Regression with Gaussian Processes

Exercise Lecture

Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Why Gaussian Processes?

Gaussian process regressors are **powerful** and **flexible** methods that exploit the correlation between input space points

Provide us **not only the mean**, but also the **uncertainty** of the estimation

Gaussian Process

A Gaussian process is completely specified by its **mean** and **covariance function**

$$\boldsymbol{f}(\boldsymbol{x}) \sim \boldsymbol{GP}(\boldsymbol{m}(\boldsymbol{x}), \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}'))$$

Gaussian Process

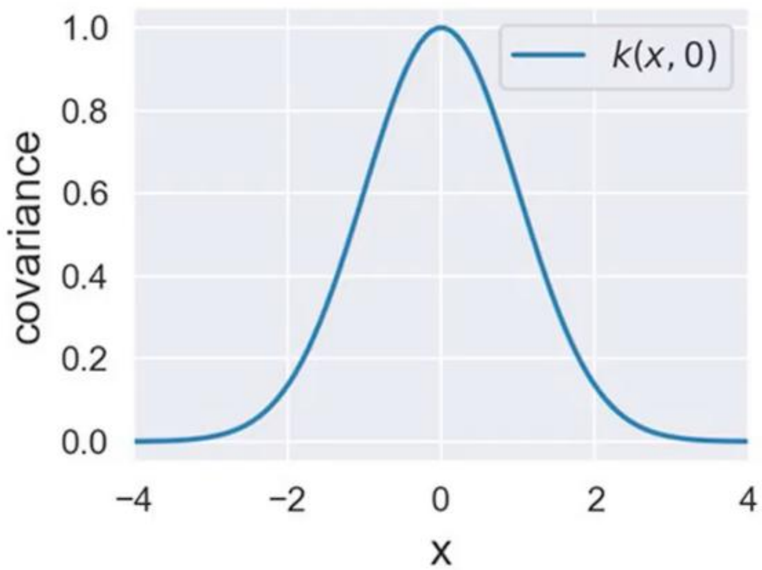
If we have not any prior information about the mean $\boldsymbol{m}(\boldsymbol{x})$, we will take it to be zero

The covariance is given by the kernel function :

$$E[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)] = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$$

Kernel function

The kernel function provides a measure of similarity between two points.



Kernel function

The kernel function needs to be **positive-definite** and **symmetric**

A common choice is the **squared exponential kernel**:

$$k(x, x') = \theta^2 e^{-\frac{(x - x')^2}{2l^2}}$$

θ^2 is a **scale factor**

l is the **lengthscale** (controls the "wiggliness" of the function)

GP Prediction

For a single test point $\boldsymbol{x}_*$ *, the mean and variance prediction is given by:*

$$\mu_* = k_*^T \left[K_N + \sigma_n^2 I \right]^{-1} y$$

$$\sigma_*^2 = k(x_*, x_*) - k_{*N}^T \left[K_N + \sigma_n^2 I \right]^{-1} k_*.$$

