Kernel density estimation

Explanation and example

Abdelwahid Benslimane

wahid.benslimane@gmail.com

The distribution of data in the vector space to which they belong can be effectively described by a probability density function. This characterization of the data distribution has multiple benefits, such as gaining a better understanding of the data, detecting outliers, and being able to construct classification models based on the probability densities of different classes.

One possible approach is to use parametric methods that make assumptions about the membership of the density function to a specific set. The goal is then to estimate the parameters of the best candidate function $\hat{f}$ within this set.

Among the parametric methods, one notable example is the kernel density estimation method (KDE). This method involves selecting a kernel ϕ and a smoothing parameter h, then centering the kernel on each observation $x_i \in \mathbb{R}^d$ of the sample $\mathcal{D}_N$ whose distribution is being estimated. The density at a point x is then estimated by:

$$\hat{f}_h(x) = \frac{1}{N}\sum_{i=1}^{N} \phi_h(x, x_i).$$

In general, the kernels $\phi_h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ that are used are obtained from one-dimensional kernels $\phi : \mathbb{R} \to \mathbb{R}^+$ by $\phi_h(x, y) = \frac{1}{h}\phi(\frac{||x-y||}{h})$, where $||.||$ is the norm of the difference of the two vectors arguments of the function ϕ_h.

For the estimate thus obtained to be itself a density function, it is sufficient to select as kernel function a probability density function, as:

- values are all (non strictly) positive: $\phi(u) \geq 0, \forall u \in \mathbb{R}$;
- the normalization condition is satisfied: $\int_{\mathbb{R}} \phi(u) du = 1$.

If the choice of the kernel function has little impact on the quality of the estimation (provided that the kernel is smooth, as for example the Gaussian kernel or the logistic kernel above), the smoothing parameter has a very significant impact on the results. Indeed, a too low value for h produces meaningless results, as the density is non-zero only in the immediate neighbourhood of the observations, while a too high value produces excessive smoothing with loss of details.

The difference between the real (unknown) density function f and an estimate $\hat{f}_h$ is usually measured by the mean integrated square error ($MISE$):

$$MISE = E[\int (\hat{f}_h(x) - f(x))^2 dx].$$

The optimal value of h is the one that minimizes the $MISE$. This value will generally depend on the number of observations. The higher it is, the more we can reduce h to refine the estimation locally without risking to lose the regularity brought by the smoothing.

By expanding the expression under the integral, we obtain:

$$MISE = E[\int \hat{f}_h^2(x)dx - 2\int \hat{f}_h(x)f(x)dx + \int f^2(x)dx].$$

As $E[\int f^2(x)dx] = \int f^2(x)dx$ does not depend on h, we can write:

$$h^* = \underset{h>0}{\text{argmin}} E[\int \hat{f}_h^2(x)dx - 2\int \hat{f}_h(x)f(x)dx].$$

Also, $E[2\int \hat{f}_h(x)f(x)dx] = 2E[\hat{f}_h(X)]$ where X is a random variable of density f.

Let us introduce $\hat{f}_{h,-i}$ the estimate obtained by excluding the observation x_i from $\mathcal{D}_N$:

$$\hat{f}_{h,-i} = \frac{1}{N-1}\sum_{j\neq i}\phi_h(x, x_j).$$

Under certain conditions concerning the density f and the kernel ϕ we can show that $\int \hat{f}^2(x)dx$ is an unbiased estimator of $E[\int \hat{f}^2(x)dx]$ and $\frac{1}{N}\sum_{i=1}^{N}\hat{f}_{h,-i}(x_i)$ an unbiased estimator of $E[\hat{f}_h(X)]$.

The expression for $\int \hat{f}^2(x)dx$ can be obtained analytically for some kernels ϕ (such as the Gaussian kernel).

