Yue Pan
yuepan@student.ethz.ch

Assignment 6

Image Interpretation
December 13, 2019

Exercise 1

Topic: K-Means.

For this exercise, please refer to *ii_kmeans.m*.

Firstly, for the fixed dataset, I change the number of cluster K to compare K-means' performance. The converged result can be shown in Fig.1. Since the reasonable number of cluster is 6 for this dataset, it is found that when $K = 6$, K-means get the most reasonable result. K-means would merge several natural cluster into one when $K < 6$ and split one natural cluster into several parts when $K > 6$.

Then we test K-means on the randomly generated dataset and get the results shown in Fig.2 and Fig.3. In this case, the natural reasonable number of cluster is hard to tell. So we can notice that the setting of K is really subtle here. In practice, to properly set K, we should take some prior knowledge of the data and the application into account and also tune K according to the feedback. This is also a common issue for unsupervised learning, since what we learn is just some distribution or inner structure of the data. How to further use them, assign the semantic meaning of the clusters and guide the model to better provide such kind of information is we human beings's duty.

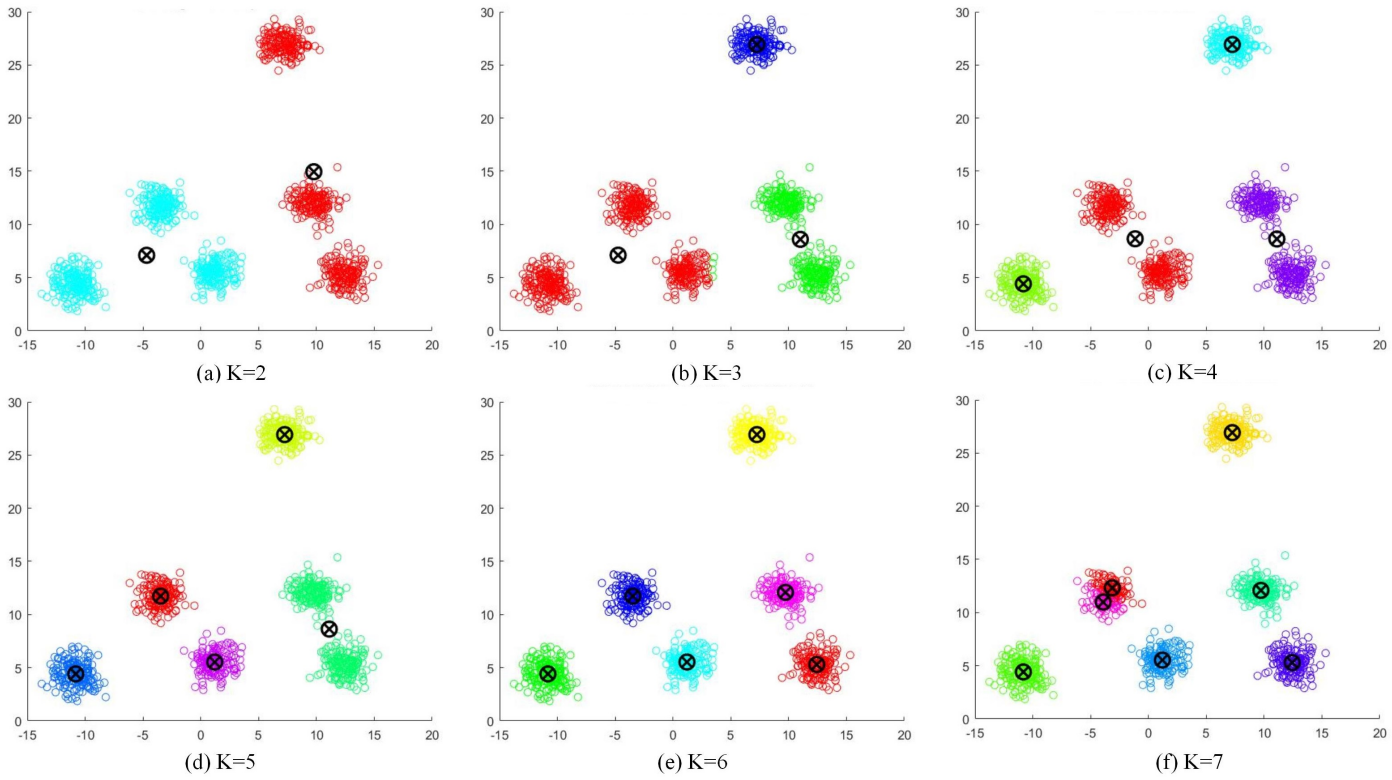

Figure 1: KMeans with different K on fixed dataset

Actually, K-Means algorithm is a special case of the Expectation Maximization (EM) algorithm. Since K-means use the hard coding of label (or probability of label) instead of the fuzz coding, it's very likely to converge to local optima when the initialization is not good enough. In Fig.5, we can also notice that KMeans failed on this case due to its sensitivity to initialization.