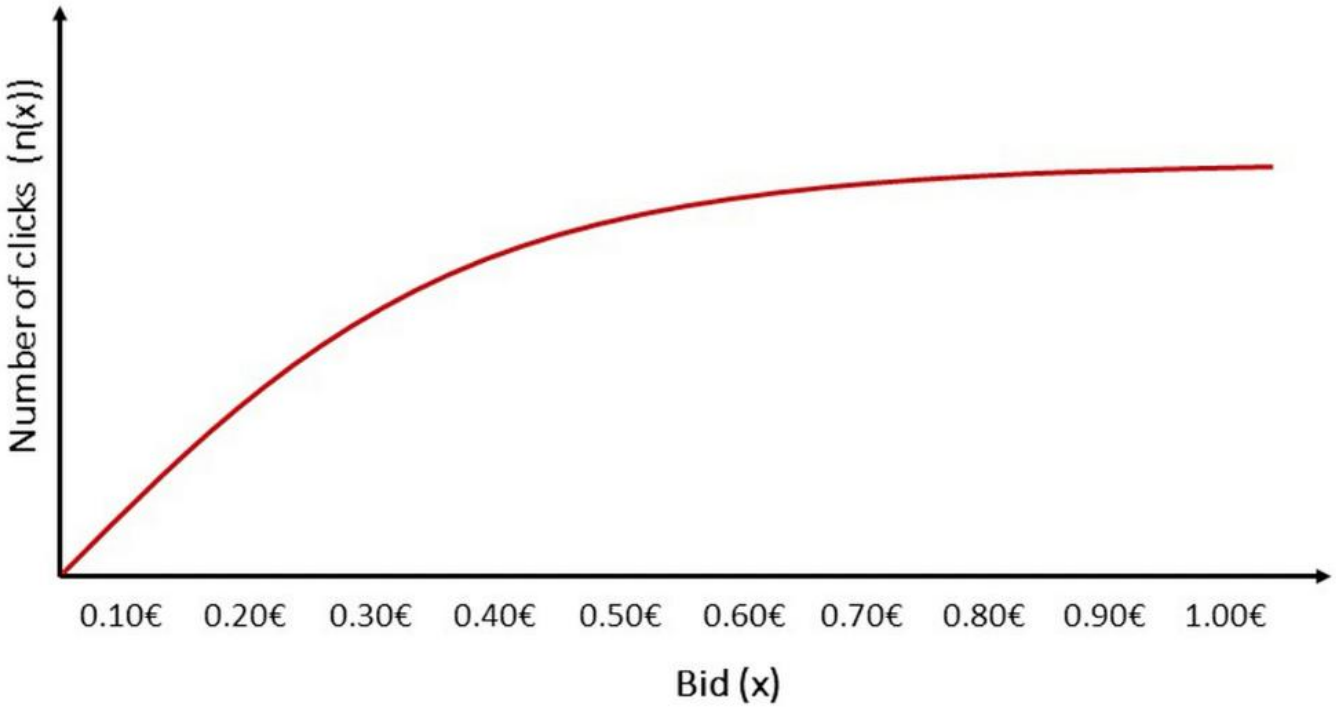
where $\boldsymbol{k}_*$ is the covariance vector between the test point and training points, $\boldsymbol{K}_N$ the covariance matrix, $\boldsymbol{\sigma}_n^2$ the noise variance and $\boldsymbol{y}$ the targets vector

References

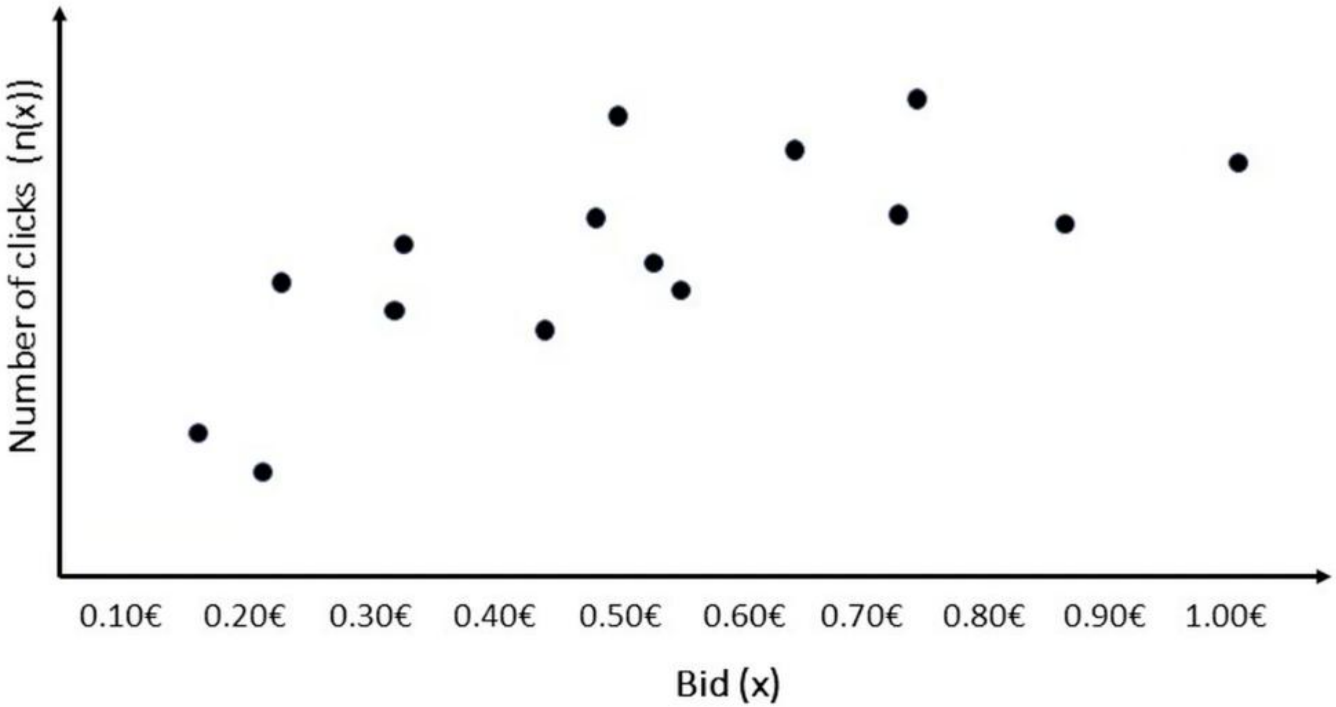
Gaussian Processes for Machine Learning
Rasmussen and Williams (2006)
http://www.gaussianprocess.org/gpml/chapters/

