Lab Report 3: Density Estimators

Nicholas Piël
npiel@science.uva.nl
cknr: 0104612

2007-10-10

Abstract

In this lab we implement two density estimators, a nearest neighbor density estimator and a kernel density estimator with a Gaussian kernel. I will show the result of their ability to classify on two different problems. Namely the 'iris-dataset' and some sort of banana distribution.

1 Introduction

In this lab report we will work with two different estimators and two different datasets leading to four possible combinations. I will start this report by giving a quick overview of the different approaches, then give a quick overview of the datasets after which i will discuss my expected result. In section 5 i will show the results and in section 6 i will discuss them.

2 Approach

2.1 Parzen Density Estimator

The first approach is the Parzen density estimator based on the Gaussian kernel. We have chosen the Gaussian kernel because we will then obtain a smoother density model. We define the density at x as follows:

$$p(x) = \frac{1}{N} \sum_{n=1}^{n} \frac{1}{\sqrt{2\pi h^2}} exp\{-\frac{||x - x_n||^2}{2h^2}\} \tag{1}$$

In here x is the point to be tested and x_n belongs to the training data, h represents the standard deviation of the Gaussian components and this parameter allows us to tweak the estimator.

2.2 Nearest Neighbour

The Nearest Neighbour approach is really simple and its unconditional density is given by:

$$p(x) = \frac{K}{NV} \tag{2}$$

In this case K is the parameter we can tweak, N is the total number of points and V is the volume of the sphere that surrounds K points.

2.3 Classifying

Now we know our density estimators, we can easily derive the unconditional density and the conditional density from above function. This in combination with the class priors gives us enough information to use Bayes theorem to calculate the posterior probability of class membership:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} \tag{3}$$

In this case $p(x)$ is the density estimator based on the complete training set, and $p(x|C_k)$ is the estimator based on the trainingset with only elements of class C_k. The class priors is simply given by:

$$p(C_k) = \frac{N_k}{N} \tag{4}$$

Where N_k is the number of points in the training set belonging to class k and N is the total number of points. From this we can derive a really nice simplification to the k-NN classifier: $p(C_k|x) = \frac{K_k}{K}$.

2.4 Cross Folding

Now that we know our formulas, know where to plugin our data and which parameters to tweak we can easily create a loop that runs over a range of parameters and measure the performance of each combination. For this we will use the following measure:

$$accuracy = \frac{L_{correct}}{L_{total}} \tag{5}$$

Where we simply divide the correct returned labels on the total amount of labels. Since we want to limit the effect of randomness we repeat the experiment 10 to 100 times with a different permutation of the data. Then from this permutation we use 80% for training and 20% for testing.

3 Datasets

3.1 Iris Dataset

The iris dataset is a set of data with a total of 150 instances and 3 different classes. This data is static and commonly used in machine learning tasks.

3.2 Banana Dataset

The banana dataset is a set of data generated from a function. Since i need to port the function that generated this data from Matlab to Python i have included a plot (figure 3.2) so the correctness of this function can be verified.

The banana dataset contains a total of 600 points divided equally over 2 classes distributed in some sort of bananashape.

4 Expectations

As already explained we have two parameters to tweak, i expect that when using the Parzen kernel there will be some sot of optimum where the accuracy is high and the standard deviation is low. When choosing a lower value i think that the accuracy will drop really fast since the model is too noisy. And a higher value will make the model too smooth and thus increasing its standard deviation. With the k-NN approach i think there is more a gradual decent, in which both the accuracy will go down and its standard deviation will go up. This because the higher we choose k, the fewer regions we will have and the more it will generalise.