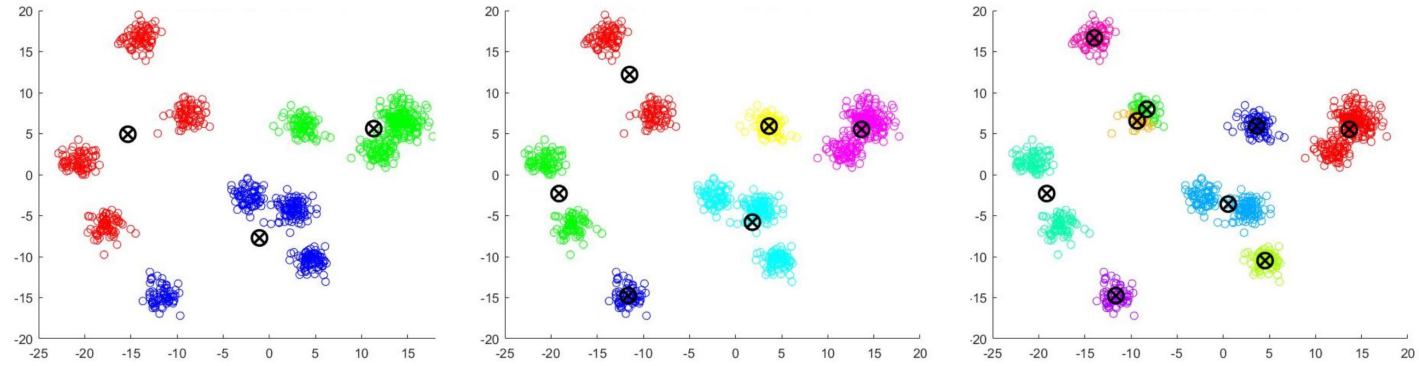

Figure 2: KMeans with different K on random dataset 1

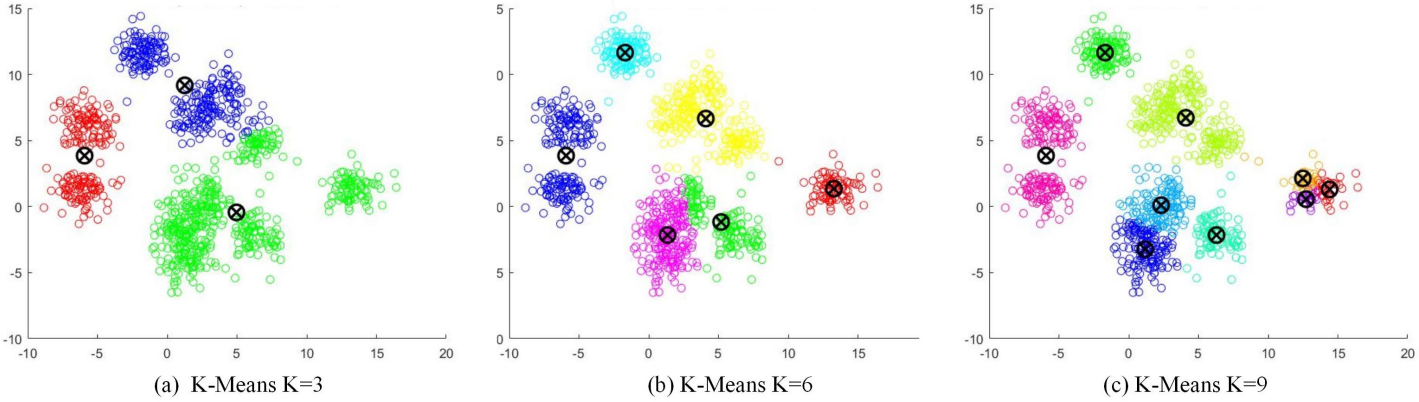

Figure 3: KMeans with different K on random dataset 2

The proper number of K and the proper initialization method of cluster would help K-means to avoid failure. A option of the initialization strategy is the so-called Furthest Point Heuristic, which means each time we select the farthest point to the current point as the next initial position of cluster center.

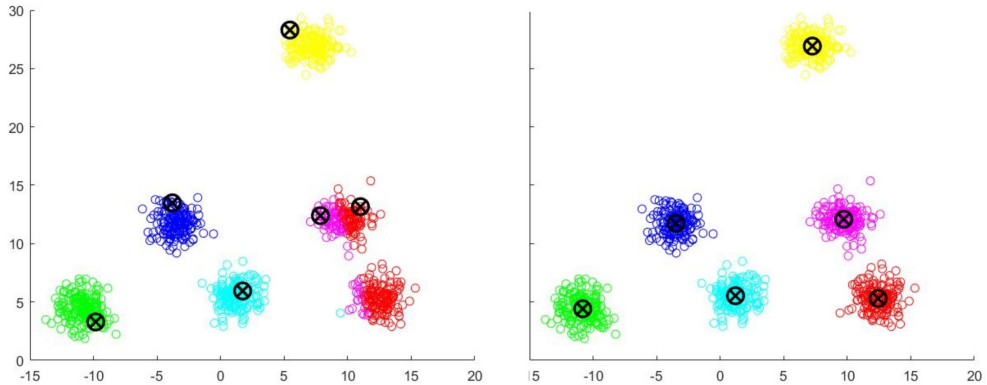

Figure 4: KMeans successful case on fixed dataset

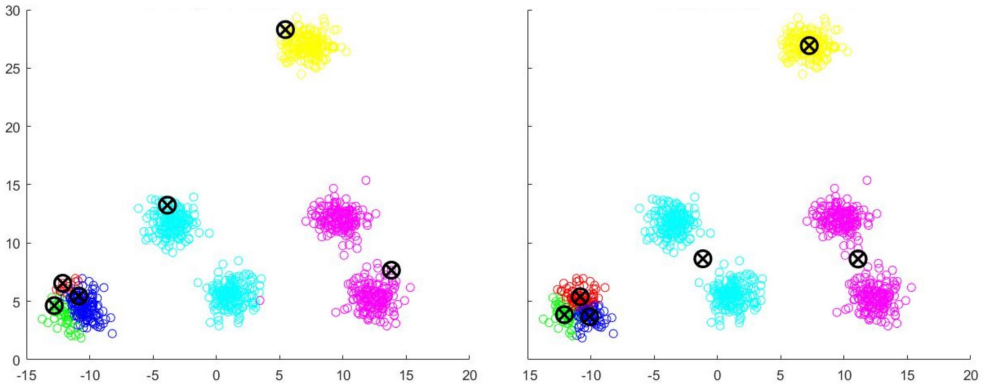

Figure 5: KMeans failed case on fixed dataset

Exercise 2

Topic: Kernel Density Estimation

For this exercise, please refer to *ii_kde.m*.

Kernel density estimation (KDE) is a kind of non-parametric method to estimate the approximate probability density function of a random variable with a local kernel function with a fixed bandwidth. Different from using histogram to approximately describe the probability density function, we use a combination of these kernels to fit the density function so that we can get a continuous estimation.

There are two key factors, one is the kernel function and the other is the bandwidth of the kernel. So we compare the perfomance with different bandwidth (0.5, 1 and 2) and kernel (Parzen, Normal and Epanechnikov) for different datasets. The results are shown from Fig.6 to Fig.9.

For the single sample case (Fig.6), we can see the shape of each kernel with different bandwidth clearly. So the next step is to use the combination of such kind of single kernels to fit the dataset with more samples.