Gaussian Processes with Python

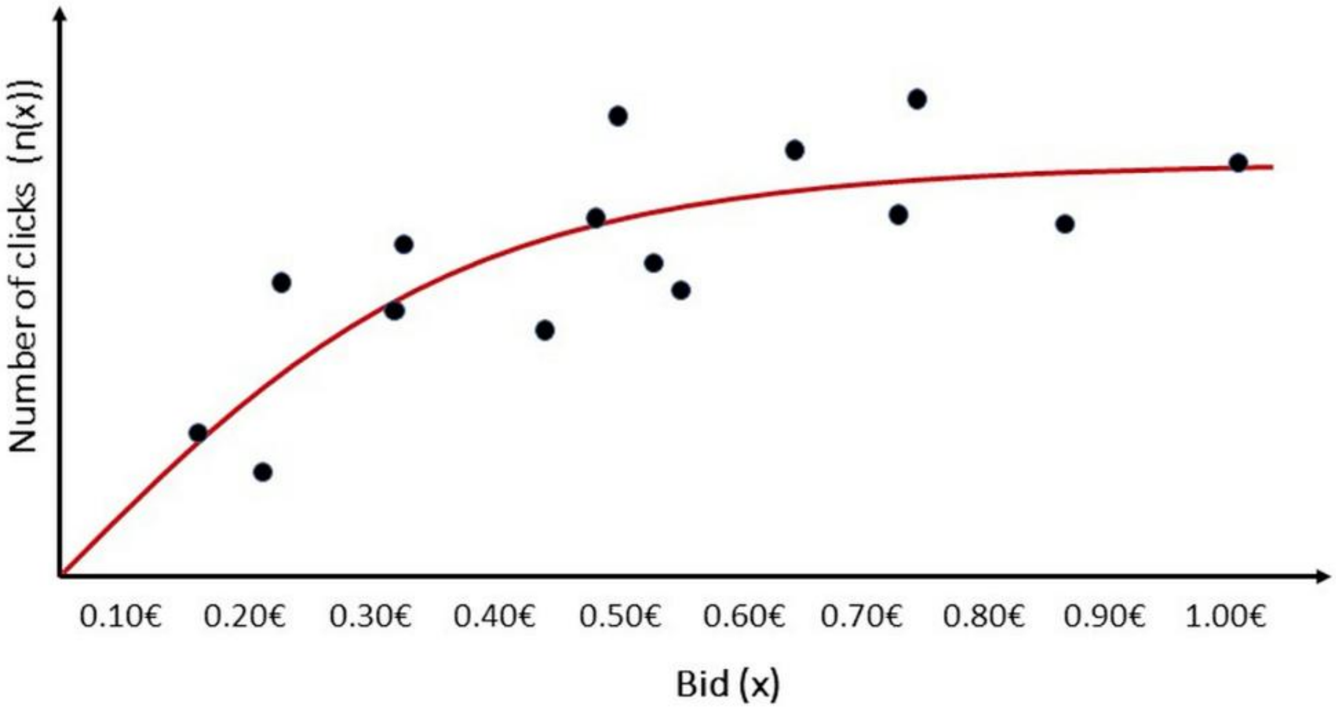
Example: Clicks Estimation



Example: Clicks Estimation



Example: Clicks Estimation



```
(1.0 - np.exp(-5.0*x)) * 100
