Report for Spring 2019 PRML Assignment # 1

Date: March 20, 2019

Abstract: The purpose of this assignment was to gain experience with the three non-parametric density estimation algorithms. In this assignment, we were tasked to design the three algorithms: histogram estimation, kernel density estimation, and nearest neighbor methods. After designing the algorithms, we are supposed to vary the number of data and compare the results Overall, this assignment served to familiarize us with the three algorithms and warm us up for future assignments.

1 Introduction

For this assignment, I experienced in designing simple machine learning algorithms. I followed the report requirements [1] and instructions from our T.A. Zhifeng. We were assigned a task of designing and analyzing histogram estimation, kernel density estimation, and nearest neighbor methods . This assignment consisted mainly of understanding the algorithms, coding them, and comparing the results while varying the data size.

This report is organized with Section 2 goes over the task assigned. Section 3 has the conclusion.

2 Assignment

2.1 Histogram estimation

The coding portion for this algorithm is fairly simple since most of it was given already. The code is shown in Figure 1.

```
def hist_est(bins_num):
    plt.hist(sampled_data, normed=True, bins=bins_num)
    plt.show()
```

Figure 1. Code for Histogram Estimation

Changing the number of bins of the histogram will affect the resolution of the estimation, shown in Figure 2. To pick the good amount for bins we need to consider how different is the data. For example, if the data does not vary by a lot we can pick a low number of bins to get a close result. However, if the data varies a lot we need to increase the number of bins. In addition, when we increase the number of bins to a certain point the graph will have a less change. Therefore, we can increase the number of bins until the change in graph is less noticeable.

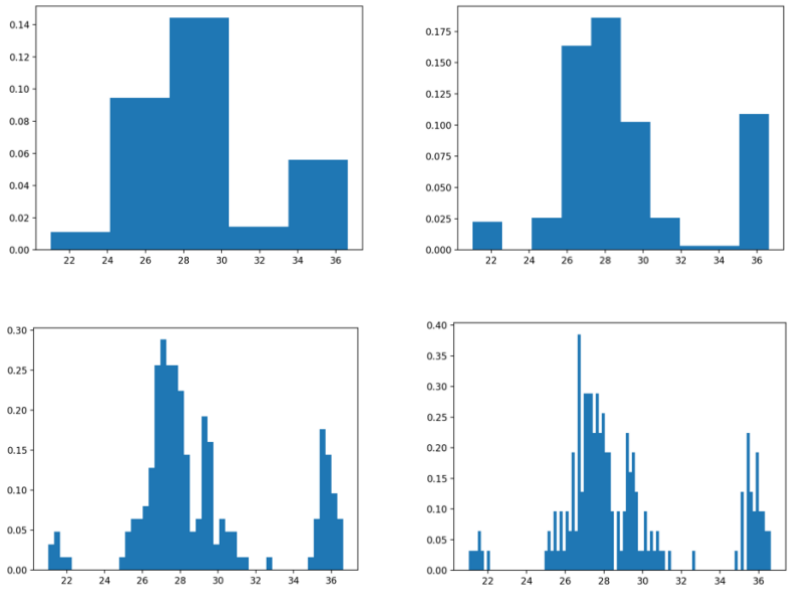

Figure 2. Graph for 5, 10, 50, 100 Bins Respectively

2.3 Kernel Density Estimation

For this code I implemented the gaussian kernel as instructed, shown in Figure 3.

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{1/2}}\exp\left\{-\frac{||\mathbf{x}-\mathbf{x}_n||}{2h^2}\right\}$$

Figure 3. Equation of Gaussian Kernel [1]

```
def kde(h):
    N=len(sampled_data)
    term=[]
    mini=min(sampled_data)
    maxi=max(sampled_data)
    maxi_y=0;
    c=1/(N*(2*math.pi*h**2)**0.5)
    x=np.linspace(mini,maxi,num=200)
    for e in x:
        s=0
        for d in sampled_data:
            s=s+math.e**(-(e-d)**2/(2*h**2))
        term.append(s*c)
        if s*c>maxi_y:
            maxi_y=s*c
    maxi_y=maxi_y*1.1
    plt.plot(x,term)
    plt.show()
```

Figure 4. Code for Gaussian Kernel Estimation

The task was to tune the bandwidth h for the estimation for 100 data samples. The first method I tried was the trial and error, shown in Figure 5. For h = 0.5 has the best result out of all the trials.