Then for the general one dimensional data, the results are shown in Fig.7. It is noticed that the larger the bandwidth, the coarser the estimation would be. Besides, we can find that Epanechnikov kernel with bandwidth 0.5 has the best estimation of the probability density function. The curve of Parzen kernel is close to the ground truth but not smooth enough due to the binary property of Parzen kernel (within bandwidth: 1 , out of bandwidth: 0). The normal kernel's curve is very smooth but is far away from the ground truth.

For the 2-dimensional case (Fig.8), the conclusion is similar. Epanechnikov kernel has the best fit of the probability density function. However, in this case, the one with larger bandwidth ($h = 2$) is more smooth and close to the true distribution.

Theoretical speaking, it can be proved that the Epanechnikov kernel is optimal in the sense of mean square error, which is also validated by the experiment here.

Finally, for the dataset with data gaps (Fig.9), Epanechnikov kernel's curve is not smooth any more while Normal kernel's curve is still smooth. In this case, Normal kernel with bandwidth 0.5 has the better fitting perfomance than Epanechnikov kernel. This is due to the fact that Epanechnikov kernel is similar to normal distribution within the bandwidth while has zero response out of the bandwidth. When

the sample of the actual distribution is not uniform (which means there are some gaps), Epanechnikov may lose its accuracy since the local samples are not enough. However, Normal kernel can "see" further so that even if there's some gap of sample locally, it can still involved the generally complete global samples of the distribution.