```

Observations Generation

Define a set of possible bids $X = \{0.10, 0.15, \dots, 1.00\}$

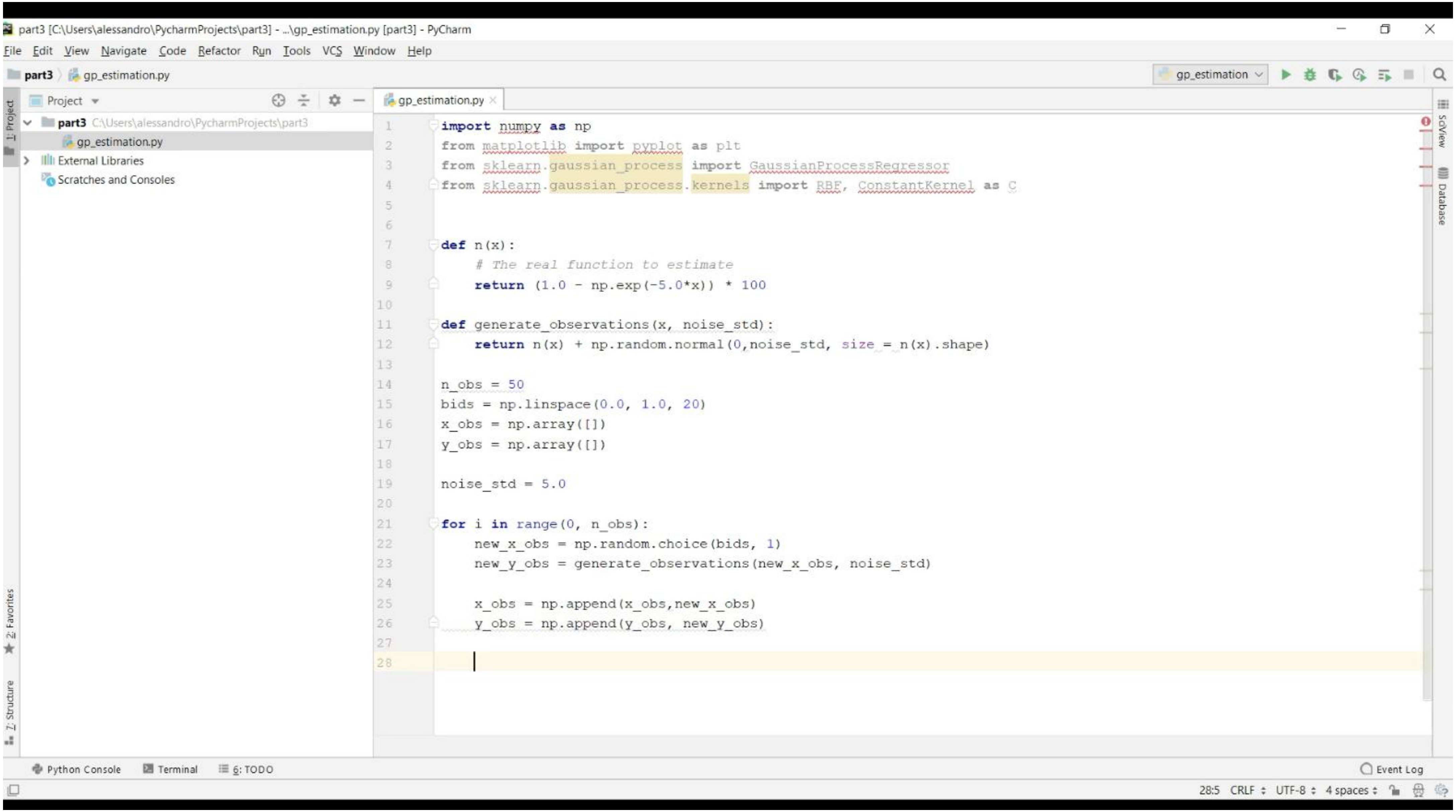
Define function $n(x)$ generating the number of clicks:

$$n(x) = 100(1 - e^{-5x})$$

Observations Generation

We will generate samples **one by one** to show how the **GP reduces the uncertainty** of its estimation once it collects a new sample

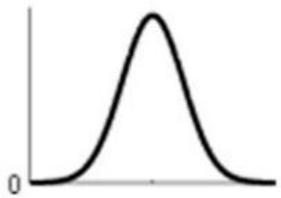
Let’s implement it!


part3 [C:\Users\alessandro\PycharmProjects\part3] - ...\gp_estimation.py [part3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
part3 gp_estimation.py
gp_estimation
Project
part3 C:\Users\alessandro\PycharmProjects\part3
gp_estimation.py
External Libraries
Scratches and Consoles
gp_estimation.py
import numpy as np
from matplotlib import pyplot as plt
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C
def n(x):
# The real function to estimate
return (1.0 - np.exp(-5.0*x)) * 100
def generate_observations(x, noise_std):
return n(x) + np.random.normal(0,noise_std, size = n(x).shape)
n_obs = 50
bids = np.linspace(0.0, 1.0, 20)
x_obs = np.array([])
y_obs = np.array([])
noise_std = 5.0
for i in range(0, n_obs):
new_x_obs = np.random.choice(bids, 1)
new_y_obs = generate_observations(new_x_obs, noise_std)
x_obs = np.append(x_obs,new_x_obs)
y_obs = np.append(y_obs, new_y_obs)
1: Project
2: Favorites
7: Structure
SciView
Database
Python Console
Terminal
6: TODO
Event Log
28:5 CRLF UTF-8 4 spaces

GP estimation

Normalize Data

Specify the kernel function



Squared Exponential

Periodic

Linear

GP estimation

Normalize Data

Specify the kernel function

Set the kernel hyper-parameters

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\theta}^2 \boldsymbol{e}^{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2\boldsymbol{l}^2}}$$

- Lengthscale $\boldsymbol{l} = \mathbf{1}$
- Scale factor (variance) $\boldsymbol{\theta} = \mathbf{1}$

