An Approach to Classification using Kernel Density Estimation and Bayesian Classification

Rishabh Gupta

December, 2020

1 Introduction

Classification is a common class of problem in Machine/Statistical learning. There are many methods to approach a classification problem. The method I will be exploring is using kernel density estimation (abbreviated as KDE). KDE is a nonparametric method to estimate the probability density of the data. It is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample. Once we get a smooth density estimate for our data,we then use Bayesian classifier to predict its class. This method is useful when our data does not follow a known distribution (for example when our data is multimodal) or when the assumptions of our models are violated. Then the widely used parametric methods fail to make any meaningful inference. We can then use this approach to make make inferences with fewer assumptions, which makes it more robust. This makes it applicability much wider and can be useful when we have limited domain knowledge.Since this methdology relies on data smoothing,in essence it attempts to capture only the important patterns, while filtering noise and ignoring the data structures that are deemed not relevant.

2 Methodology

This methodology I will be using involves finding the density estimate of data for each classes using the Kernel density estimation and then using this density estimate to predict the classes using Bayesian classifier.

The univariate kernel density estimator for a random sample X_1, X_2, . . . X_n drawn from a common and usually unknown density f is given by

$$\begin{aligned}\hat{f}(x,h) &= \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-X_i}{h}) \\ &= \frac{1}{n}\sum_{i=1}^{n} K_h(x-X_i)\end{aligned}$$

where h is a positive number known as the "bandwidth" paramter and K is a kernel function. A general form of the d-dimensional multivariate kernel density estimator is given by

$$\begin{aligned}\hat{f}(x,H) &= n^{-1}\sum_{i=1}^{n} |H|^{-1/2} K(H^{-1/2}(x-X_i) \\ &= n^{-1}\sum_{i=1}^{n} K_H(x-X_i)\end{aligned}$$

where H is a n X n bandwidth matrix which is symmetric and positive definite. K and K_H are unscaled and scaled multivariate kernel functions.

The most important thing in this method is selecting our bandwidth parameter 'h'(or 'H') to get a accurate estimate. There are three main class of bandwidth selectors Rule-of-Thumb, Plug-ins and Cross-Validation selectors. In this paper I'll be mainly comparing the two selectors: Rule-of-thumb and Cross-validation selector.

Kernel Density Classification: Once we have calculated our kernel density estimate we use it for classification in a straight-forward fashion using Bayes theorem. Suppose for a J class problem we fit nonparametric density estimates $\hat{f}_j$ (X) , j = 1, . . . , J separately in each of the classes, and we also have estimates of the class priors $\hat{\pi_j}$ (usually the sample proportions). Then

$$\hat{P}(G = j|X = x_0) = \frac{\hat{\pi_j}\hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi_k}\hat{f}_k(x_0)}$$

Another trick is that if classification is the ultimate goal, then we need only to estimate the posterior well near the decision boundary. Learning the separate class densities is unnecessary. In learning the separate densities from data, one might decide to settle for a rougher, high-variance fit to capture these features, which are irrelevant for the purposes of estimating the posterior probabilities.

Naive Bayes Classifier using KDE: This is appropriate to use when the dimension p of the feature space is high, making density estimation unattractive. The naive Bayes model assumes that given a class G = j, the features X_k are independent:

$$\hat{f}_j(X) = \sum_{k=1}^{p} \hat{f_j k}(X_k)$$

While this assumption is generally not true, it does simplify the estimation dramatically:

- The individual class-conditional marginal densities $f_j k$ can each be estimated separately using one-dimensional kernel density estimates. This is in fact a generalization of the original naive Bayes procedures, which used univariate Gaussians to represent these marginals.
- If a component X_j of X is discrete, then an appropriate histogram estimate can be used. This provides a seamless way of mixing variable types in a feature vector.

It is widely known that despite these optimistic assumptions, Naive Bayes algorithm has comparable accuracy or even outperforms other complex models in real world usage.That being said, I will also compare the performance of the naive Bayes classifier (using KDE) with the multivariate KDE classifier ahead.

