LA03_Ex2_KDE

April 26, 2018

1 Hochschule Bonn-Rhein-Sieg

2 Learning and Adaptivity, SS18

3 Assignment 03 (24-April-2018)

3.1 Sathiya Ramesh, Pradheep Krishna Muthukrishnan Padmanabhan, Naresh Kumar Gurulingan

4 Task1

4.1 Compare the outcomes of different implementations of KDEs.

There are several options available for computing KDE in Python. - SciPy: gaussian_kde. - Statsmodels: KDEUnivariate and KDEMultivariate. - Scikit-learn: KernelDensity.

4.2 1). Generate synthethic data and plot them

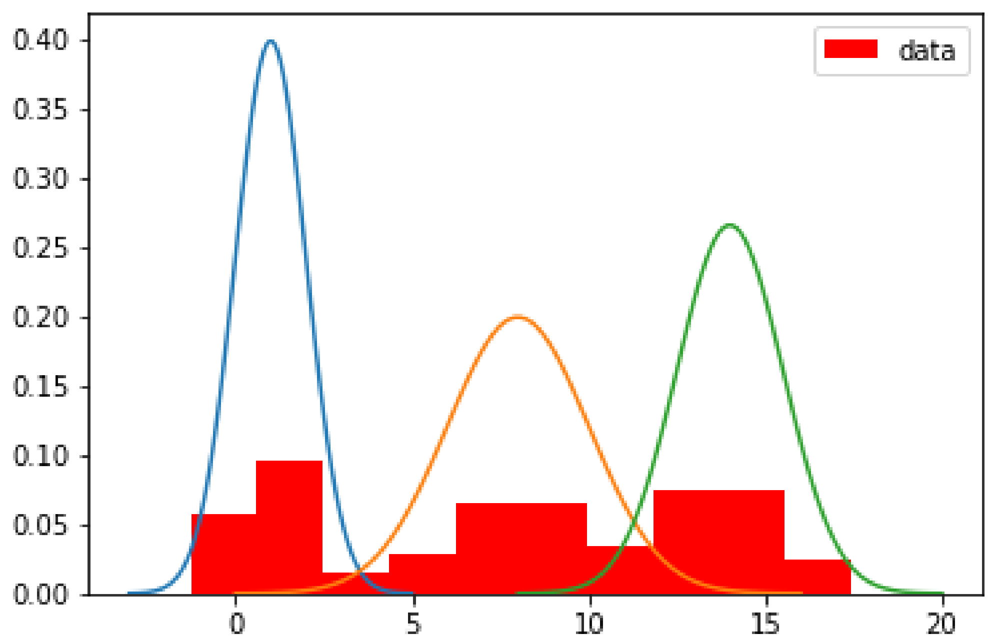
Generate synthetic dataset the distribution of which can be presented as a combination of three Gausian distributions with the following parameters: μ_1=1, σ_1=1 and μ_2=8, σ_2=2 and μ_2=14, σ_2=1.5. Generate 1000 samples from the distribution. Plot the pdf of this distribution and the generated samples. 3) Use the generated samples to perform - (i) KDE with Scipy, - (ii) Univariate KDE with Statsmodels, - (iii) Multivariate KDE with Statsmodels as well as - (iv) KDE with Scikit-learn. 4) Plot all four distributions on one figure.

```
In [1]: import numpy as np
        from matplotlib import mlab
        import matplotlib.pyplot as plt
        from scipy import stats
        from statsmodels.nonparametric.kde import KDEUnivariate
        from statsmodels.nonparametric.kernel_density import KDEMultivariate
        from sklearn.neighbors import KernelDensity
        from __future__ import print_function
```

```
In [2]: def plot_synthetic_data(mu, sigma):
            x = np.linspace(mu - 4*sigma, mu + 4*sigma, 100)
            plt.plot(x, mlab.normpdf(x, mu, sigma))
```

4.2.1 Histogram plot of the samples generated from the combined distribution and an illustration of the three original gaussians combined:

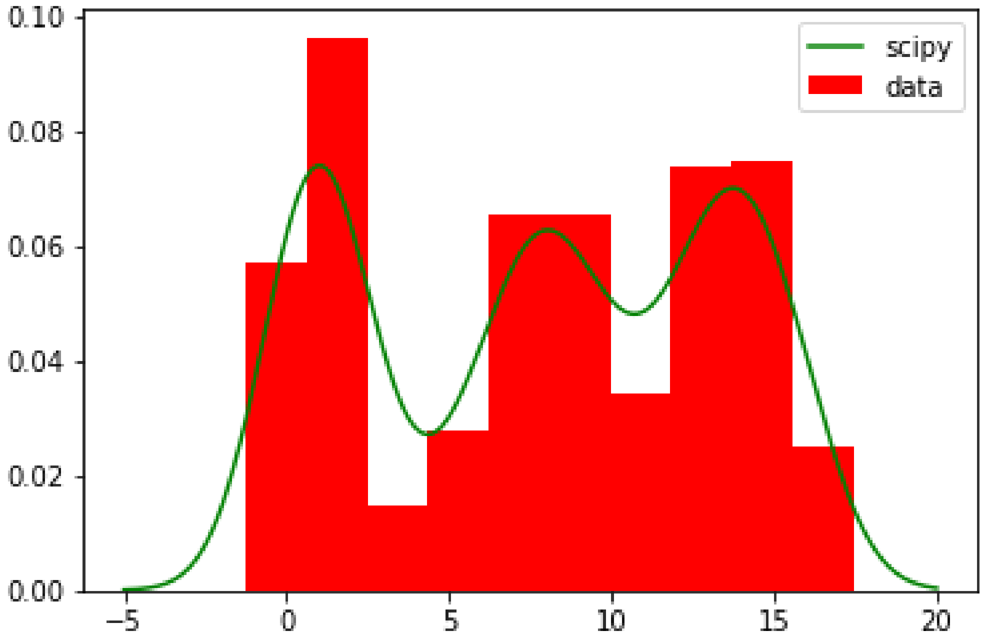
```
In [3]: mean = [1, 8, 14]
        sigma = [1, 2, 1.5]
        np.random.seed(0)

        gaussian_combination = list()
        for mu, sig in zip(mean, sigma):
            plot_synthetic_data(mu, sig)
            gaussian_combination = np.concatenate((gaussian_combination,
                                                   np.random.normal(mu, sig, 1000)))

        dataset = np.random.choice(gaussian_combination, 1000)
        plt.hist(dataset, normed=1, color= 'r', label= 'data')
        plt.legend()
        plt.show()
```



4.2.2 3) (i) KDE with Scipy:

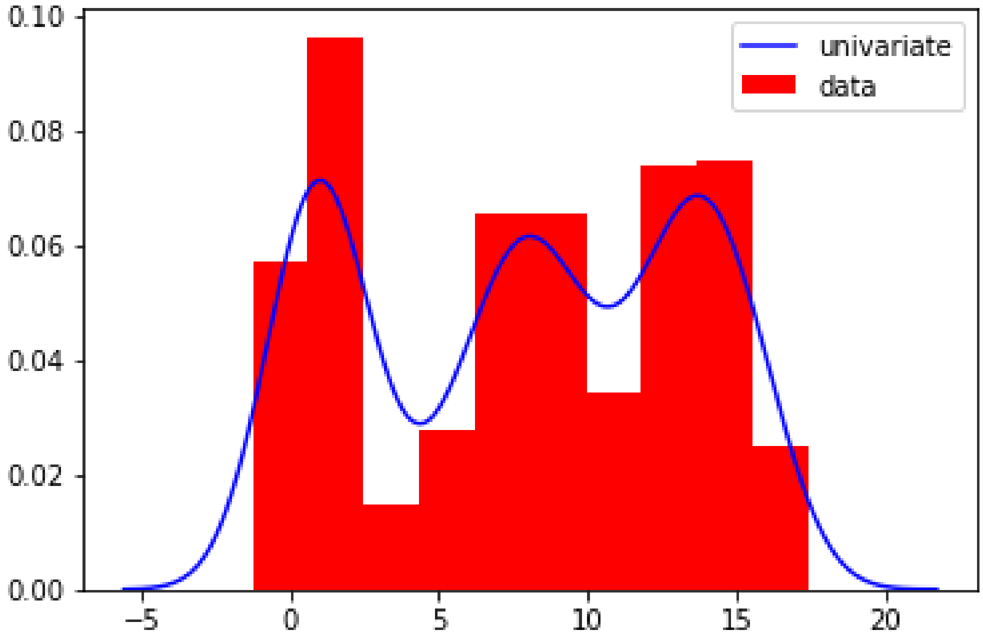
```
In [4]: X_plot = np.linspace(-5, 20, 1000)
        kde_scipy = stats.gaussian_kde(dataset)
        pdf = kde_scipy.evaluate(X_plot)
        plt.hist(dataset, normed=1, color= 'r', label= 'data')
        plt.plot(X_plot, pdf, color= 'g', label= 'scipy')
```

```
plt.legend()
plt.show()
```