GP estimation

Normalize Data

Specify the kernel function

Set the kernel hyper-parameters

Fit the GP

GP estimation

Normalize Data

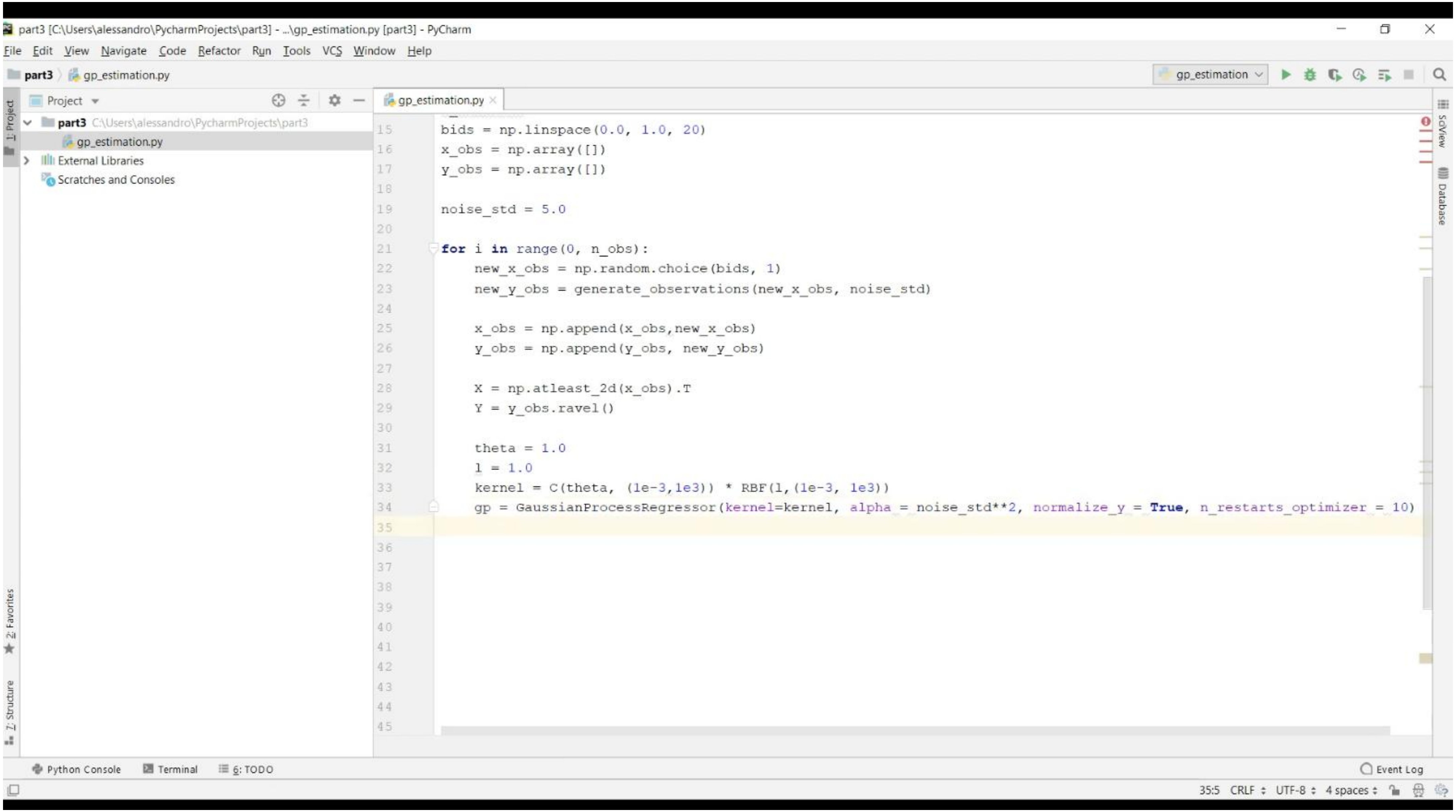
Specify the kernel function

Set the kernel hyper-parameters

Fit the GP

Estimate hyper-parameters from data

Let's implement it!



part3 [C:\Users\alessandro\PycharmProjects\part3] - ...\gp_estimation.py [part3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
part3 gp_estimation.py
gp_estimation
Project
part3 C:\Users\alessandro\PycharmProjects\part3
gp_estimation.py
External Libraries
Scratches and Consoles
gp_estimation.py
bids = np.linspace(0.0, 1.0, 20)
x_obs = np.array([])
y_obs = np.array([])
noise_std = 5.0
for i in range(0, n_obs):
    new_x_obs = np.random.choice(bids, 1)
    new_y_obs = generate_observations(new_x_obs, noise_std)
    x_obs = np.append(x_obs,new_x_obs)
    y_obs = np.append(y_obs, new_y_obs)
    X = np.atleast_2d(x_obs).T
    Y = y_obs.ravel()
    theta = 1.0
    l = 1.0
    kernel = C(theta, (1e-3,1e3)) * RBF(l,(1e-3, 1e3))
    gp = GaussianProcessRegressor(kernel=kernel, alpha = noise_std**2, normalize_y = True, n_restarts_optimizer = 10)
Python Console
Terminal
6: TODO
Event Log
35:5 CRLF UTF-8 4 spaces

