MertonModel NBD

弘前大学4年：坂西和也

ガウス過程回帰

参考資料

- 『ガウス過程と機械学習』(持橋大地／大羽成征・著 2019/03/07)
- 『Gaussian Processes for Machine Learning』https://gaussianprocess.org/gpml/ (https://gaussianprocess.org/gpml/)
- 『パターン認識と機械学習』https://www.maruzen-publishing.co.jp/item/?book_no=294524 (https://www.maruzen-publishing.co.jp/item/?book_no=294524)

確率過程の定義

確率過程$y(x)$とは，任意の有限な値集合$y(x_1), \ldots, y(x_N)$に対して，矛盾のない同時分布を与えるもの．

ガウス過程の定義

関数$y(x)$の上の確率分布として定義され，任意のxに対して$y(x)$がガウス分布に従うような確率過程．記号として一般に$\mathcal{GP}(\vec{m}, K)$を用いる．

カーネル関数

カーネル関数は，ガウス過程における分散共分散行列Kの各要素であり，ガウス過程の性質を決定する．

以下でよく用いられるカーネル関数を示す．

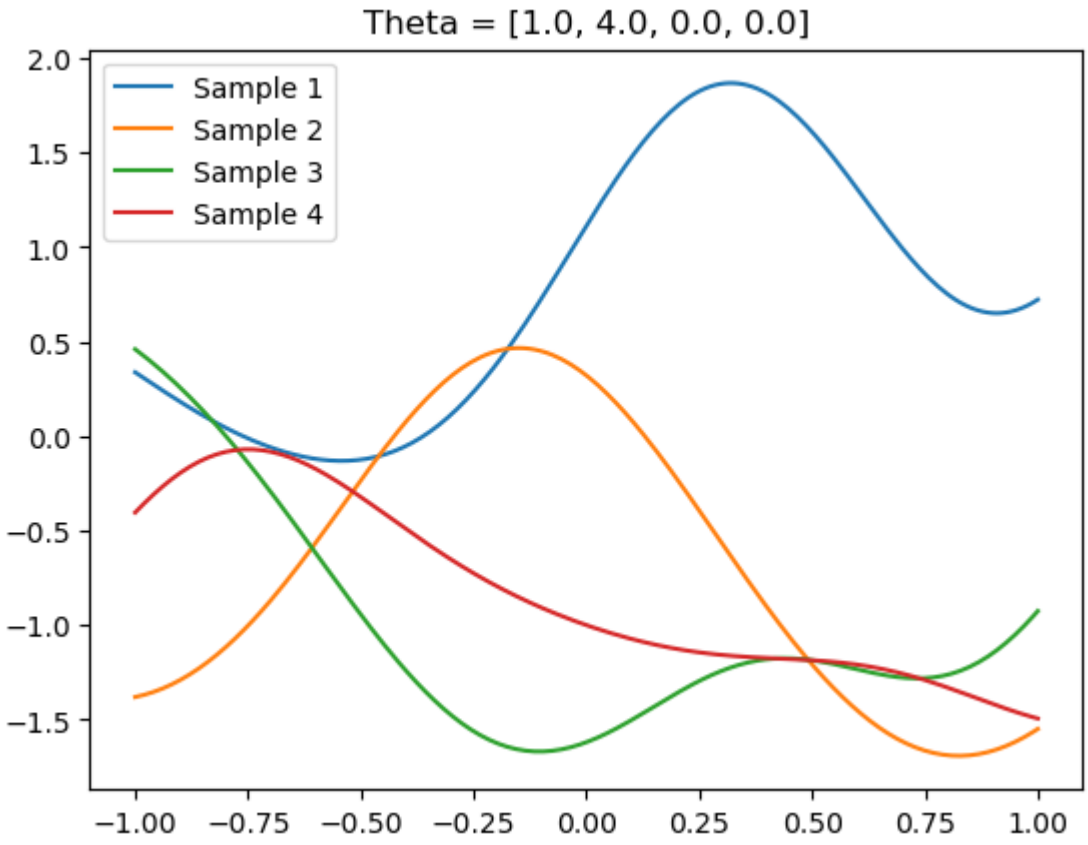
$$k(x, x') = \theta_0 \exp\left\{ -\frac{\theta_1}{2} \| x - x' \|^2 \right\} + \theta_2 + \theta_3 x^T x'$$

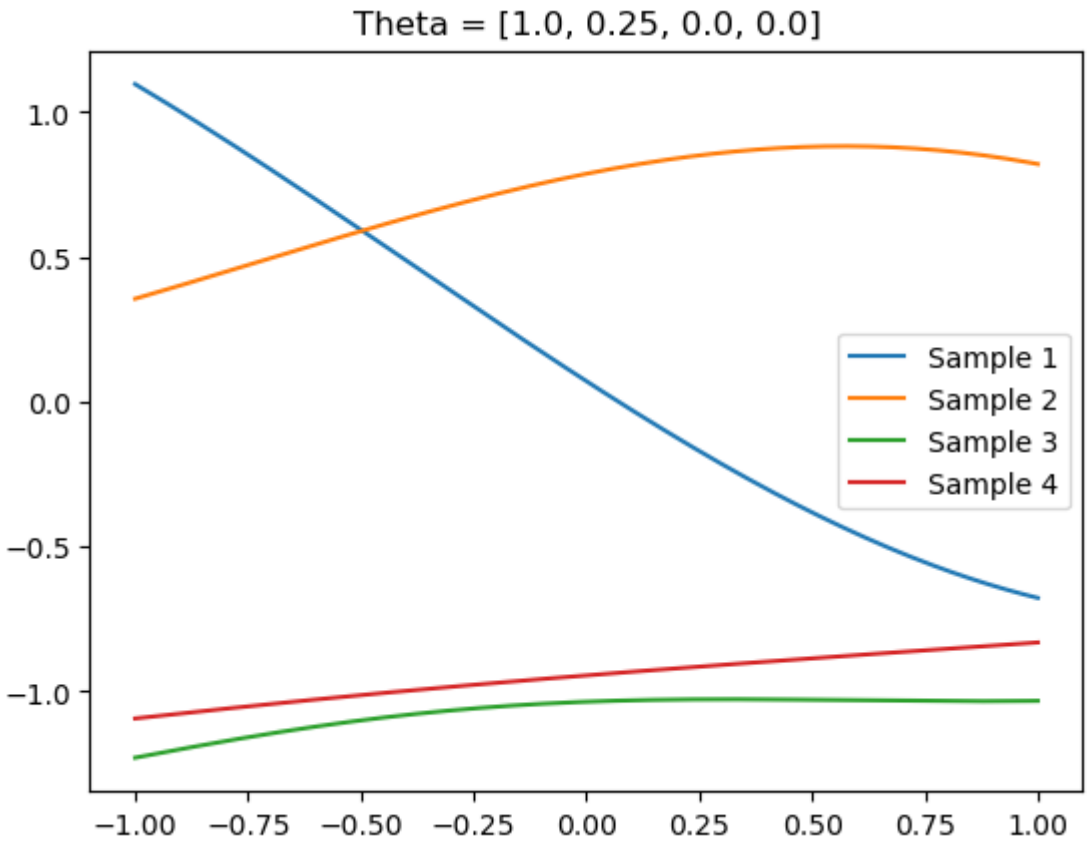
様々なパラメータを設定したときの，ガウス過程による事前分布からのサンプルを示す．