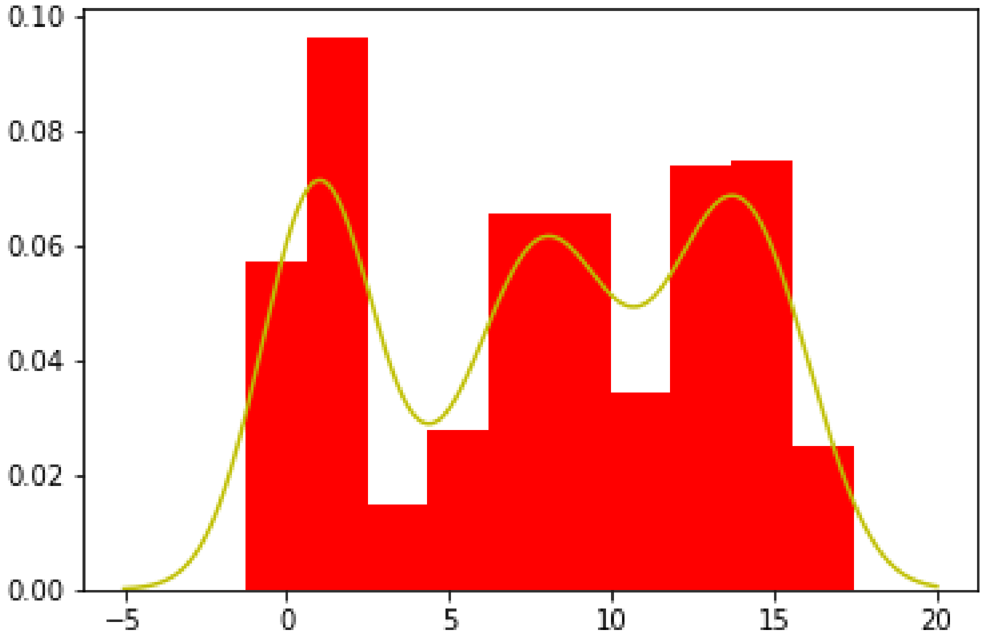
4.2.3 3) (ii) Univariate KDE with Statsmodels:

```
In [5]: kde_univariate = KDEUnivariate(dataset)
        kde_univariate.fit()
        plt.hist(dataset, normed=1, color= 'r', label= 'data')
        plt.plot(kde_univariate.support, kde_univariate.density, color= 'b',
                 label= 'univariate')
        plt.legend()
        plt.show()
```



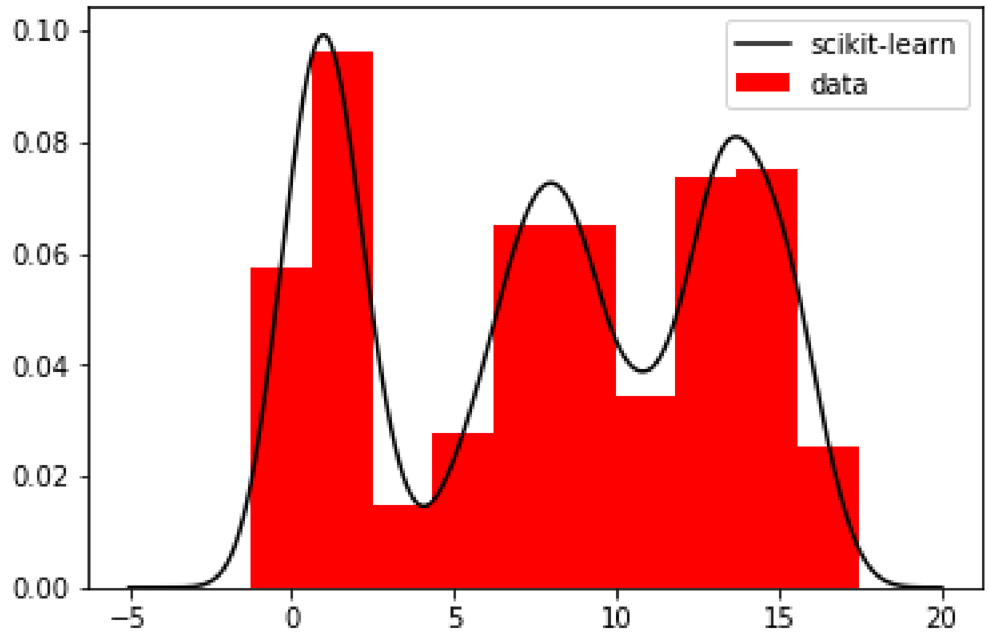
4.2.4 3) (iii) Multivariate KDE with Statsmodels:

```
In [6]: X_plot = np.linspace(-5, 20, 1000)
        kde_multivariate = KDEMultivariate(dataset, var_type= 'c')
        predicted_data = kde_multivariate.pdf(X_plot)
        plt.hist(dataset, normed=1, color= 'r', label= 'data')
        plt.plot(X_plot, predicted_data, color= 'y', label= 'multivariate' )
        plt.show()
```