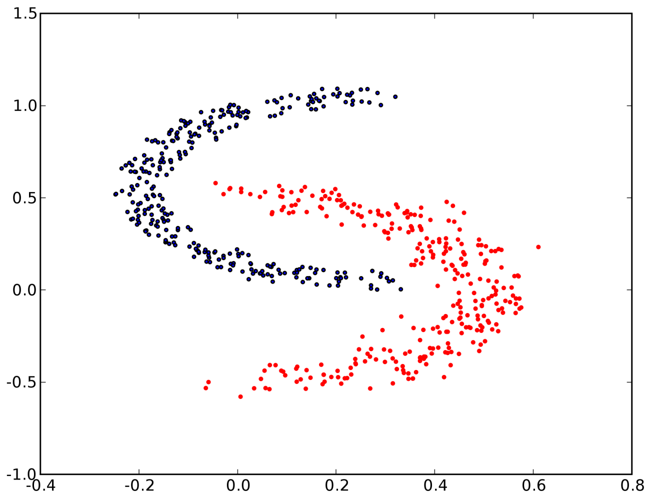


Figure 1: Plot of the banana distribution

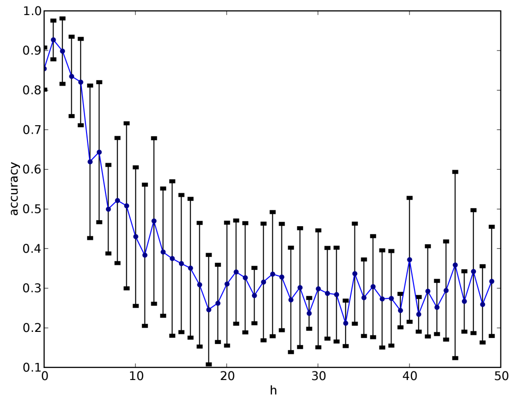
5 Results

The resuls can be seen in figure 2. It contains a plot of the mean over the accuracy and the standard deviation over the accuracy.

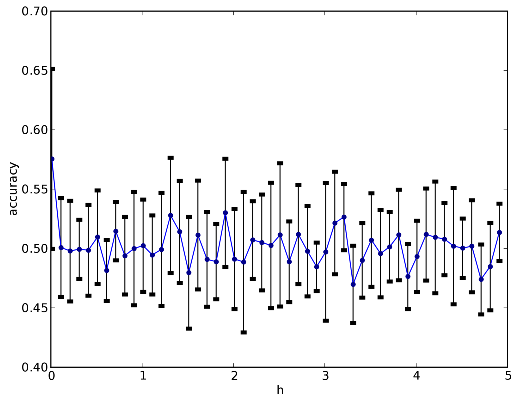
6 Discussion

We can see that in both datasets k-NN outperforms the Parzen estimator. While the Parzen estimator has an OK performance on the iris dataset it performs really bad on the bananaset. We can see that for k-NN its best performance is indeed for lower values of k, and that the standard deviation increases when we increase k as expected.

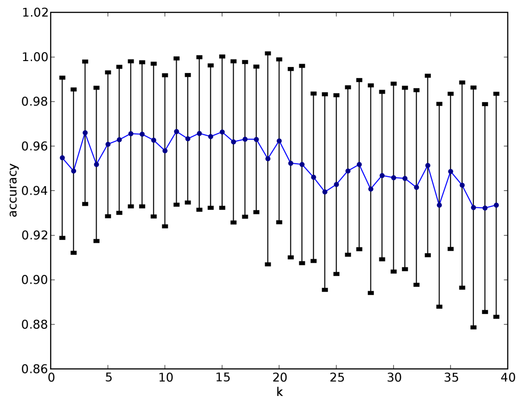
An interesting question that remains us, is why is the Parzen estimator performing so bad en the banana dataset. The reason is that the problem is not linear separable, and thus the Gaussian kernel cannot differentiate between the two different classes.



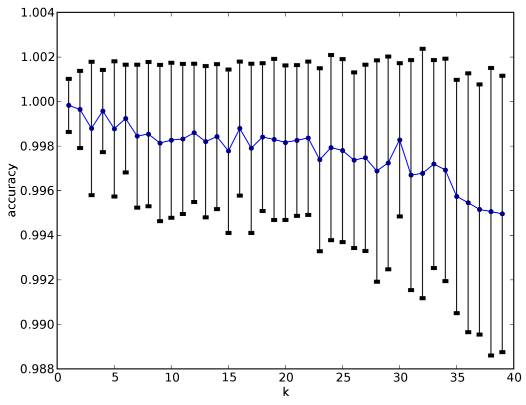
(a) Parzen estimator on the iris dataset



(b) Parzen estimator on the banana dataset



(c) k-NN estimator on the iris dataset



(d) k-NN estimator on the banana dataset

Figure 2: Density Estimators