の $t\in\mathbb{R}$ に対して, $\varphi(t)\in\mathbb{R}$ である。

11.

```python
"""問題11"""
import numpy as np


def k(s: list, t: list) -> int:
    """木カーネル。

    木sのi番目の要素s[i]は
    s[i] = [{頂点iのラベル}, [{頂点iの子}]]
    である。
    """
    m, n = len(s), len(t)
    def C(i: int, j: int) -> int:
        """頂点i, jを根とする共通の部分木の個数。"""
        S, T = s[i], t[j]
        # 木sの頂点iまたは木tの頂点jが子孫を持たない場合0。
        if S[1] is None or T[1] is None:
            return 0
        # 木sの頂点iと木tの頂点jのラベルが一致しないときは0。(1)(a)
        if S[0] != T[0]:
            return 0
        # 子の数が異なるときは0。(1)(b)
        if len(S[1]) != len(T[1]):
            return 0
        U = [s[x][0] for x in S[1]]
        U1 = sorted(U)
        V = [t[y][0] for y in T[1]]
        V1 = sorted(V)
        m = len(U)
        # 子のラベルが一致しないときは0。(1)(c)
        for h in range(m):
            if U1[h] != V1[h]:
                return 0
        U2 = np.array(S[1])[np.argsort(U)]
        V2 = np.array(T[1])[np.argsort(V)]
        W = 1
        for h in range(m):
            W *= 1 + C(U2[h], V2[h])
        return W

    kernel = 0
    for i in range(m):
        for j in range(n):
            if C(i, j) > 0:
                kernel += C(i, j)
    return kernel

if __name__ == "__main__":
    s = [
        ["G", [1, 3]],
        ["T", [2]],
        ["C", None],
        ["A", [4, 5]],
        ["C", None],
        ["T", None],
    ]
    t = [
        ["G", [1, 4]],
        ["A", [2, 3]],
        ["C", None],
        ["T", None],
        ["T", [5, 6]],
        ["C", None],
        ["A", [7, 8]],
        ["C", None],
        ["T", None],
    ]
    print(f"{k(s, s)=}")
    # k(s, s)=6
```

$k(s,s) = 6$.

12.

```python
"""問題12（誤植訂正後）"""

def string_kernel(x: str, y: str, p: int) -> int:
    """String kernel."""
    m, n = len(x), len(y)
    if m < p or n < p:
        return 0
    S = 0
    for i in range(m - p):
        for j in range(n - p):
            if x[i:i + p] == y[j:j + p]:
                S += 1
    return S

if __name__ == "__main__":
    import numpy as np
    rs = np.random.RandomState(42)
    x = "".join(rs.choice(["0", "1"], size=10))
    y = "".join(rs.choice(["0", "1"], size=10))
    print(f"{x=}")
    # x='0100010001'
    print(f"{y=}")
    # y='0000101110'
    print(f"{string_kernel(x, y, 2)=}")
    # string_kernel(x, y, 2)=18
```

13. 文字列カーネル

長さが p の文字列の集合を Σ^p, 有限長の文字列(空文字列を含む)の集合を $\Sigma^* = \bigsqcup_i \Sigma^i$ とする。さらに, $E_1 = E_3 = \Sigma^*$, $E_2 = \Sigma^p$ とする。

また, $(x_1, x_2, x_3) \in E_1 \times E_2 \times E_3$ を連結すると,

$R(x_1, x_2, x_3) = x \in E$ とみなすことができる。

そして, $k_1 : E_1 \times E_1 \to \mathbb{R}$, $k_2 : E_2 \times E_2 \to \mathbb{R}$, $k_3 : E_3 \times E_3 \to \mathbb{R}$ をそれぞれ

$$k_1(x_1, y_1) = 1,\quad k_2(x_2, y_2) = I(x_2 = y_2) = \begin{cases} 1 & \text{if } x_2 = y_2, \\ 0 & \text{otherwise} \end{cases}$$ とすると,

k_1, k_2, k_3 はそれぞれ正定値カーネルであり, 文字列カーネルは,

$x_2 = u$ となるものが x に $C_u(x)$ 個, $y_2 = u$ となるものが y に $C_u(y)$ 個あれば,

$$C_u(x)C_u(y) = \sum_{R(x_1,x_2,x_3)=x} \sum_{R(y_1,y_2,y_3)=y} k_1(x_1, y_1)\, k_2(x_2, y_2 : x_2 = y_2 = u)\, k_3(x_3, y_3)$$

となり, $\underset{\text{文字列カーネル}}{k(x,y)} = \sum_u C_u(x) C_u(y) = \sum_{R(x_1,x_2,x_3)=x} \sum_{R(y_1,y_2,y_3)=y} k_1(x_1, y_1) k_2(x_2, y_2) k_3(x_3, y_3)$

となる。これは式(1.10)の形であるので, 文字列カーネルはたたみこみカーネルである。また, k_1, k_2, k_3 は正定値カーネルであるので, 命題4から,