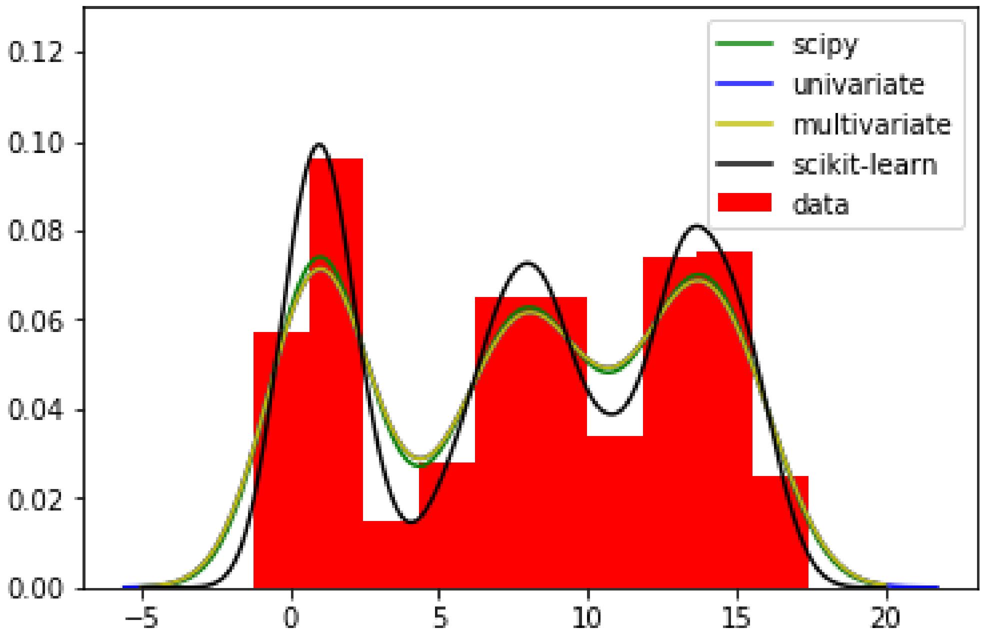
4.2.5 3) (iv) KDE with Scikit-learn:

```
In [7]: X = np.linspace(-5, 20, 1000)[:, np.newaxis]
        kde_skl = KernelDensity(kernel='gaussian', bandwidth=0.75).fit(dataset.reshape(-1,1))
        log_dens = kde_skl.score_samples(X)
        plt.hist(dataset, normed= 1, color= 'r', label= 'data')
        plt.plot(X[:, 0], np.exp(log_dens), color= 'k', label= 'scikit-learn')
        plt.legend()
        plt.show()
```



4.2.6 4) Plotting all four distributions on one figure:

```
In [8]: plt.hist(dataset, normed=1, color='r', label= 'data')
        plt.plot(X_plot, pdf, color='g', label= 'scipy')
        plt.plot(kde_univariate.support, kde_univariate.density, color='b',
                 label= 'univariate')
        plt.plot(X_plot, predicted_data, color= 'y', label= 'multivariate' )
        plt.plot(X[:, 0], np.exp(log_dens), color='k', label= 'scikit-learn')
        plt.ylim(ymax= 0.13)
        plt.legend()
        plt.show()
```


scipy
univariate
multivariate
scikit-learn
data
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−5
0
5
10
15
20