3 Simulation Study

First we begin our simulation study by comparing the two bandwidth selector methods "Rule-of-thumb" and "cross-validation" to see its performance. I have used Silverman's rule-of-thumb selector though there are other rule-of-thumb selectors such as Scott, Bowmann. Each of these had similar performance so I choose one for my simulation study. Similarly I used least square cross validation (LSCV) selectors for my simulation.

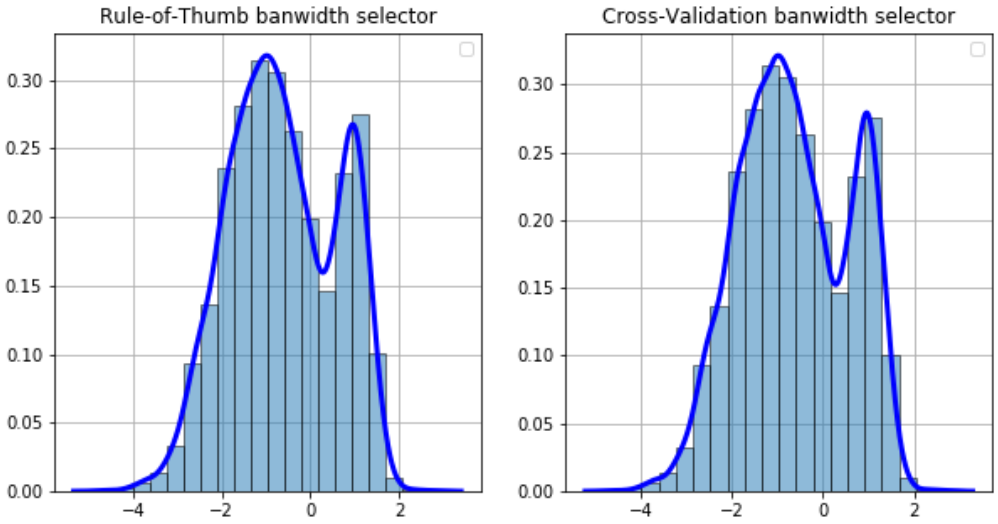

Figure 1: Density Estimation for Bimodal Normal: RoT vs CV bandwidth selector

Selector	D-Statistic	p-value
Silverman	0.586	3.46e-33
LSCV	0.546	1.92e-28

Table 1: Kolmogorov-Smirnov test for β (0.5,0.5): Silveman RoT vs Least Squares-Cross Validation Bandwidth selector

Firstly I have choosen a bimodal distribution (mixture of two Gaussians) to see the performance of both the selector. The distribution used for data simulation is given as below:

$$f(x) = \frac{4}{5}N(x; -1, 1) + \frac{1}{5}N(x; 1, 0.3).$$

As expected (Figure 1) both the selectors do a good job in bandwidth selection and density estimation using both of these is very similar. Though that is expected as rule-of-thumb selector does a good job when the distribution is similar to a Gaussian. So, we will next see how it performs on a non-Gaussian distribution. I have choosen to do a simulation on Beta(α, β) distribution with $\alpha = \beta = 0.5$.

As we see clearly from the visualization (Figure 2), cross-validation selector does a much better job of selecting the bandwidth. Also here (Table 1) is the summary of the Kolmogorov-Smirnov test for the goodness of fit of our estimate using both the selectors.

When our disribution is non-Gaussian (non-elliptical) it is always a good idea to use a cross-validation selectors to get a good fit for our estimate.