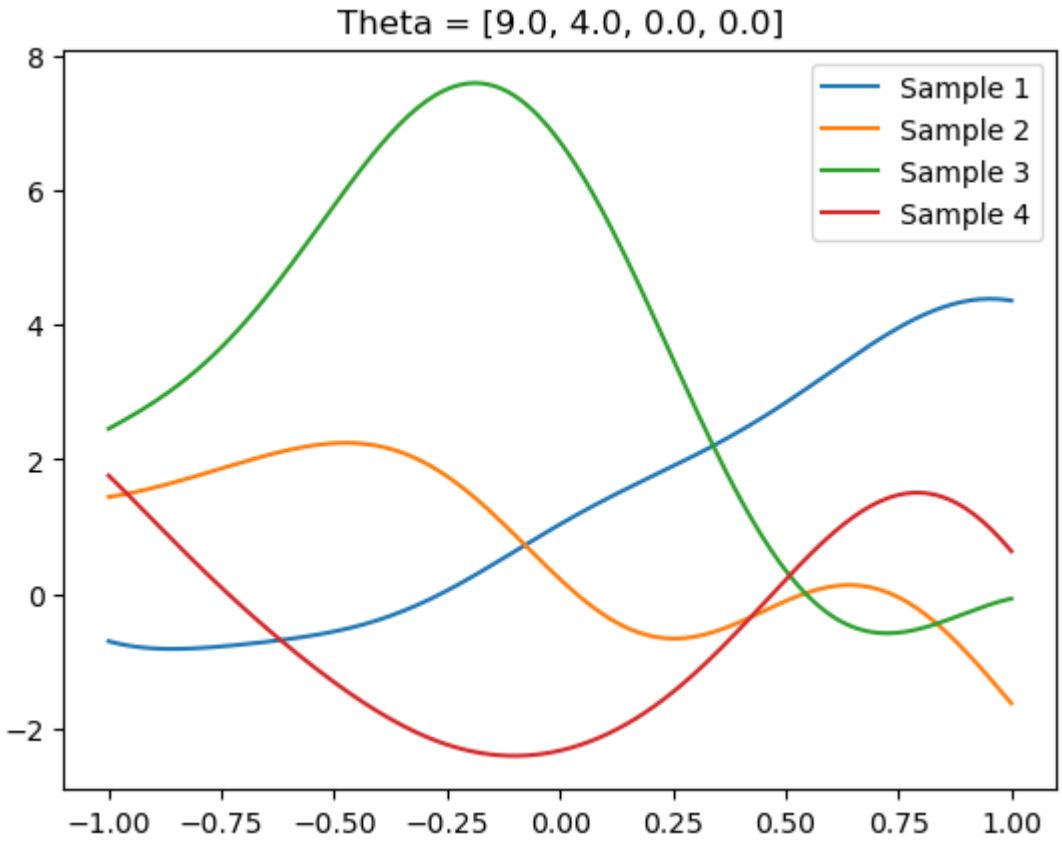
入力 [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

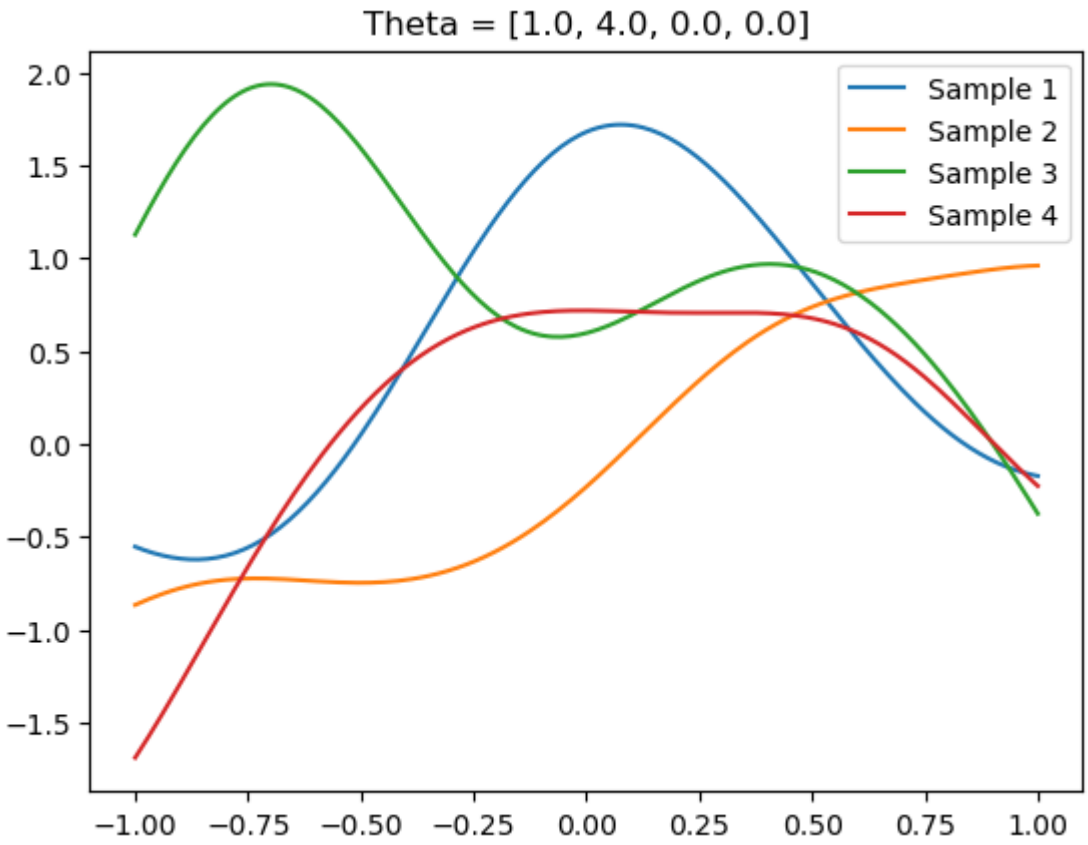
入力 [3]:

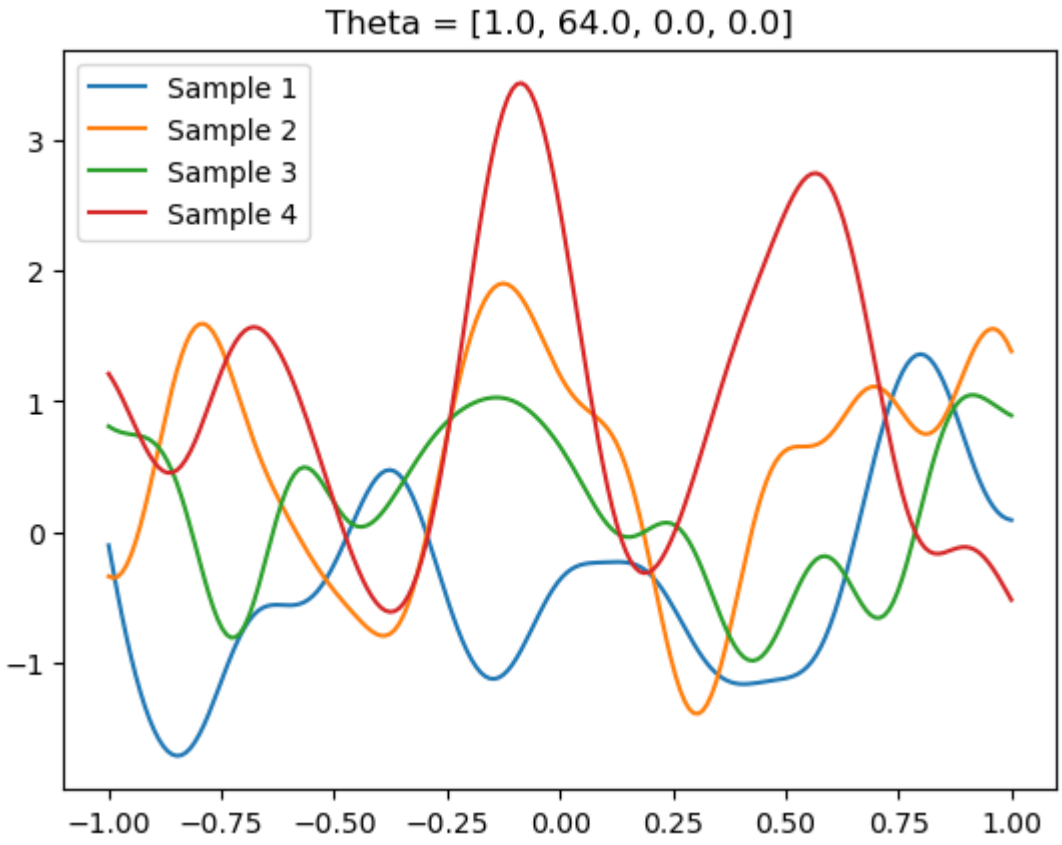
```
# カーネル関数
def kernel(x1, x2, theta):
    return theta[0] * np.exp(-0.5 * theta[1] * np.linalg.norm(x1 - x2)**2) \
        + theta[2] + theta[3] * np.dot(x1, x2)
# グラム行列（カーネル行列）を計算
def compute_gram_matrix(X, theta):
    n = X.shape[0]
    K = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            K[i, j] = kernel(X[i], X[j], theta)
    return K
# ガウス過程からサンプルを生成
def sample_gp(X, theta, num_samples):
    # グラム行列を計算
    K = compute_gram_matrix(X, theta)

    # ガウス過程からサンプルを生成
    samples = np.random.multivariate_normal(np.zeros(len(X)), K, num_samples)

    return samples
# パラメータセット
theta_sets = [
    [1.00, 4.00, 0.00, 0.00],
    [1.00, 0.25, 0.00, 0.00],
    [9.00, 4.00, 0.00, 0.00],
    [1.00, 4.00, 0.00, 0.00],
    [1.00, 64.00, 0.00, 0.00],
    [1.00, 4.00, 0.00, 5.00]
]
# 入力値
X = np.linspace(-1, 1, 200)

np.random.seed(123)
for i, theta in enumerate(theta_sets, start=1):
    # サンプリング
    samples = sample_gp(X, theta, 4)

    # プロット
    plt.figure(i)
    for j in range(4):
        plt.plot(X, samples[j], label=f'Sample {j+1}')
    plt.title(f'Theta = {theta}')
    plt.legend()

plt.show()
```