The optimal value of the smoothing constant (or window width) will then be:

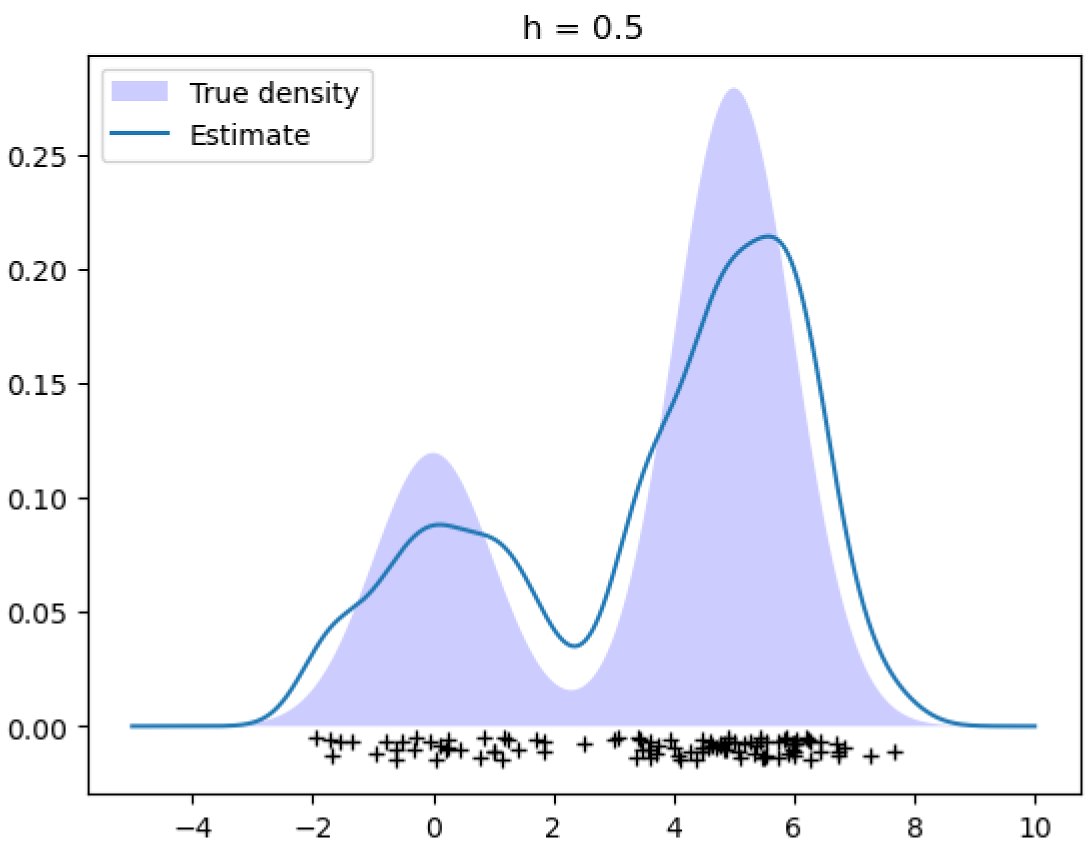
$$h^* = \underset{h>0}{\operatorname{argmin}}[\int \hat{f}^2(x)dx - \frac{2}{N}\sum_{i=1}^{N}\hat{f}_{h,-i}(x_i)].$$

Example using Python

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

from scipy.stats import norm
from sklearn.neighbors import KernelDensity

# we generate the sample from 2 Gaussian laws.
N = 100
X = np.concatenate((np.random.normal(0, 1, int(0.3 * N)),
                    np.random.normal(5, 1, int(0.7 * N))))[:, np.newaxis]

#we prepare the points where we will calculate the density
X_plot = np.linspace(-5, 10, 1000)[:, np.newaxis]

#we start with the true density
#the weights of the laws in the sum are the weights of the laws
#of the sample X generated above
true_density = (0.3 * norm(0,1).pdf(X_plot[:,0]) + 0.7 * norm(5,1).pdf(X_plot[:,0]))
```

The variable true_density corresponds to the real distribution of our data. In normal circumstances, it is unknown. We will try to estimate it using KDE.

In [5]:
```
#the estimate of the density will be made with Gaussian kernels
#bandwidth is the smoothing parameter
kdensity = KernelDensity(kernel='gaussian', bandwidth=0.5).fit(X)