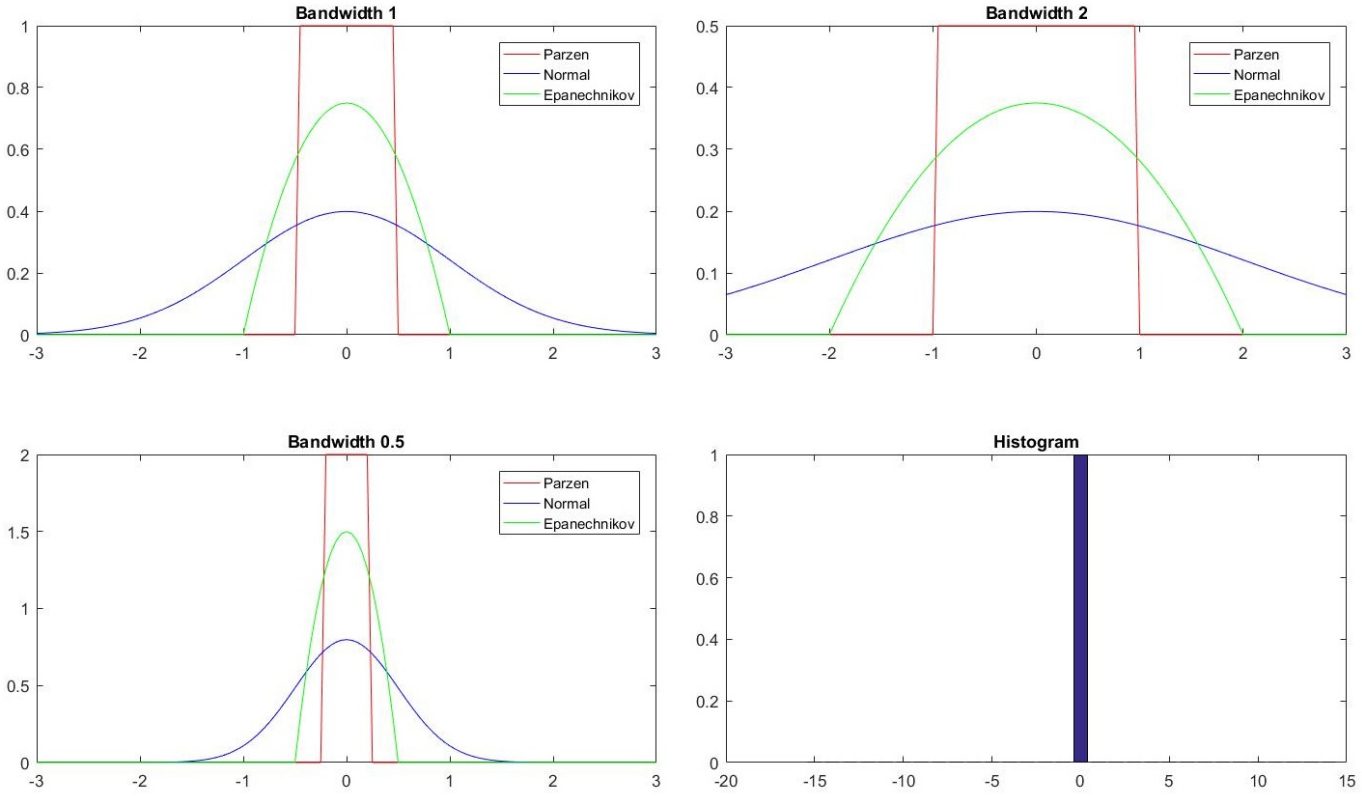

Figure 6: KDE for 1D data (single sample) using different kernel function and different bandwidth

Exercise 3

Topic: Mean-shift

For this exercise, please refer to *ii_meanshift.m* and *find_mode.m*.

Firstly, I ran Meanshift on the fixed dataset and with different parameters (converge threshold θ and bandwidth h). It is found that θ does not have too much influence on the clustering result while h affects the result to a large extent, as shown in Fig.10. On the randomly generated dataset, we can the result shown in Fig.11. Generally speaking, the change of bandwidth would affect the number of points in certain window, which would also affect the converged mode. Besides, since the threshold for merging two modes is set as $h/5$, it would also affect the number of modes. From the experiment, it's found that generally speaking, the larger h is, the less clusters would be found.

To compare Meanshift with K-means, I'd like to claim that Meanshift would be more stable than K-means because Meanshift do not need a initialization which should be the first step of K-Means. There's no