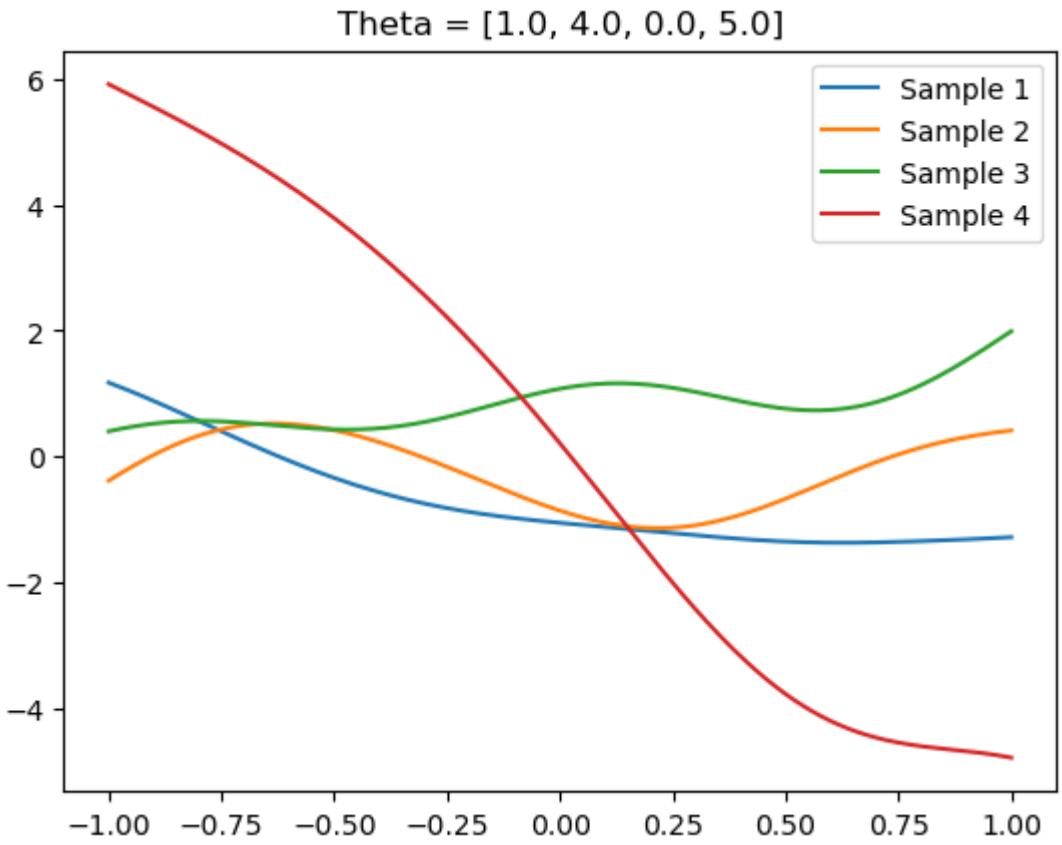

Theta = [1.0, 4.0, 0.0, 0.0]
Sample 1
Sample 2
Sample 3
Sample 4
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Theta = [1.0, 0.25, 0.0, 0.0]
Sample 1
Sample 2
Sample 3
Sample 4
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Theta = [9.0, 4.0, 0.0, 0.0]
Sample 1
Sample 2
Sample 3
Sample 4
8
6
4
2
0
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Theta = [1.0, 4.0, 0.0, 0.0]
Sample 1
Sample 2
Sample 3
Sample 4
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00





ガウス過程による回帰

観測される目標変数の値に含まれるノイズを考える．

$$t_n = y(x_n) + \epsilon_n$$
$$\epsilon_n = \mathcal{N}(\epsilon_n|0, \beta^{-1})$$

ノイズは各データ点に対して独立に決定されガウス分布とする．

$y = (y_1, \ldots, y_N)^T$が与えられた下での目標値$t = (t_1, \ldots, t_N)^T$の同時分布は以下の等方的なガウス分布に従う．

$$p(t|y) = \mathcal{N}(t|y, \beta^{-1} I_N)$$

ガウス過程の定義から，yの同時分布は以下のガウス分布に従う．

$$p(y) = \mathcal{N}(y|0, K)$$

ただし，Kはカーネル行列であり，$K_{nm} = k(x_n, x_m)$である．

入力値xが与えられた下での目標値tの周辺分布は計算により以下のガウス分布に従う．

$$p(t) = \mathcal{N}(t|0, C)$$

ただし，$C = K + \beta^{-1} I_N$である．

予測分布$p(t^*|t)$を求める．

まず，t^*とtの同時分布$p(t^*, t)$を求める．

$$p(t^*, t) = \mathcal{N}\left(\begin{bmatrix} t \\ t^* \end{bmatrix} \middle| 0, \begin{bmatrix} C & k_* \\ k_*^T & c \end{bmatrix} \right)$$

ただし，$[k_*]_n = k_{*n} = k(x_*, x_n)$であり，$c = k(x_*, x_*) + \beta^{-1}$である．

ここから，データ点tが与えられたときのt^*の周辺分布$p(t^*|t)$を求めと

$$p(t^*|t) = \mathcal{N}(t^*|m(x_*), \sigma^2(x_*))$$

ただし，$m(x_*) = k^T C^{-1} t$，$\sigma^2(x_*) = c - k^T C^{-1} k$である

ハイパーパラメータの最適化

ガウス過程による回帰モデルにおける対数尤度関数は，標準的なガウス分布を用いて以下のように表される．

$$\ln p(t|\theta) = -\frac{1}{2} \ln |C| - \frac{1}{2} t^T C^{-1} t - \frac{N}{2} \ln(2\pi)$$