文字列カーネルは正定値カーネルである。 □

13. (続き)

木カーネル

木 x, y に対して，$C_t(x)$, $C_t(y)$ をそれぞれ木 x, y における部分木 t の頻度とする。このとき，木 $x_1, \dots, x_n \in E$ および任意の $z_1, \dots, z_n \in \mathbb{R}$ について，

木カーネル

$$\sum_{i=1}^{n}\sum_{j=1}^{n} z_i z_j k(x_i, x_j) = \sum_t \left\{\sum_{i=1}^{n} z_i C_t(x_i)\right\}^2 \geq 0$$

が成り立つ。

よって，(1.8) 式から，木カーネルは正定値カーネルである。

周辺化カーネル

離散値確率変数 X, Y について，ありうる値の集合をそれぞれ E_X, E_Y とする。また，$E_{XY} := E_X \times E_Y$ について，$k_{XY} : E_{XY} \times E_{XY} \to \mathbb{R}$ を正定値カーネルとする。

このとき，ある特徴写像 $\Phi : E_{XY} \ni (x, y) \mapsto \Phi(x, y)$ が存在し，

$$k_{XY}((x, y), (x', y')) = \langle \Phi((x, y)), \Phi((x', y')) \rangle$$ が成り立つ。

よって，

周辺化カーネル

$$k(x, x') = \sum_{y \in E_Y} \sum_{y' \in E_Y} P(y|x) P(y'|x') \langle \Phi((x, y)), \Phi((x', y')) \rangle$$

$$= \left\langle \sum_{y \in E_Y} P(y|x)\Phi((x, y)), \sum_{y' \in E_Y} P(y'|x')\Phi((x', y')) \right\rangle$$

が成り立つため，新たに特徴写像

$$\Phi' : E_X \to \mathcal{H}, \quad \Phi'(x) := \sum_{y \in E_Y} P(y|x)\Phi((x, y))$$ を定義すれば，

$$k(x, x') = \langle \Phi'(x), \Phi'(x') \rangle_{\mathcal{H}}$$ と書けるので，

周辺化カーネルは正定値カーネルである。

13.（続き）

グラフカーネル

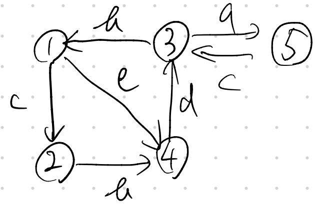
$$k(G_1, G_2) := \sum_{\pi_1} \sum_{\pi_2} P(\pi_1 | G_1) P(\pi_2 | G_2) I(L(\pi_1) = L(\pi_2))$$

について，$k_{XY}((G_1, \pi_1), (G_2, \pi_2)) = I(L(\pi_1) = L(\pi_2))$ とすれば，

周辺化カーネルであることが分かる。

（このことから，グラフカーネルは正定値カーネルであることが分かる。）

14. (a) $\frac{1}{5}\left(\frac{2}{3}\cdot\frac{1}{2}\right)\left(\frac{2}{3}\cdot\frac{1}{2}\right)\left(\frac{2}{3}\cdot 1\right)\left(\frac{2}{3}\cdot\frac{1}{2}\right)\cdot\frac{1}{3}$



$p=\frac{1}{3}$,

$$=\frac{1}{5}\cdot\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}$$

$$=\frac{2}{5\cdot 3^5}=\frac{2}{1215}$$

(b) $\frac{1}{5}\left(\frac{2}{3}\cdot\frac{1}{2}\right)\left(\frac{2}{3}\cdot 1\right)\left(\frac{2}{3}\cdot 0\right)\left(\frac{2}{3}\cdot\frac{1}{2}\right)\cdot\frac{1}{3}=0$

(c) $\frac{1}{5}\left(\frac{2}{3}\cdot\frac{1}{2}\right)\left(\frac{2}{3}\cdot 1\right)\left(\frac{2}{3}\cdot\frac{1}{2}\right)\frac{1}{3}$

$$=\frac{1}{5}\cdot\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}$$

$$=\frac{2}{5\cdot 3^4}=\frac{2}{405}$$

15.

```python
"""問題15"""

def prob(node: list[list[int]], s: list[int], p: float) -> float:
    """Prob."""
    if len(node[s[0]]) == 0:
        return 0
    m = len(s)
    if m == 1:
        return p
    return (1 - p) / len(node[s[0]]) * prob(node, s[1:m], p)

def k(node: list[list[int]], s: list[int], p: float) -> float:
    """Graph kernel."""
    return prob(node, s, p) / len(node)

if __name__ == "__main__":
    node = [
        [1, 3],
        [3],
        [0, 4],
        [2],
        [2],
    ]
    # 1-indexedとして経路1->3は存在し得ないものである。（頂点1から3への有向辺は存在しないので。）
    # このとき、p(\pi = <1, 3>))=0となるべきである。
    print(f"{k(node, [0, 2], 1/2)=}")
    # 0.025
    # しかし上記の実行結果のように、非ゼロの値が得られる。
```