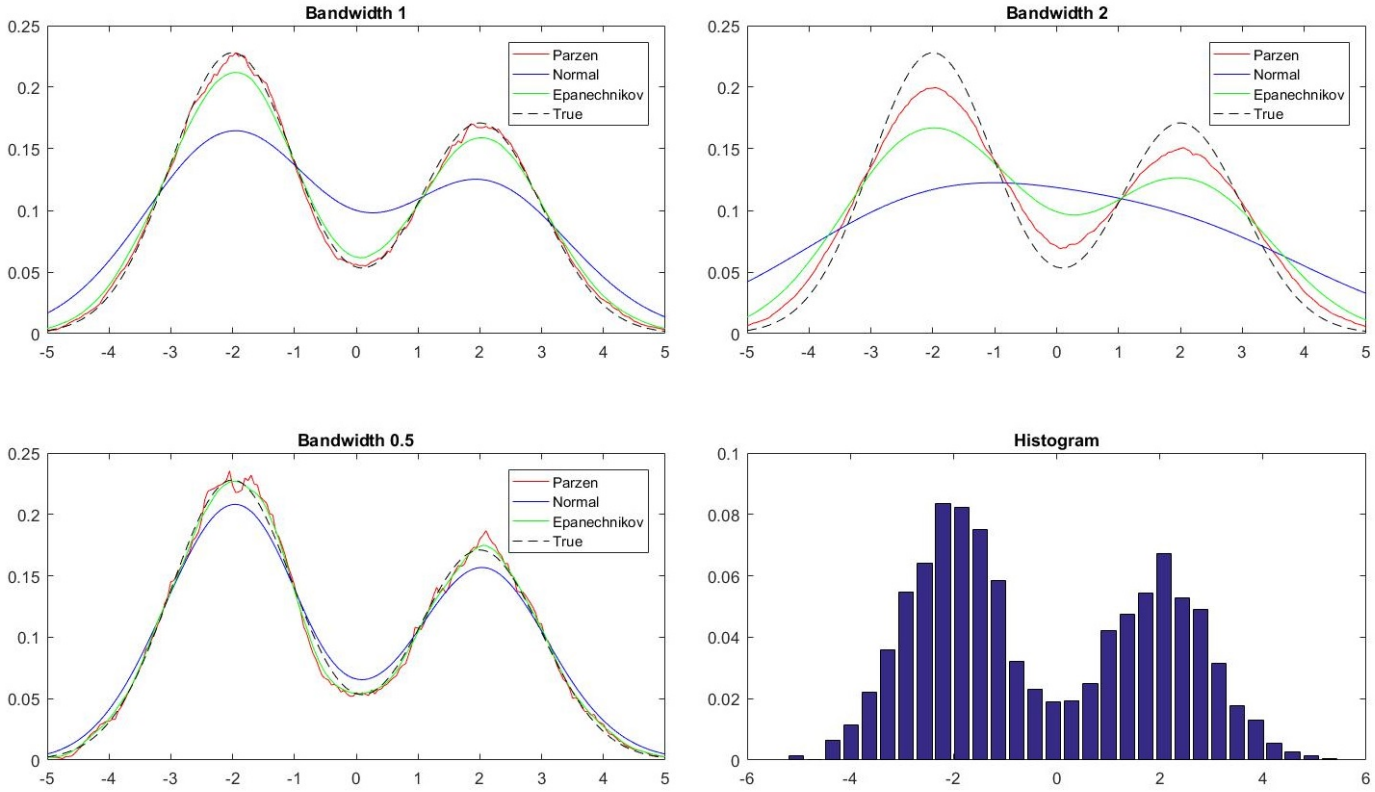

Figure 7: KDE for 1D data using different kernel function and different bandwidth

randomness in Meanshift so it would always get exactly the same clustering result. It is noticed that when $d = 3, 5$, Meanshift gets the accurate clustering while K-means may fail.

Another case is the time complexity comparison. K-means 's time complexity is $O(nkd)$, in which n and d are the number and dimension of the input data respectively and k is the number of clusters. Meanshift is a bit more time-consuming since its time complexity is $O(n^2d)$. So if K can be known accurately as a prior knowledge and a good initialization strategy is used, K-means may be a better choice than Meanshift.