各パラメータの微分は以下のようになる．

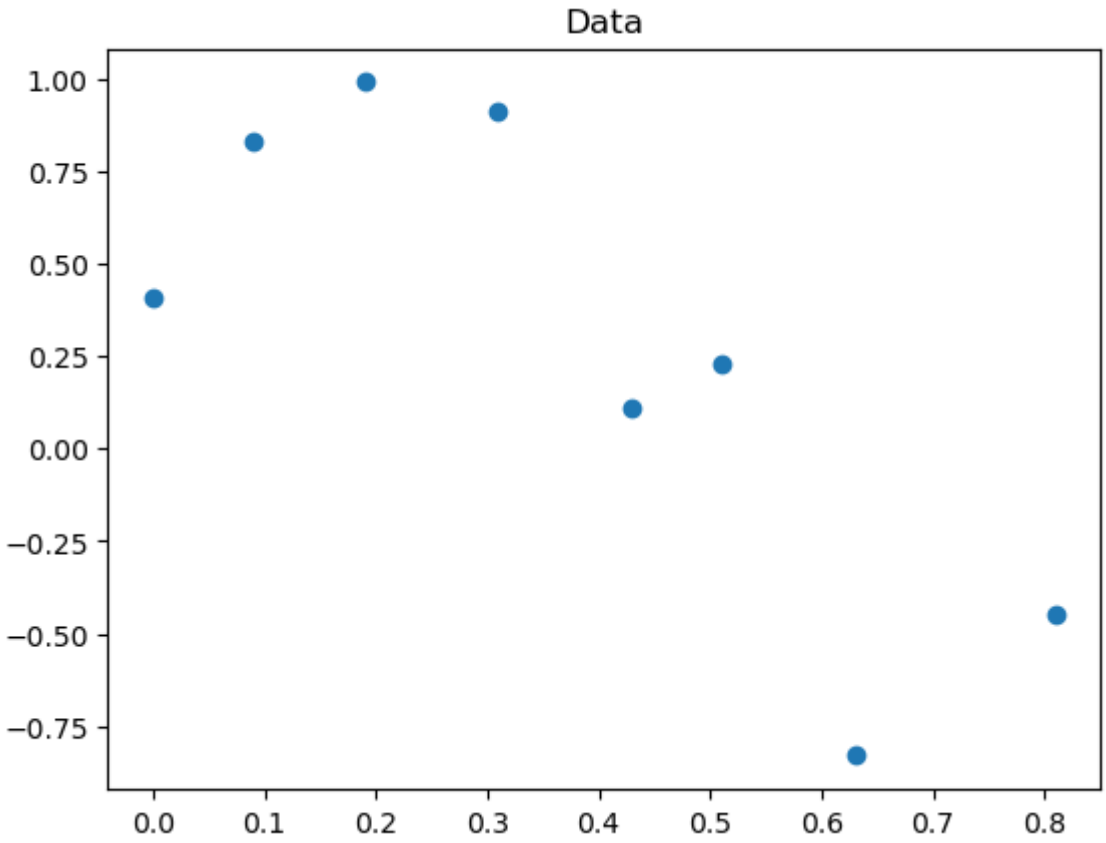
$$\frac{\partial}{\partial \theta_i} \ln p(t|\theta) = \frac{1}{2} \mathrm{Tr}\left(C^{-1} \frac{\partial C}{\partial \theta_i} \right) - \frac{1}{2} t^T C^{-1} \frac{\partial C}{\partial \theta_i} C^{-1} t$$

一般には最大化する$\ln p(t|\theta)$は非凸であり，局所解に陥りやすい．

簡単な実装

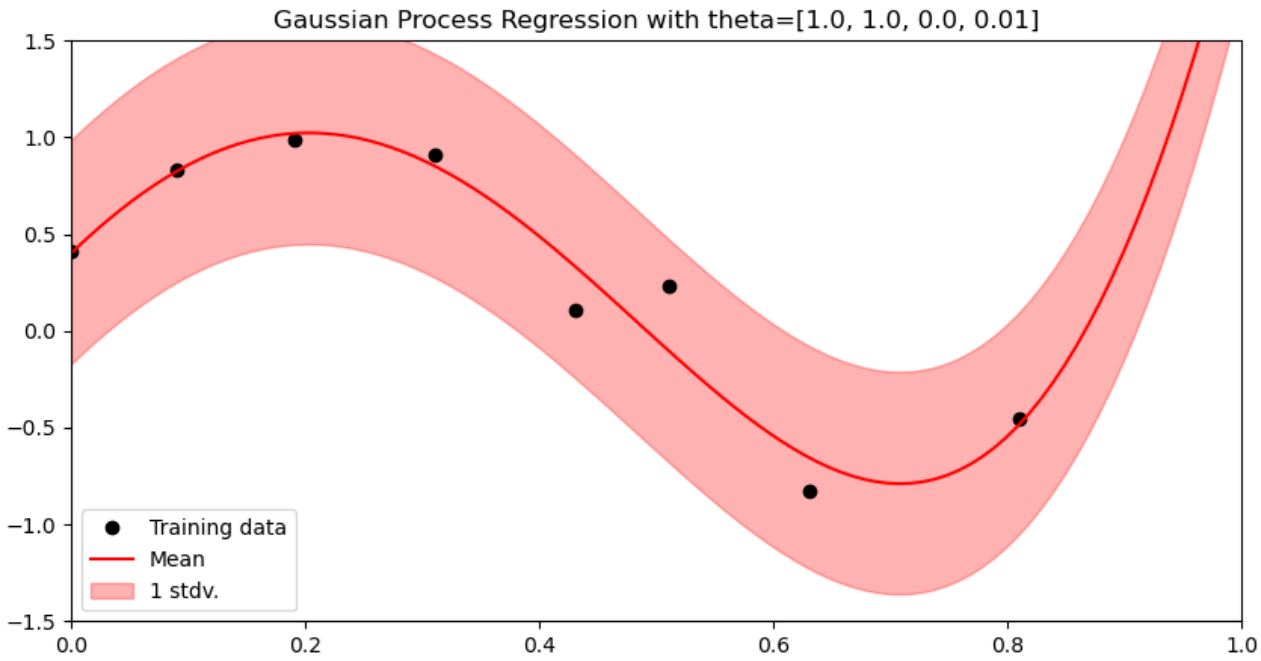
入力 [4]:

```
# データの生成
X = np.array([0.00 , 0.09 , 0.19 , 0.31 , 0.43 , 0.51 , 0.63 , 0.81])
y = np.array([0.41 , 0.83 , 0.99 , 0.91 , 0.11 , 0.23 , -0.83 , -0.45])

# プロット
plt.figure(0)
plt.scatter(X, y)
plt.title('Data')
plt.show()
```



入力 [6]:

```
# ガウス過程回帰をクラスでまとめる
class GaussianProcessRegressor:
    def __init__(self, kernel, theta):
        self.kernel = kernel
        self.theta = theta
    # グラム行列（カーネル行列）を計算
    def compute_gram_matrix(self, X1, X2):
        n, m = X1.shape[0], X2.shape[0]
        K = np.zeros((n, m))
        for i in range(n):
            for j in range(m):
                K[i, j] = self.kernel(X1[i], X2[j], self.theta)
        return K
    # 学習
    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train
        self.K = self.compute_gram_matrix(X_train, X_train)
        self.K_inv = np.linalg.inv(self.K + 1e-6 * np.eye(len(self.K)))  # Regularization
    # 予測
    def predict(self, X_test):
        k = self.compute_gram_matrix(X_test, self.X_train)
        mu = k @ self.K_inv @ self.y_train
        var = np.diag(self.kernel(X_test, X_test, self.theta) - k @ self.K_inv @ k.T)
        return mu, var

# Kernel function
def kernel(x1, x2, theta):
    return theta[0] * np.exp(-0.5 * theta[1] * np.linalg.norm(x1 - x2)**2) \
        + theta[2] + theta[3] * np.dot(x1, x2)

# Parameter
theta = [1.00, 1.00, 0.00, 0.01]

# Test data
X_test = np.linspace(0, 1, 100)

# Create Gaussian process regressor and fit to data
gpr = GaussianProcessRegressor(kernel, theta)
gpr.fit(X, y)

# Predict using the GPR
mu, var = gpr.predict(X_test)
stdv = np.sqrt(var)

# Plot
plt.figure(figsize=(10, 5))
plt.plot(X, y, 'ko', label='Training data')
plt.plot(X_test, mu, 'r', label='Mean')
plt.fill_between(X_test, mu - stdv, mu + stdv, color='r', alpha=0.3, label='1 stdv.')
plt.title(f'Gaussian Process Regression with theta={theta}')
plt.xlim(0, 1)
plt.ylim(-1.5, 1.5)
plt.legend()
plt.show()
```