#score_samples returns the log of the estimated probability density,
#thus we need to use the exp fuction to get the probability density
density = np.exp(kdensity.score_samples(X_plot))

# we display the real density and its estimate
fig = plt.figure()
ax = fig.add_subplot(111)
ax.fill(X_plot[:,0], true_density, fc='b', alpha=0.2, label='True density')
ax.plot(X_plot[:,0], density, '-', label="Estimate")
ax.plot(X[:, 0], -0.005 - 0.01 * np.random.random(X.shape[0]), '+k')
ax.legend(loc="upper left")
plt.title("h = 0.5")
plt.show()
```

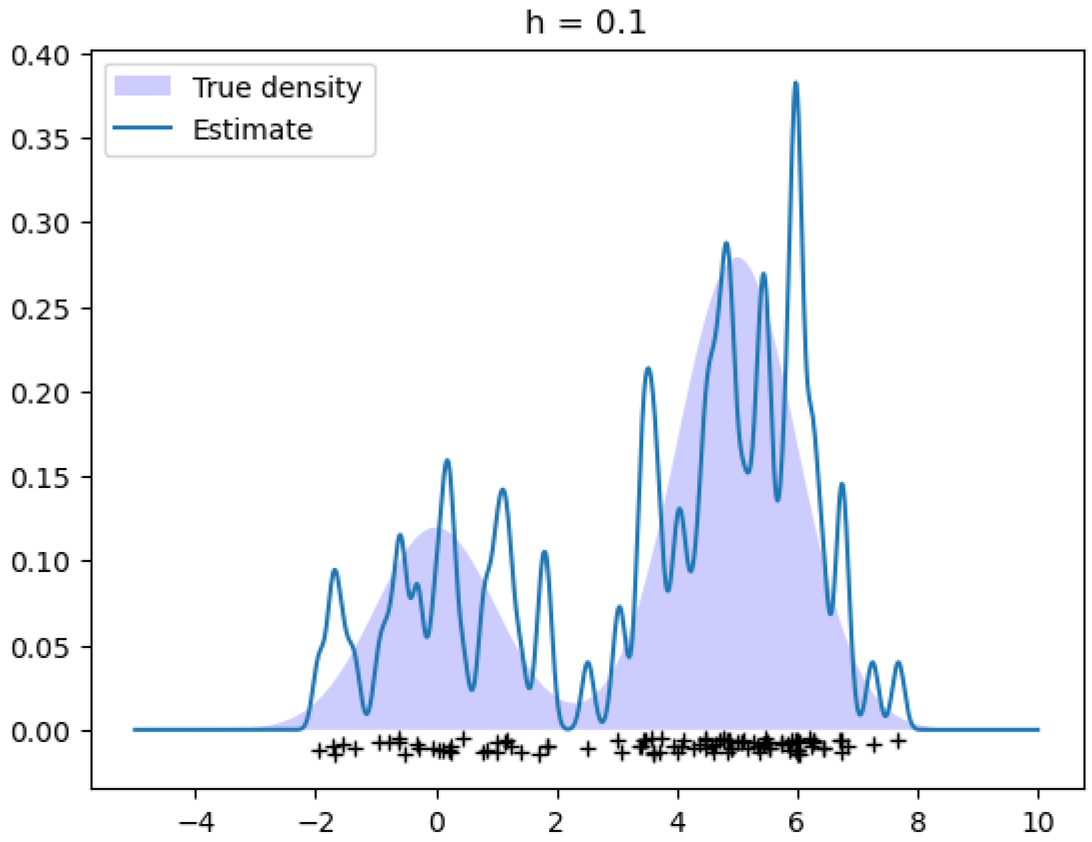


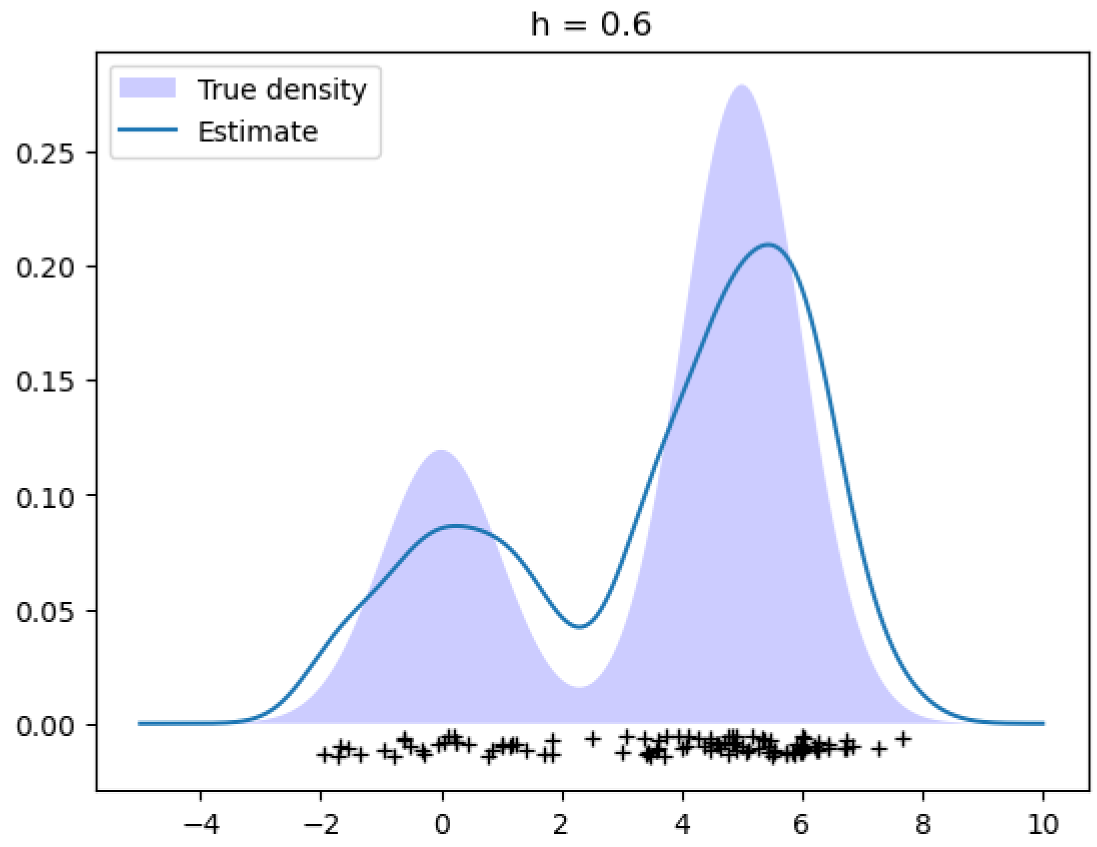
Impact of the smoothing parameter

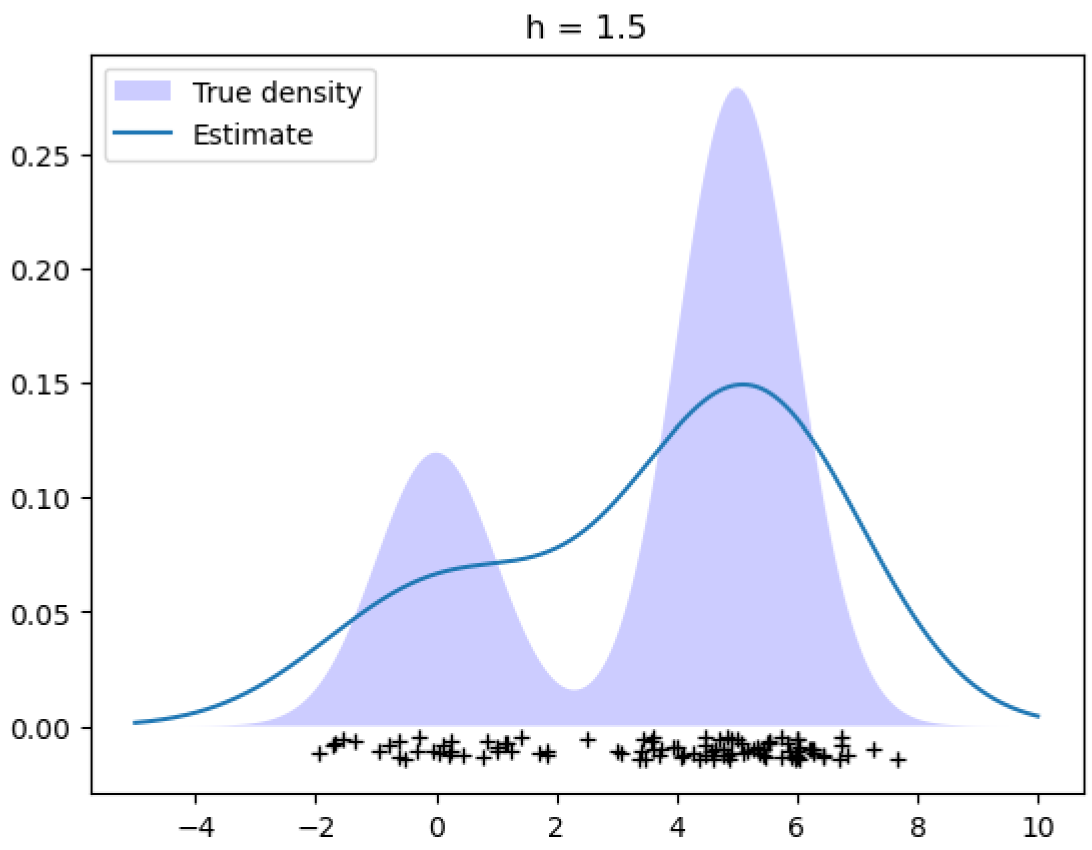
Let's vary the smoothing parameters and see the results

In [6]:
```
#h=0.1
kdensity = KernelDensity(kernel='gaussian', bandwidth=0.1).fit(X)
density = np.exp(kdensity.score_samples(X_plot))