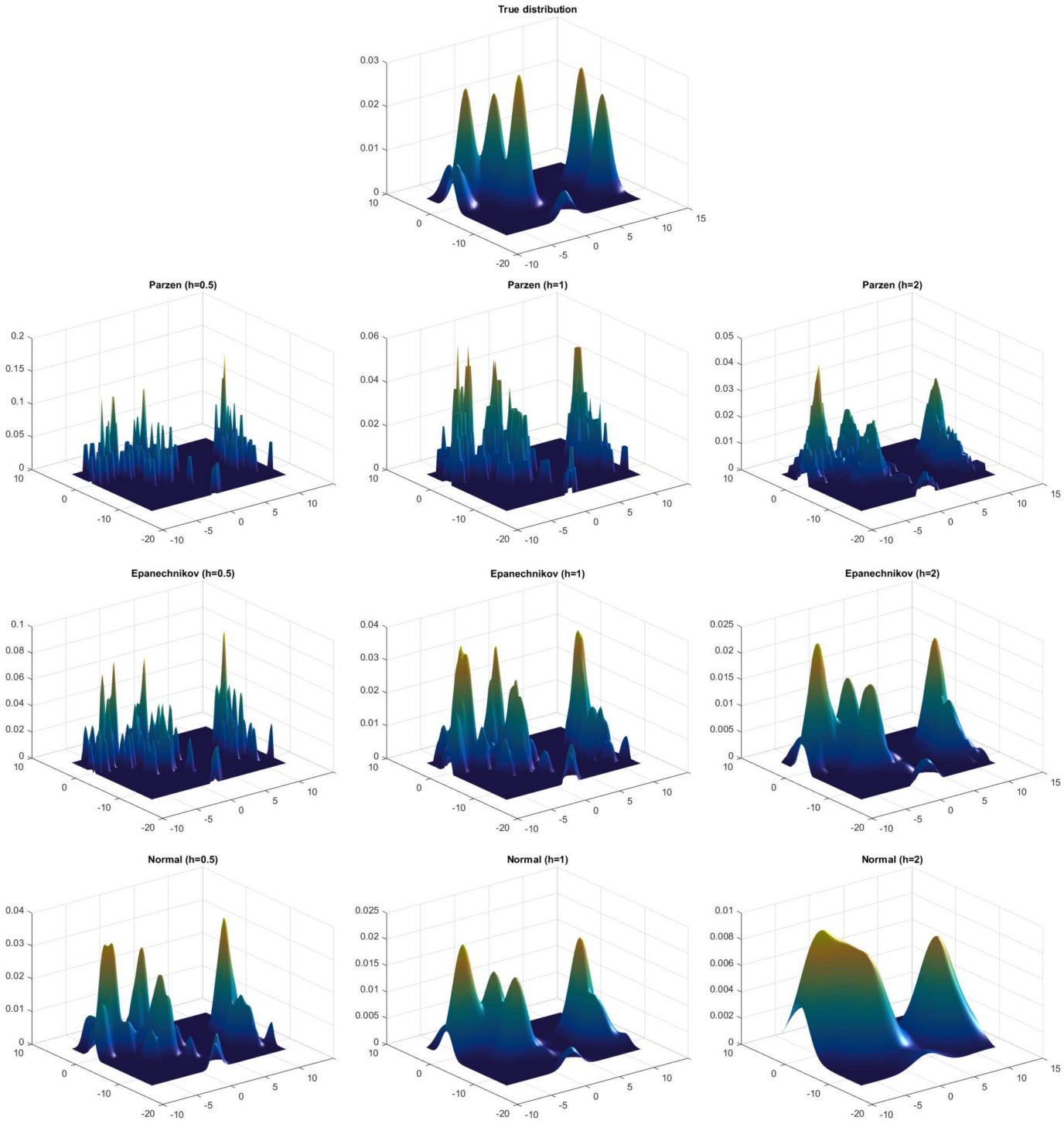

Figure 8: KDE for 2D data using different kernel function and different bandwidth