尤度を変えてガウス過程回帰を行う

ベルヌーイ尤度（ロジスティック回帰をガウス過程を用いて）

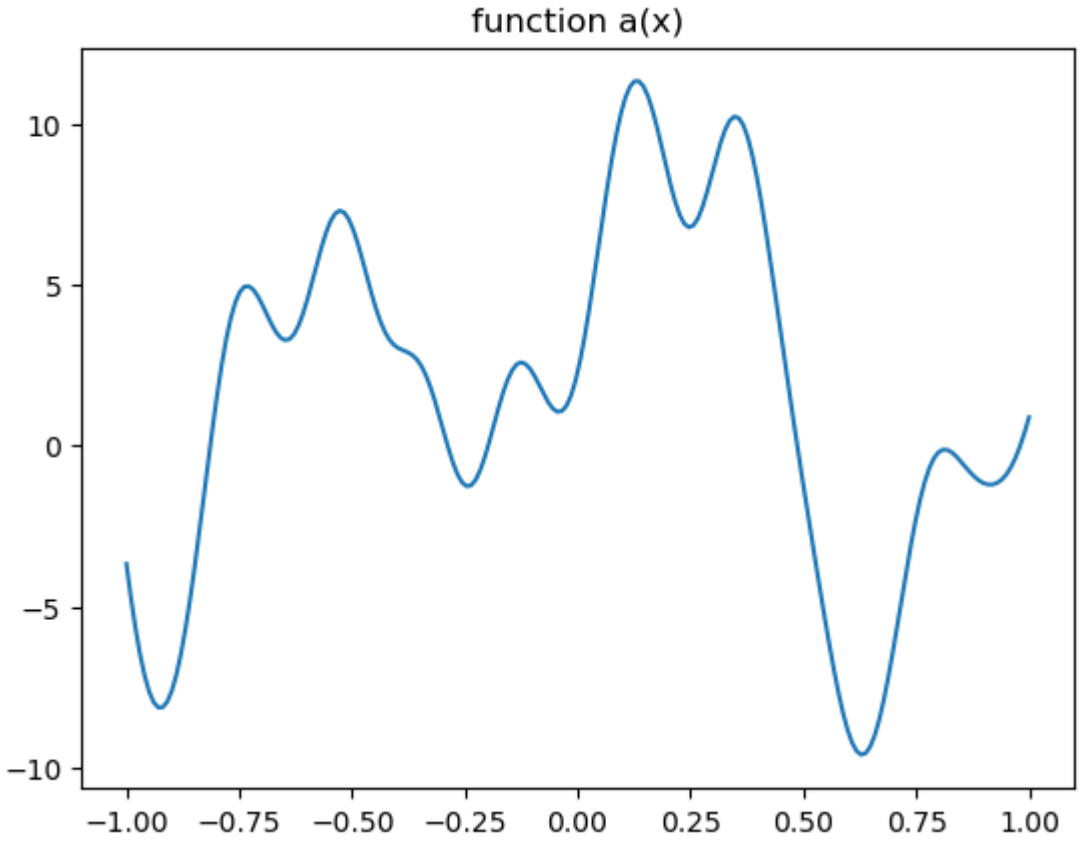
関数$a(x)$の上でのガウス過程を定義する．

これをシグモイド関数$y = \sigma(a)$で変換することで，$y \in (0, 1)$であるような関数$y(x)$の上での非ガウス確率過程が得られる．

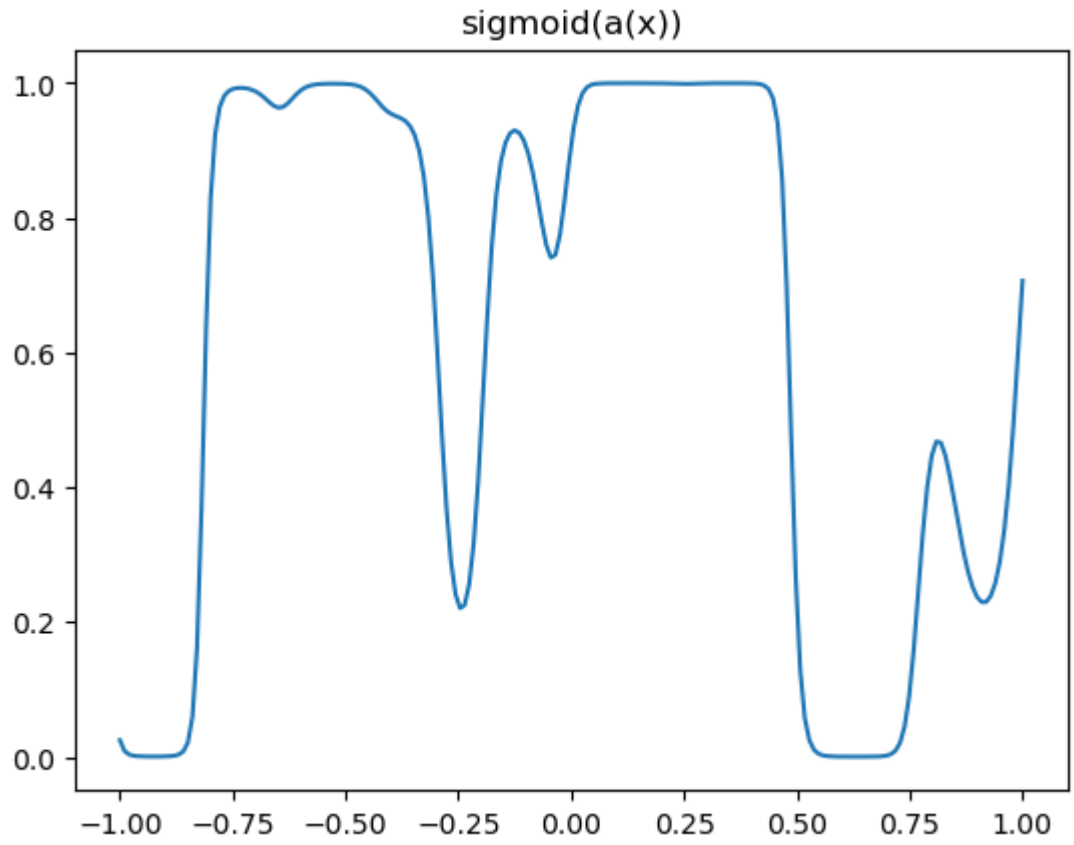
以下にガウス過程からのサンプルをシグモイド関数で変換した様子を示す．

入力 [8]:

```
# ガウス過程からのサンプリング
from Gaussian_method import *
import numpy as np
import matplotlib.pyplot as plt

theta = [50.0, 100.0, 15.0, 10.0]

# 入力値
X = np.linspace(-1, 1, 200)

np.random.seed(0)
samples = sample_gp(X, theta , 2)

# プロット
plt.plot(X, samples[1])
plt.title(f'function a(x)')
plt.show()
```



入力 [9]:
```
# シグモイド関数で変換
def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

y = sigmoid(samples[1])

# プロット
plt.plot(X, y)
plt.title(f'sigmoid(a(x))')
plt.show()
```



目標変数tの確率分布はベルヌーイ分布となる．

$$p(t|y(a(x))) = \sigma(a(x))^t(1-\sigma(a(x)))^{1-t}$$

目標は予測分布$p(t^*|t)$を決定することである．

$$p(t^*=1|t) = \int p(t^*=1|a^*)p(a^*|t)da^* = \int \sigma(a^*)p(a^*|t)da^*$$

この積分が解析的には不可能であるから，様々な近似手法が必要になる．

今回は『パターン認識と機械学習』に習ってラプラス近似を用いる．

a^*の事後分布はベイズの定理と$p(t|a^*,a)=p(t|a)$より

$$p(a^*|t) = \int p(a^*|a,t)p(a|t)da = \int p(a^*|a)p(a|t)da$$

期待値と分散は計算をすると

$$\mathbb{E}[a^*|t] = k_*^T(t-\sigma(a))$$
$$\mathrm{var}[a^*|t] = k(x^*,x^*) - k_*^T(W^{-1}+C)^{-1}k_*$$