fig = plt.figure()
ax = fig.add_subplot(111)
```

```
ax.fill(X_plot[:,0], true_density, fc='b', alpha=0.2, label='True density')
ax.plot(X_plot[:,0], density, '-', label="Estimate")
ax.plot(X[:, 0], -0.005 - 0.01 * np.random.random(X.shape[0]), '+k')
ax.legend(loc="upper left")
plt.title("h = 0.1")
plt.show()

#h=0.6
kdensity = KernelDensity(kernel='gaussian', bandwidth=0.60).fit(X)
density = np.exp(kdensity.score_samples(X_plot))

fig = plt.figure()
ax = fig.add_subplot(111)
ax.fill(X_plot[:,0], true_density, fc='b', alpha=0.2, label='True density')
ax.plot(X_plot[:,0], density, '-', label="Estimate")
ax.plot(X[:, 0], -0.005 - 0.01 * np.random.random(X.shape[0]), '+k')
ax.legend(loc="upper left")
plt.title("h = 0.6")
plt.show()

#h=1.5
kdensity = KernelDensity(kernel='gaussian', bandwidth=1.5).fit(X)
density = np.exp(kdensity.score_samples(X_plot))

fig = plt.figure()
ax = fig.add_subplot(111)
ax.fill(X_plot[:,0], true_density, fc='b', alpha=0.2, label='True density')
ax.plot(X_plot[:,0], density, '-', label="Estimate")
ax.plot(X[:, 0], -0.005 - 0.01 * np.random.random(X.shape[0]), '+k')
ax.legend(loc="upper left")
plt.title("h = 1.5")
plt.show()
```


h = 0.1
True density
Estimate
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
6
8
10


h = 0.6
True density
Estimate
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
6
8
10



Conclusion

For the same number of examples, a too low value for the smoothing parameter (bandwidth parameter in the KernelDensity method) can increase the number of peaks in the estimate, especially if the sample is small. A too high value for bandwidth results in excessive smoothing.