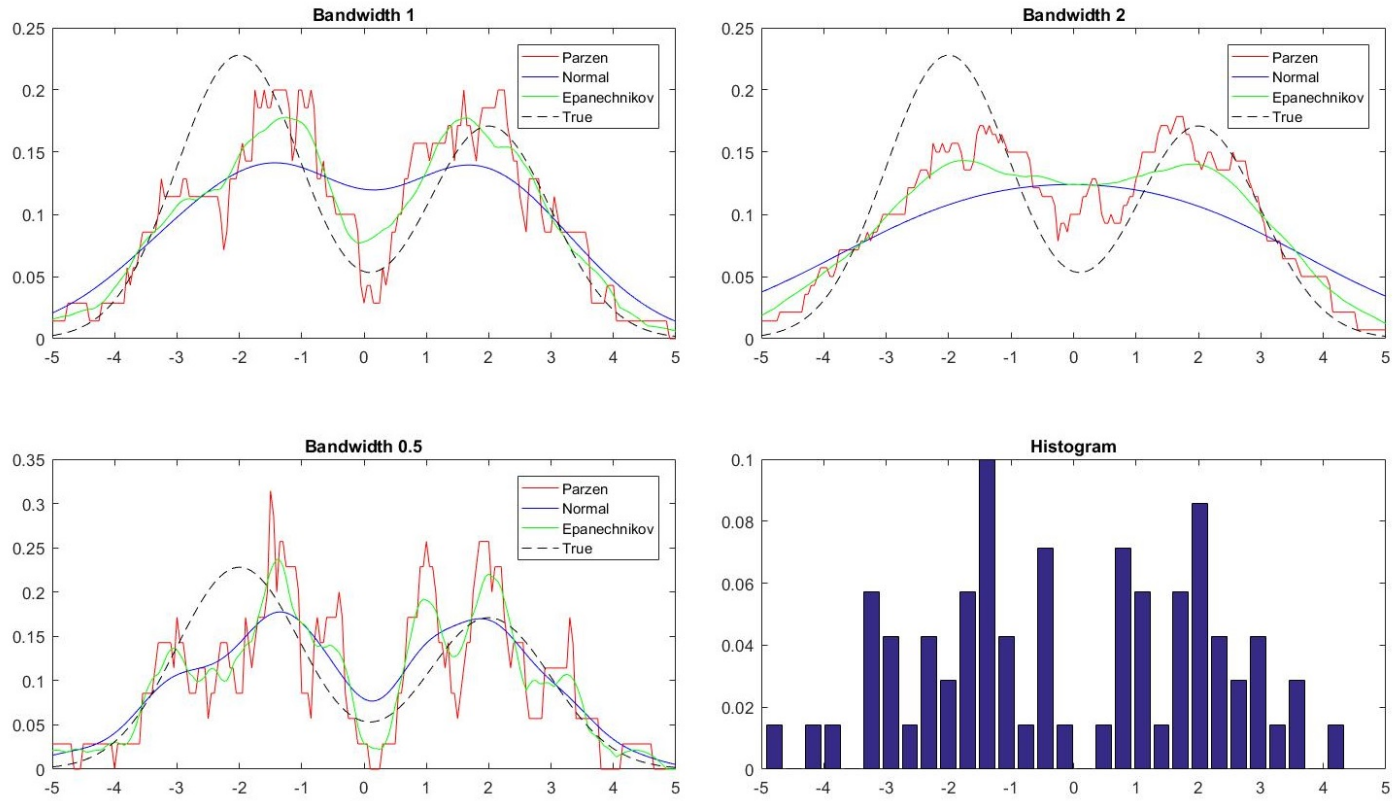

Figure 9: KDE for 1D data (few samples with gaps) using different kernel function and different bandwidth

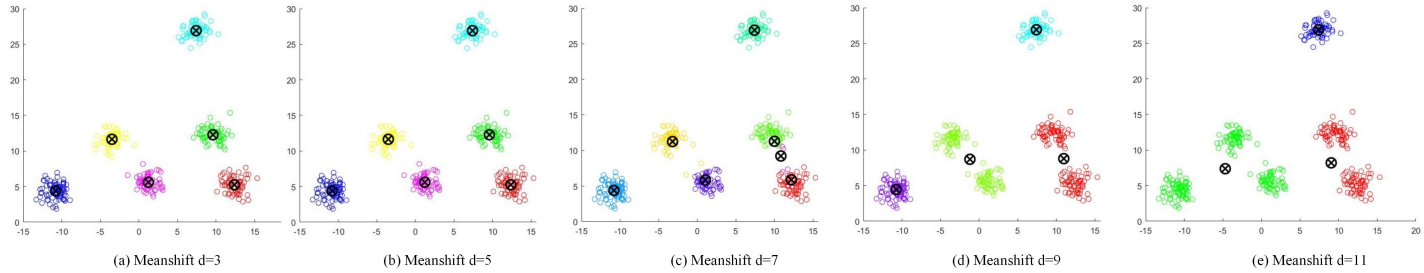

Figure 10: Meanshift performance on fixed dataset w.r.t different bandwidth

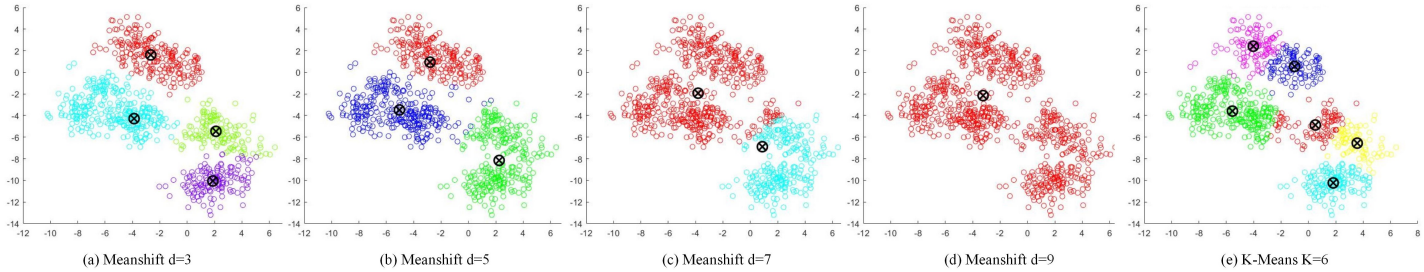

Figure 11: Comparison of Meanshift with different bandwidth and K-Means on random dataset