ただし，Wは$\sigma(a_n)(1-\sigma(a_n))$を対角要素とする$N\times N$の対角行列である．

以上から予測分布は

$$p(t^*=1|t) = \int \sigma(a^*)p(a^*|t)da^* \approx \int \sigma(a^*)\mathcal{N}(a^*|\mathbb{E}[a^*|t],\mathrm{var}[a^*|t])da^*$$

と近似して

$$p(t^*=1|t) \approx \sigma(\mathbb{E}[a^*|t]\kappa(\mathrm{var}[a^*|t]))$$

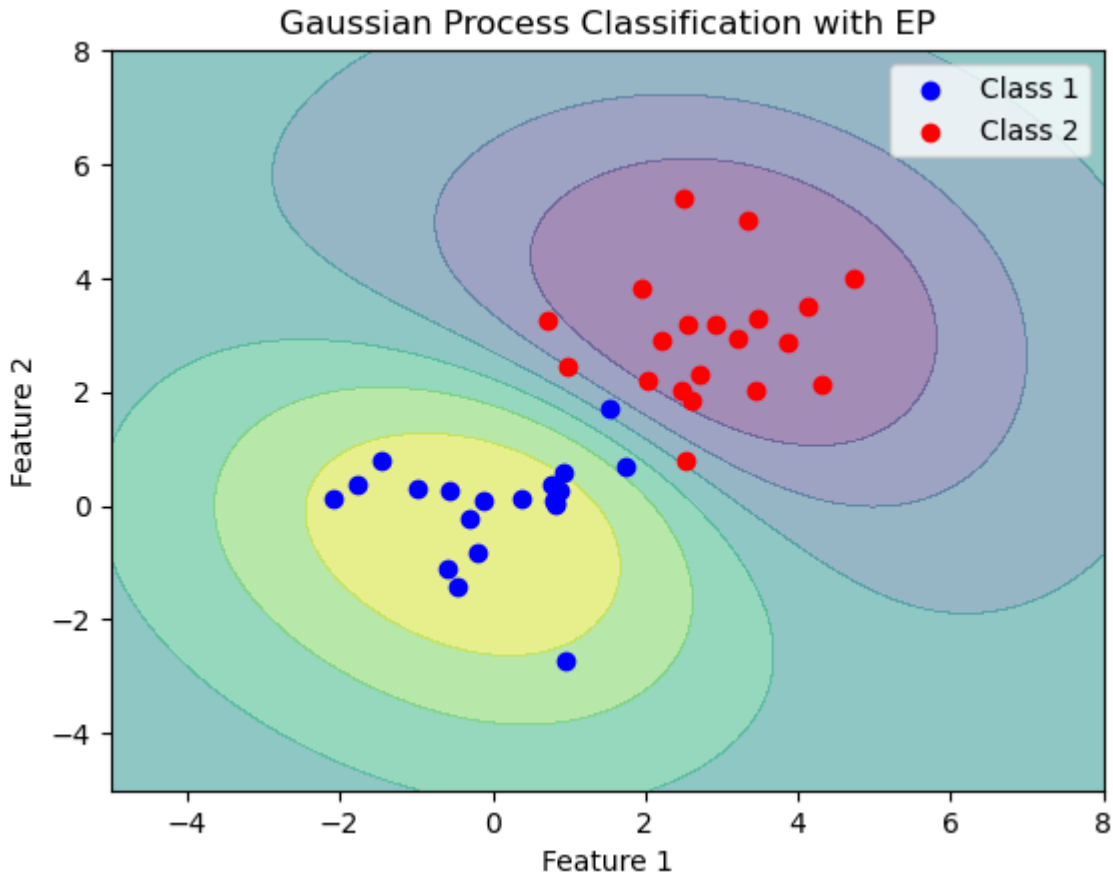
ただし，$\kappa(s^2) = \dfrac{1}{\sqrt{1 + \frac{\pi s^2}{8}}}$

以上のように尤度をガウス分布以外にすると近似手法をうまく使って計算する必要がある．

以下では,ベルヌーイ尤度の場合にうまく実装ができる期待値伝搬法(EP法)を用いた実装を行う．このような実装はGPvをつかって簡単に実装ができる．

入力 [10]:

```
# EP法による近似を用いた実装
import numpy as np
import GPy
import matplotlib.pyplot as plt

# サンプルデータを生成します。
# クラス1のデータ
X1 = np.random.randn(20, 2)
Y1 = np.ones((20, 1))

# クラス2のデータ
X2 = np.random.randn(20, 2) + 3
Y2 = np.zeros((20, 1))

# データを結合します。
X = np.vstack((X1, X2))
Y = np.vstack((Y1, Y2))

# ガウス過程分類モデルを作成します。
kernel = GPy.kern.RBF(input_dim=2, variance=1., lengthscale=1.)
model = GPy.models.GPClassification(X, Y, kernel=kernel)

# EP法を使用して最適化します。
model.optimize('bfgs', max_iters=100)

# 新しいデータ点の予測を行います。
x_test = np.linspace(-5, 8, 100).reshape(-1, 1)
y_test = np.linspace(-5, 8, 100).reshape(-1, 1)
x_test_grid, y_test_grid = np.meshgrid(x_test, y_test)
X_test = np.vstack((x_test_grid.ravel(), y_test_grid.ravel())).T
probs, _ = model.predict(X_test)

# 予測をプロットします。
plt.contourf(x_test_grid, y_test_grid, probs.reshape(100, 100), alpha=0.5)
plt.scatter(X1[:, 0], X1[:, 1], c='blue', label='Class 1')
plt.scatter(X2[:, 0], X2[:, 1], c='red', label='Class 2')
plt.legend()
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Gaussian Process Classification with EP')
plt.show()
```



ガウス過程ポアソン回帰

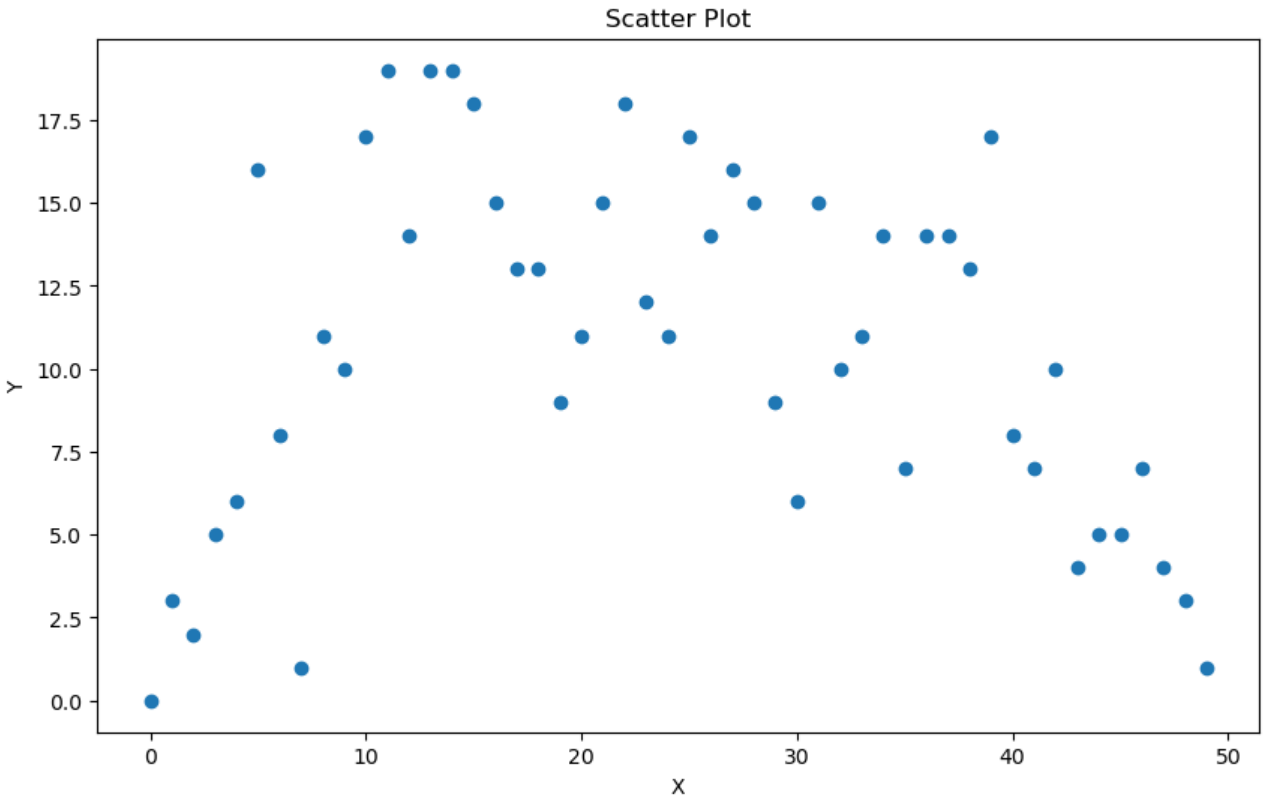
事前分布にガウス過程を導入したものがガウス過程ポアソン回帰.