Next, is a simulation study to see how a KDE classifier work using a univariate data. Here I have taken three classes and each class has data coming from a three different distribution. I have kept all three classes balanced (500

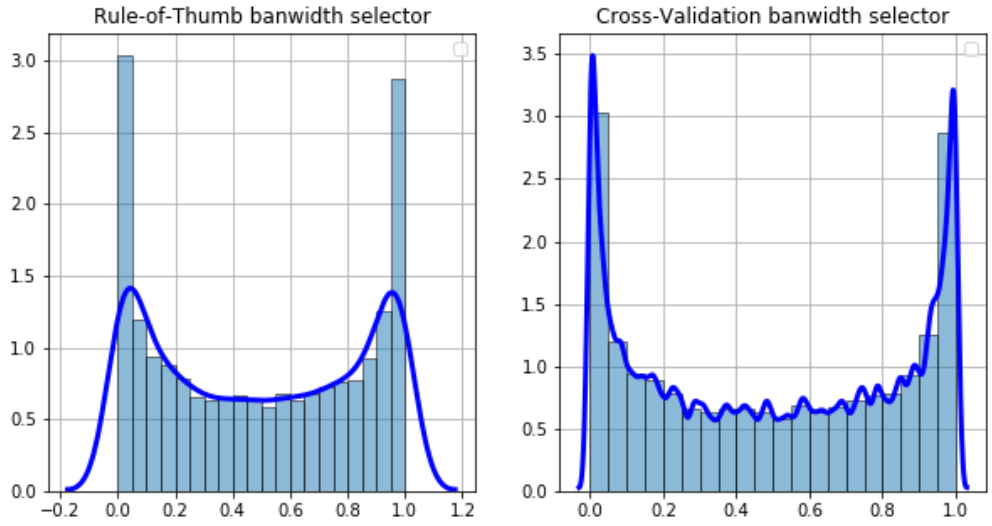

Figure 2: Density Estimation β (0.5,0.5): RoT(Silveman) vs CV(Least Squares)

samples from each class). I have kept 10 percent of the simulated data for testing to measure the performance the classifier. Distribution for each of the class (D1 for class 1, D2 for class 2 and D3 for class 3) is given below:

D1: $N(x; 60, 20)$
D2: Mixture of Gaussians ($\frac{1}{2}N(x; 100, 5) + \frac{1}{2}N(x; 15, 15)$)
D3: Scaled Beta($\alpha = 3, \beta = 1, a = 100, c = 150$)

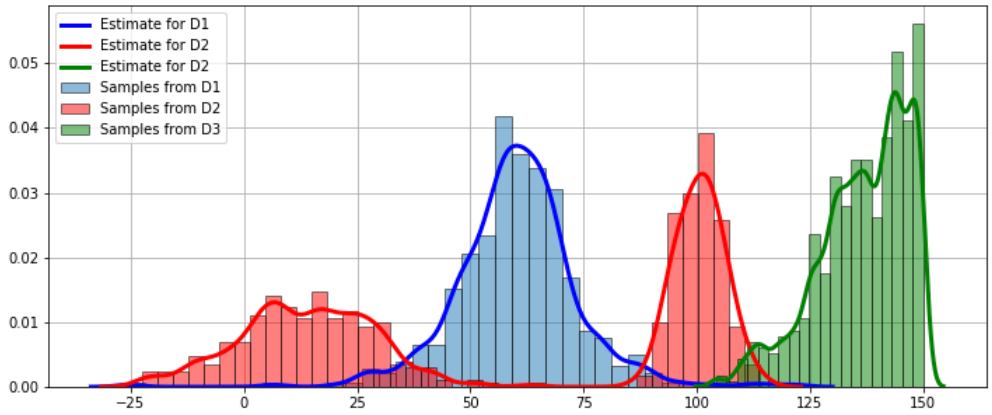

Figure 3: Univariate model Simulation

KDE of each class as given above and then these densities are used to predict the classes using the kDE classifier. I have measured the performance of classifier using the test data and the classification report is as below (Figure 4).