part3 [C:\Users\alessandro\PycharmProjects\part3] - ...\gp_estimation.py [part3] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

part3 〉 gp_estimation.py

gp_estimation.py

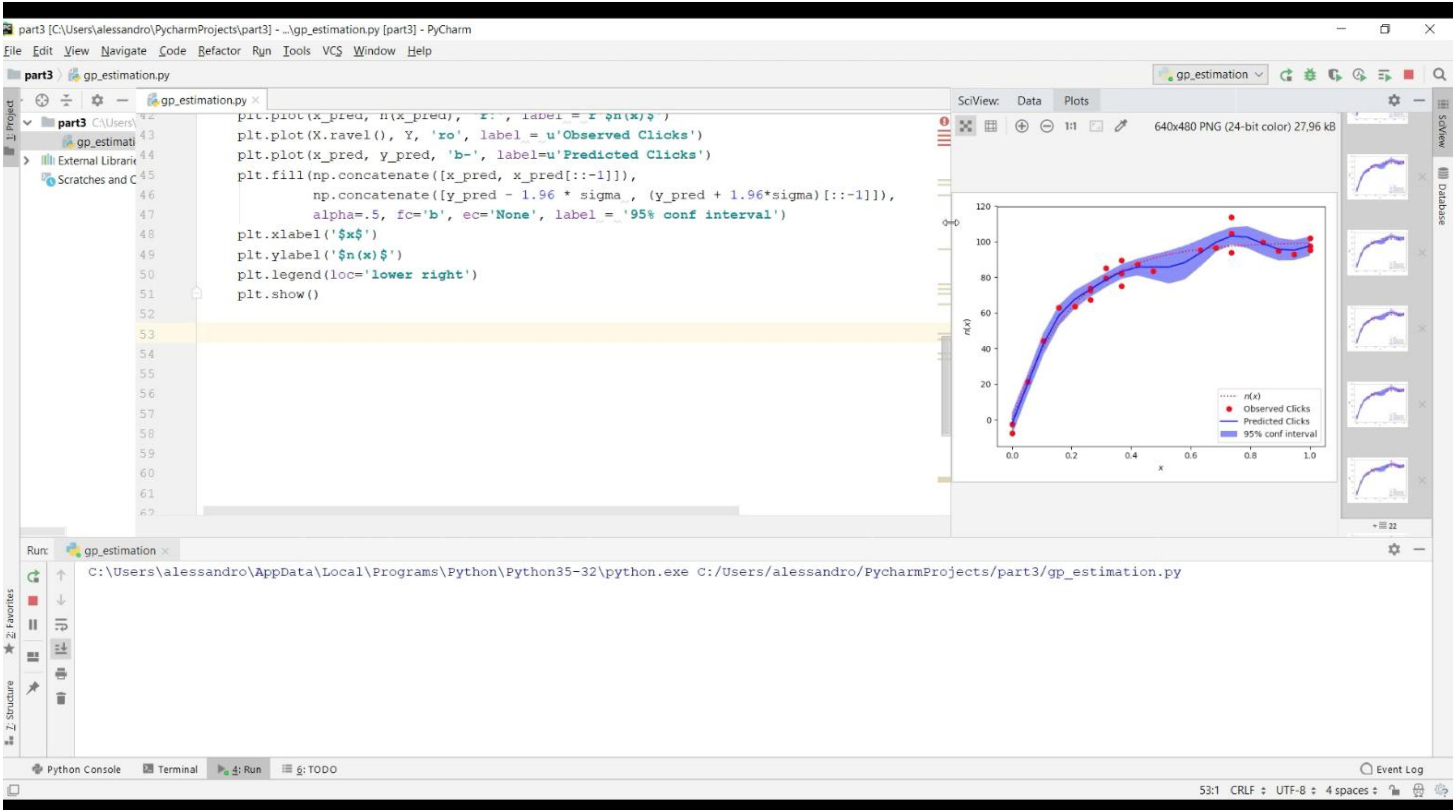
```python
    y_obs = np.append(y_obs, new_y_obs)

    X = np.atleast_2d(x_obs).T
    Y = y_obs.ravel()

    theta = 1.0
    l = 1.0
    kernel = C(theta, (1e-3,1e3)) * RBF(l,(1e-3, 1e3))
    gp = GaussianProcessRegressor(kernel=kernel, alpha = noise_std**2, normalize_y = True, n_restarts_optimizer = 10)

    gp.fit(X,Y)

    x_pred = np.atleast_2d(bids).T
    y_pred, sigma = gp.predict(x_pred, return_std=True)

    plt.figure(i)
    plt.plot(x_pred, n(x_pred), 'r:', label = r'$n(x)$')
    plt.plot(X.ravel(), Y, 'ro', label = u'Observed Clicks')
    plt.plot(x_pred, y_pred, 'b-', label=u'Predicted Clicks')
    plt.fill(np.concatenate([x_pred, x_pred[::-1]]),
             np.concatenate([y_pred - 1.96 * sigma , (y_pred + 1.96*sigma)[::-1]]),
             alpha=.5, fc='b', ec='None', label = '95% conf interval')
    plt.xlabel('$x$')
    plt.ylabel('$n(x)$')
    plt.legend(loc='lower right')
    plt.show()
```