生成モデルを以下に書くと

$$y_i|\lambda_i \sim Poisson(\lambda_i)$$
$$\lambda_i = g^{-1}(u_i) = \exp(u_i) = \exp(f(x_i))$$
$$f(x_i) \sim GP(0, k(x, x'))$$

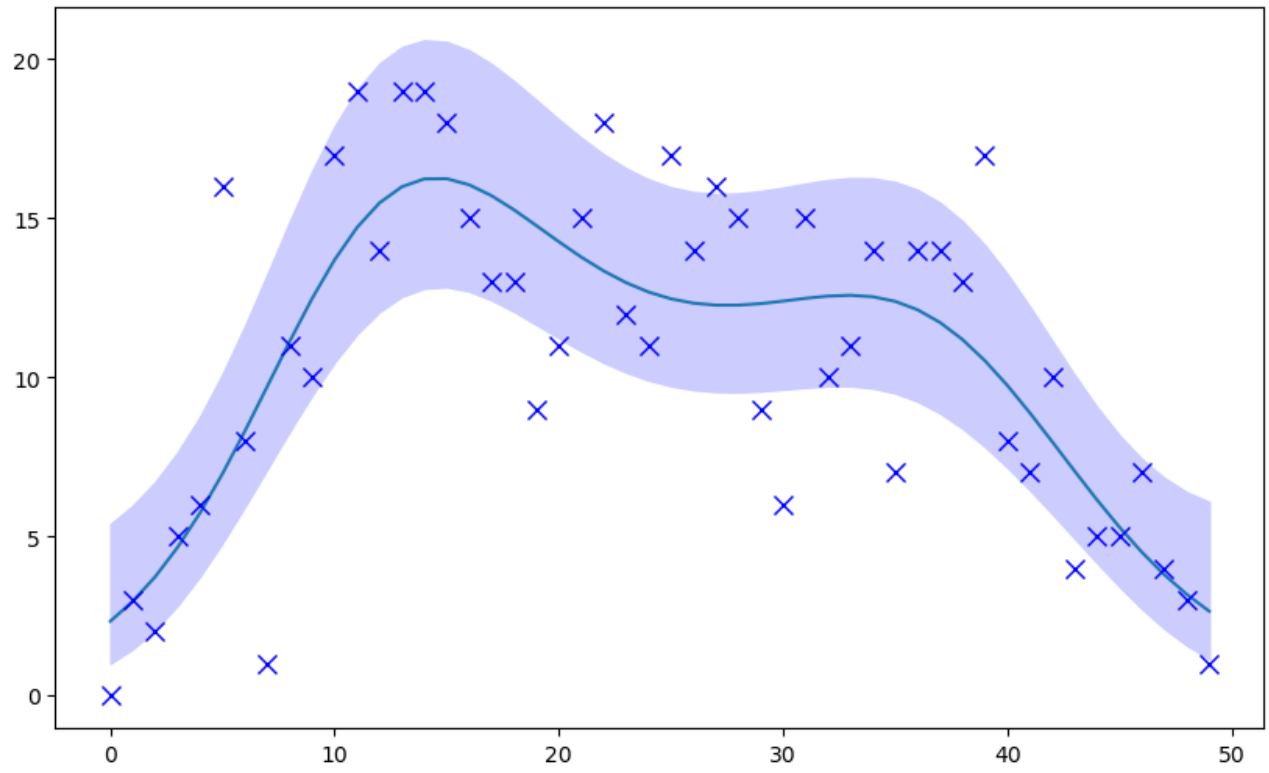
入力 [11]:

```
X = np.arange(0, 50).reshape(-1, 1)
Y = np.array([0,3,2,5,6,16,8,1,11,10,17,19,14,19,19,18,15,13,13,9,11,15,18,12\
              ,11,17,14,16,15,9,6,15,10,11,14,7,14,14,13,17,8,7,10,4,5,5,7,4,3,1])
plt.figure(figsize=(10, 6))  # 幅10インチ、高さ6インチの図を作成
# 散布図を作成
plt.scatter(X, Y)

# グラフのタイトルと軸ラベルを設定
plt.title('Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')

# グラフを表示
plt.show()
```



入力 [12]:

```
import GPy
import numpy as np
import matplotlib.pyplot as plt
from numpy import exp, sqrt

def gpr_poisson(X, Y):
    N = len(Y)
    model = GPy.core.GP(X=X, Y=Y[:,None],
                        kernel=GPy.kern.RBF(1),
                        inference_method=GPy.inference.latent_function_inference.Laplace(),
                        likelihood=GPy.likelihoods.Poisson())
    model.optimize()
    mu, var = model._raw_predict(X)
    plt.plot(X, np.exp(mu))
    plt.fill_between(X[:,0], exp(mu[:,0] + 3*sqrt(var[:,0])),
                     exp(mu[:,0] - 3*sqrt(var[:,0])),
                     color='#ccccff')
    plt.plot(X, Y, 'xb', markersize=8)
    plt.show()

X = np.arange(0, 50).reshape(-1, 1)
Y = np.array([0,3,2,5,6,16,8,1,11,10,17,19,14,19,19,18,15,13,13,9,11,15,18,12\
              ,11,17,14,16,15,9,6,15,10,11,14,7,14,14,13,17,8,7,10,4,5,5,7,4,3,1])

plt.figure(figsize=(10, 6))  # 幅10インチ、高さ6インチの図を作成
gpr_poisson(X, Y)
```



Merton-NBDモデル

ガウス過程の説明を終え，以下ではMerton-NBDモデルについての計算を書く．

参考資料

[1] 『Merton model and Poisson process with Log Normal intensity function』

(June17 , 2023 /Masakado Hisakado , Kodai Hattori , Shintaro Mori)

Mertonモデル

y_t:経済全体の影響，相関が債務者に影響を与える．

ここで確率変数列$\vec{y}_T = (y_1, \ldots, y_T)^\top$ が以下のガウス過程に従うとする．

$$\vec{y}_T \sim \mathcal{GP}(\vec{0}, K)$$

ただし，Kはグラム行列であり，$K_{t,t'} = k(t,t') = \mathbb{E}[y_t\, y_{t'}]$であり，これをカーネルと呼ぶ．

カーネル関数に関しては以下のような関数を用いる．

- ガウスカーネル

$$k(t,t') = \theta_1 \exp\left(-\frac{(t-t')^2}{\theta_2}\right)$$

- 指数カーネル

$$k(t,t') = \theta_1 \exp\left(-\frac{|t-t'|}{\theta_2}\right)$$

債務者iに関連する期間tの資産$\hat{U}_{it}$とかき，資産相関ρ_Aとかくと

$$\hat{U}_{it} = \sqrt{\rho_A} y_t + \sqrt{1-\rho_A}\xi_{it} \quad (1)$$

ここで$\xi_{it} \sim \mathcal{N}(0,1) i.i.d$とモデル化する．

式(1)の定義から

$$Cov(\hat{U}_{it}, \hat{U}_{jt'}) = \rho_A k(t,t') + (1-\rho_A)\delta_{t,t'}\delta_{i,j}$$