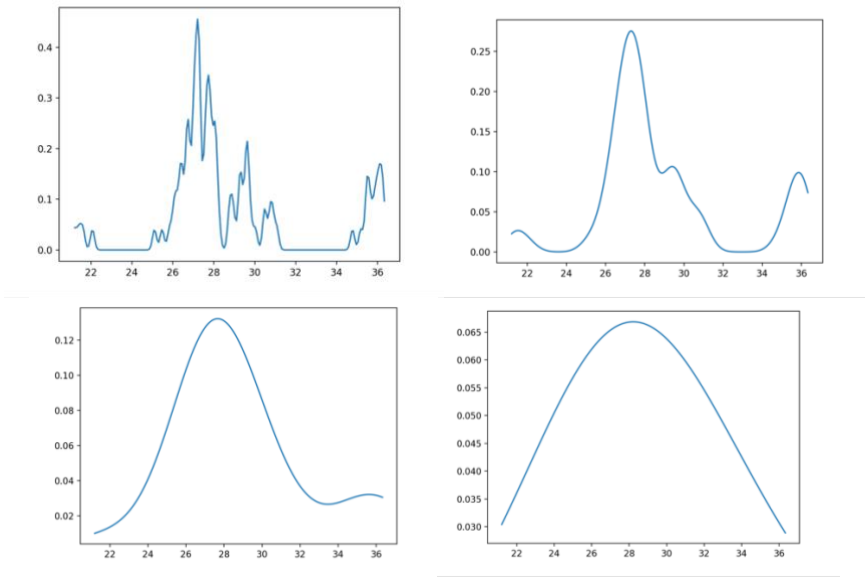

Figure 5. Graph for h = 0.1, 0.5, 2, 5 Respectively

Next, I tried the Silverman's rule of thumb, which is equation shown in Figure 6. The h came out to be approximately 1.46. The graph is shown in Figure 7.

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-1/5}$$

Figure 6. Silverman's Equation

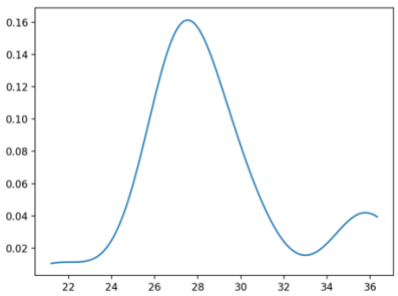

Figure 7. Graph for h = 1.46

2.4 Nearest Neighbor Method

Next, I coded the nearest neighbor method. Code shown in Figure 8.

```
def knn(K):
    mini=min(sampled_data)
    maxi=max(sampled_data)
    x = np.linspace(mini, maxi, 200)
    y = np.zeros_like(x)
    ld=len(sampled_data)
    yi=0;
    for i in x:
        dis=np.zeros_like(sampled_data)
        index=0
        for q in sampled_data:
            dis[index]=abs(q-i);
            index=index+1;
        dis.sort()
        V=dis[K-1]
        y[yi]=K/(ld*V)
        yi=yi+1
    plt.plot(x, y)
    plt.show()
```

Figure 8. Code for Gaussian Nearest Neighbor Method

I changed the K value which is the number of neighbors in to consideration, shown in Figure 9. K controls the degree of smoothing, the lower the number more the noise. This method does not always yield a valid distribution. For example when K = 1 the graph goes up to 20 which make it not converge to 1.