for i in range(0, n_obs)

Python Console   Terminal   6: TODO   Event Log

51:15 CRLF UTF-8 4 spaces


part3 [C:\Users\alessandro\PycharmProjects\part3] - ...\gp_estimation.py [part3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
part3 gp_estimation.py
gp_estimation
gp_estimation.py
part3 C:\Users\
External Librarie
Scratches and C
plt.plot(X.ravel(), Y, 'ro', label = u'Observed Clicks')
plt.plot(x_pred, y_pred, 'b-', label=u'Predicted Clicks')
plt.fill(np.concatenate([x_pred, x_pred[::-1]]),
np.concatenate([y_pred - 1.96 * sigma , (y_pred + 1.96*sigma)[::-1]]),
alpha=.5, fc='b', ec='None', label = '95% conf interval')
plt.xlabel('x')
plt.ylabel('$n(x)$')
plt.legend(loc='lower right')
plt.show()
SciView: Data Plots
640x480 PNG (24-bit color) 27,98 kB
n(x)
Observed Clicks
Predicted Clicks
95% conf interval
x
Run: gp_estimation
C:\Users\alessandro\AppData\Local\Programs\Python\Python35-32\python.exe C:/Users/alessandro/PycharmProjects/part3/gp_estimation.py
Python Console Terminal 4: Run 6: TODO
Event Log
53:1 CRLF UTF-8 4 spaces


part3 [C:\Users\alessandro\PycharmProjects\part3] - ...\gp_estimation.py [part3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
part3 gp_estimation.py
gp_estimation
gp_estimation.py
SciView: Data Plots
640x480 PNG (24-bit color) 27,96 kB
part3 C:\Users\
gp_estimati
External Librarie
Scratches and C
plt.plot(X.ravel(), Y, 'ro', label = u'Observed Clicks')
plt.plot(x_pred, y_pred, 'b-', label=u'Predicted Clicks')
plt.fill(np.concatenate([x_pred, x_pred[::-1]]),
np.concatenate([y_pred - 1.96 * sigma , (y_pred + 1.96*sigma)[::-1]]),
alpha=.5, fc='b', ec='None', label = '95% conf interval')
plt.xlabel('x')
plt.ylabel('$n(x)$')
plt.legend(loc='lower right')
plt.show()
n(x)
Observed Clicks
Predicted Clicks
95% conf interval
Run: gp_estimation
C:\Users\alessandro\AppData\Local\Programs\Python\Python35-32\python.exe C:/Users/alessandro/PycharmProjects/part3/gp_estimation.py
Python Console
Terminal
4: Run
6: TODO
Event Log
53:1 CRLF UTF-8 4 spaces