と計算ができ，$\hat{U}_{it}$の分布は

$$\hat{U}_{it} = \sqrt{\rho_A}\mathcal{N}(0, k(t,t)) + \sqrt{1-\rho_A}\mathcal{N}(0,1)$$

同時刻におけるカーネルの値が1になるカーネルならば，$\hat{U}_{it}$は標準正規分布に従う．

閾値としてYを設定して定義関数$1_{\hat{U}_{it} \le Y}$とする・

潜在変数$y_t = y$のときのデフォルト確率$G(y)$は，$\hat{U}_{it}$は標準正規分布に従うことから

$$\begin{aligned} G(y) &= Pr(X_{it} = 1 | y_t = y) \\ &= Pr(\hat{U}_{it} \le Y | y_t = y) \\ &= Pr\left(\xi_{it} \le \frac{Y - \sqrt{\rho_A} y}{\sqrt{1-\rho_A}} \middle| y_t = y\right) \end{aligned}$$

となるので、

$$G(y) = \Phi\left(\frac{Y - \sqrt{\rho_A} y}{\sqrt{1-\rho_A}}\right)$$

であり平均デフォルト率(PD)をqとして計算すると

$$\begin{aligned} q &= \mathbb{E}[1_{\hat{U}_{it} \le Y}] \\ &= Pr(\hat{U}_{it} \le Y) \\ &= \Phi(Y) \end{aligned}$$

と計算することができる．

BBDからNBDの極限

デフォルト数の無条件分布を

$$Pr[X_t = k_t] = \int_{-\infty}^{\infty} dy \phi(y) BBD\left(k_t | N, \frac{\theta}{\omega}, \frac{m-\theta}{\omega}\right)$$

とモデル化する．

ただし

$$\frac{\frac{\theta}{\omega}}{\frac{\theta+m-\theta}{\omega}} = \frac{\theta}{m} = G(y)$$
$$\frac{1}{\frac{m}{\omega}+1} = \frac{\omega}{\omega+m} = \rho$$

とおく．この尤度の期待値は

$$\mathbb{E}\left[BBD\left(k|N, \frac{\theta}{\omega}, \frac{m-\theta}{\omega}\right)\right] = N\frac{\theta}{m} = NG(y)$$

ここでBBDからNBDの極限を確認する．

$$\begin{aligned}
BBD\left(k|N, \frac{\theta}{\omega}, \frac{m-\theta}{\omega}\right) &= \frac{N!}{k!(N-k)!}\frac{Beta(\frac{\theta}{\omega}+k, \frac{m-\theta}{\omega}+N-k)}{Beta(\frac{\theta}{\omega}, \frac{m-\theta}{\omega})} \\
&= \frac{N!}{k!(N-k)!}\frac{\Gamma(\frac{\theta}{\omega}+k)\Gamma(\frac{m-\theta}{\omega}+N-k)}{\Gamma(\frac{\theta}{\omega}+\frac{m-\theta}{\omega}+N)}\frac{\Gamma(\frac{\theta}{\omega}+\frac{m-\theta}{\omega})}{\Gamma(\frac{\theta}{\omega})\Gamma(\frac{m-\theta}{\omega})} \\
&= \frac{\Gamma(k+\frac{\theta}{\omega})}{\Gamma(k+1)\Gamma(\frac{\theta}{\omega})}\frac{\Gamma(N+1)}{\Gamma(N-k+1)}\frac{\Gamma(N-k+\frac{m-\theta}{\omega})\Gamma(\frac{m}{\omega})}{\Gamma(N+\frac{m}{\omega})\Gamma(\frac{m-\theta}{\omega})} \\
&= \binom{\frac{\theta}{\omega}+k-1}{k} N(N-1)\cdots(N-k+1)\frac{(\frac{m}{\omega}-1)\cdots(\frac{m}{\omega}-\frac{\theta}{\omega}}{(N+\frac{m}{\omega}-1)\cdots(N+\frac{m}{\omega}}
\end{aligned}$$

この式変形につづき極限操作を行う．$N, m \to \infty$，$\frac{N}{m} = \triangle$と飛ばすと

$$\begin{aligned}
BBD\left(k|N, \frac{\theta}{\omega}, \frac{m-\theta}{\omega}\right) &\simeq \binom{\frac{\theta}{\omega}+k-1}{k} N^k \frac{(\frac{m}{\omega})^{\theta/\omega}}{(N+\frac{m}{\omega})^{\theta/\omega}} \\
&= \binom{\frac{\theta}{\omega}+k-1}{k}\left(\frac{N}{N+\frac{m}{\omega}}\right)^k\left(\frac{\frac{m}{\omega}}{N+\frac{m}{\omega}}\right)^{\theta/\omega} \\
&= \binom{\frac{\theta}{\omega}+k-1}{k}\left(\frac{\omega\triangle}{1+\omega\triangle}\right)^k\left(\frac{1}{1+\omega\triangle}\right)^{\theta/\omega} \\
&= NBD\left(k|\frac{\theta}{\omega}, \frac{1}{1+\omega\triangle}\right)
\end{aligned}$$

以降$\triangle = 1$と考える．

この分布の期待値は

$$\mathbb{E}\left[NBD\left(k|\frac{\theta}{\omega}, \frac{1}{1+\omega}\right)\right] = \frac{\frac{\theta}{\omega}}{1+\omega} = \theta$$

であるから極限操作前の期待値と一致する．

この極限操作は参考文献[1]と同じ操作をしている．

なぜなら ,参考論文[1]では$G(y), q \to 0, N \to \infty$としているので，

今回の場合も$m \to \infty$としたことで$G(y) = \frac{\theta}{m} \to 0$，$N \to \infty$ も同様，$q \to 0$も$m \to \infty$により閾値Yを小さくしているので同様に$q \to 0$となり実現している．

$G(y)$も参考文献[1]に従ってシグモイド近似を行う．このとき尤度によらず強度関数が対数正規分布となることも参考文献[1]を参考に示す．

$$G(y) = \Phi\left(\frac{\Phi^{-1}(q) - \sqrt{\rho_A}y}{\sqrt{1-\rho_A}}\right)$$

標準正規分布の分布関数をシグモイド関数で近似する．このとき$\beta = 1.596$として$\Phi(x) \simeq \sigma(\beta x)$とするとよく近似される．

シグモイドの逆関数は対数オッズであるから$\Phi^{-1}(q) \simeq \frac{1}{\beta}\log\frac{q}{1-q}$とできる．

$$\begin{aligned} G(y) &\simeq \frac{1}{1+\exp\left(-\frac{1}{\sqrt{1-\rho_A}}\log\frac{q}{1-q} + \sqrt{\frac{\rho_A}{1-\rho_A}}\beta y\right)} \\ &= \frac{1}{1+\left(\frac{q}{1-q}\right)^{\frac{1}{\sqrt{1-\rho_A}}}\exp\left(\sqrt{\frac{\rho_A}{1-\rho_A}}\beta y\right)} \end{aligned}$$

qが十分に小さいから$\frac{1-q}{q} \simeq \frac{1}{q}$とさらに近似をして

$$G(y) \simeq q^{\sqrt{\frac{\rho_A}{1-\rho_A}}}\exp\left(-\sqrt{\frac{\rho_A}{1-\rho_A}}\beta y\right)$$

$N \to \infty, m \to \infty$により$G(y) \to 0, q \to 0$という極限から$\lambda_0 \equiv Nq^{\frac{1}{\sqrt{1-\rho_A}}}$を固定して強度は

$$\lambda(y_t) = \lambda_0 \exp\left(-\sqrt{\frac{\rho_A}{1-\rho_A}}\beta y_t\right)$$

ここでy_tは標準正規分布に従っているため，強度関数は$\mathcal{LN}(\log\lambda_0, \frac{\rho_A}{1-\rho_A}\beta^2)$となることがわかる．