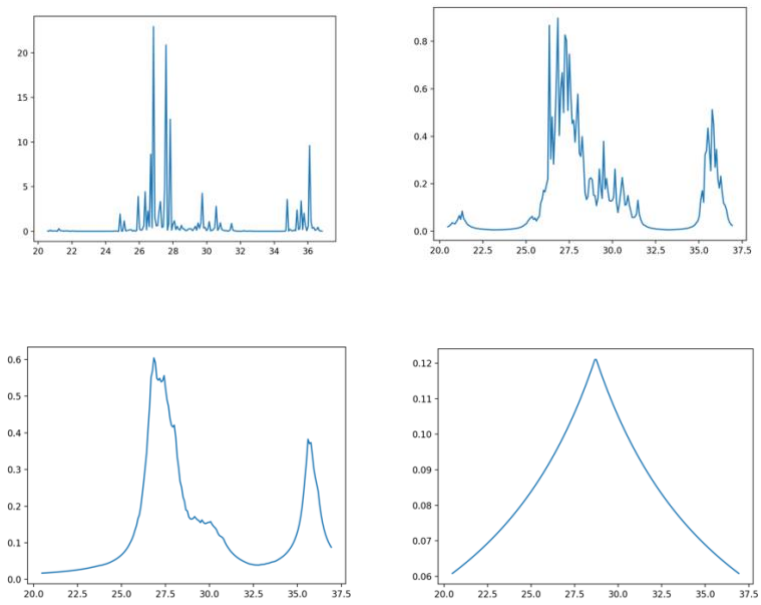

Figure 9. Graph for K = 1, 10, 100, 1000 Respectively

2.5 Number of Data Used

We were asked to vary the number of data used for all three algorithms. For simplicity purpose Figure 10 only shows the change for histogram. The amount of data does not matter as long there is sufficient amount of them.

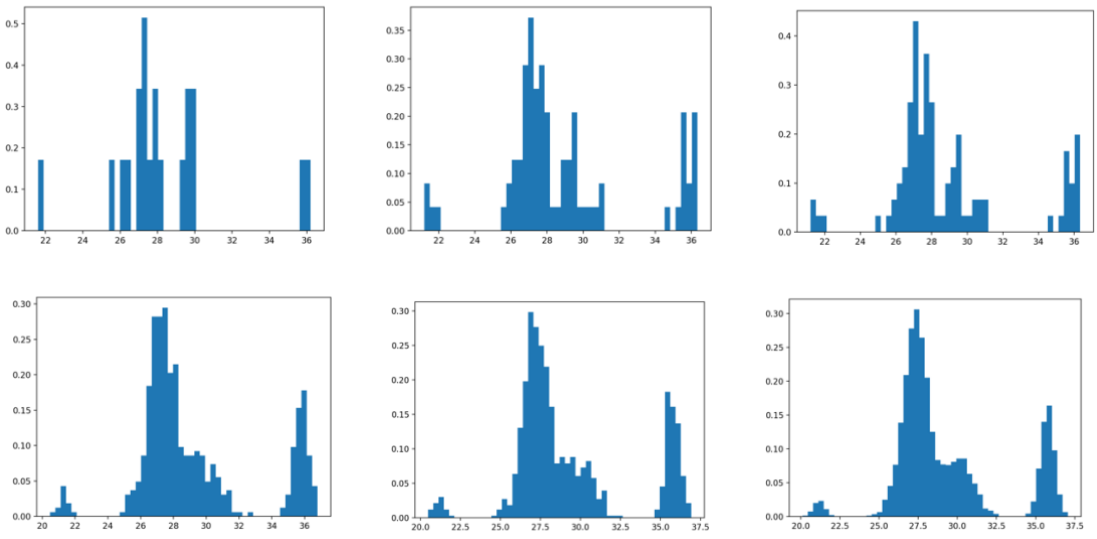

Figure 10. Graph for 20, 80, 100, 500, 1000, 10000 Numbers of Data Respectively

3 Conclusion

Overall, the assignment was successful as I was able to finish it and further understand the three algorithms. This assignment was a very good warm up for future assignments.

References

1. “Assignment 1” Pattern Recognition and Machine Learning, Fudan University, Spring 2019, <https://zfhu.ac.cn/PRML-Spring19-Fudan/assignment-1/index.html>.