To see how the multivariate KDE works, I have simulated data from a mixture of multivariate Gaussian which is sortof a dumbell shaped distribution (Figure 5). I have also plotted the multivariate Kernel denstiy estimate alongside

	precision	recall	f1-score	support
1	0.96	0.98	0.97	50
2	0.98	0.96	0.97	50
3	1.00	1.00	1.00	50
accuracy			0.98	150
macro avg	0.98	0.98	0.98	150
weighted avg	0.98	0.98	0.98	150

Figure 4: Classification Report for Univariate model simulation

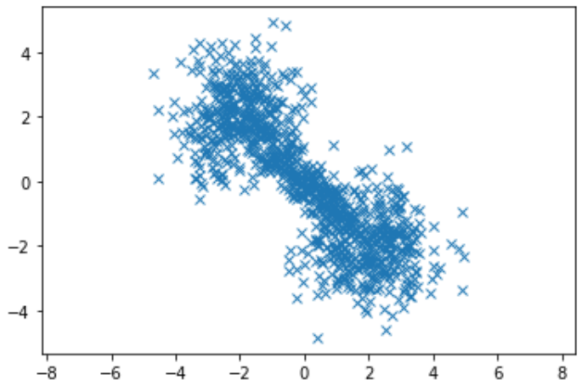

Figure 5: Samples from a dumbell shaped bivariate distribution

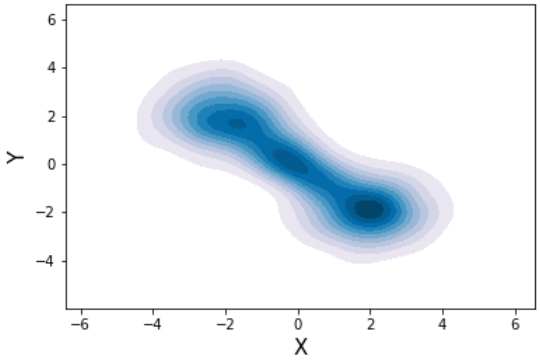

Figure 6: Density estimate for dumbell shaped bivariate distribution

it using a contour plot (Figure 6).The distribution is stated below.

$$\frac{4}{11}N(\begin{bmatrix}\mathrm{x}_1\\ \mathrm{x}_2\end{bmatrix};\begin{bmatrix}-2\\ 2\end{bmatrix},\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix})+\frac{3}{11}N(\begin{bmatrix}\mathrm{x}_1\\ \mathrm{x}_2\end{bmatrix};\begin{bmatrix}0\\ 0\end{bmatrix},\begin{bmatrix}0.8 & -0.72\\ -0.72 & 0.8\end{bmatrix})++\frac{4}{11}N(\begin{bmatrix}\mathrm{x}_1\\ \mathrm{x}_2\end{bmatrix};\begin{bmatrix}2\\ -2\end{bmatrix},\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix})$$

4 Computational Aspect of KDE

Analyzing individual formulas for KDE and bandwidth selection, has a computational complexity of O(n^2) . Consequently, for large datasets, naive computations are a very bad decision. There is a very elegant and effective method based on Fast Fourier Transform (FFT), which allows a huge computational speedups without a loss of accuracy.

The most well known implementation of FFT based algorithm is known as 'Wand's Algorithm'.This involves the concept of data binning,which can be understood as a kind of data discretization. Binning data is a must-have step before applying the FFT-based technique for fast KDE computation. We do the following: on the basis of n data points $X_1, X_2, ..., X_n$(experimental data), we find a new set of M equally spaced grid points $g_1, g_2, ..., g_M$ with associated grid counts $c_1, c_2, ..., c_M$.s. Grid counts are obtained by assigning certain weights to the grid points, based on neighbouring observations.

Then our density estimate becomes:

$$\hat{f}_j = \hat{f}(g_j, h) = \frac{1}{n}\sum_{i=1}^{n} K_h(g_j - X_i)$$

In practical usage, it is more desirable to compute KDE for equally spaced grid points g_j Using this step our kernel evaluation reduces to O (nM).

So, the next natural step could be to make use of binning, that is for every sample point X_i to be replaced by a pair of two values: the grid point g_i and the grid count c_i. We now use $\tilde{f}$ to represent as an approximation of our estimate $\hat{f}$ and can be written as:

$$\hat{f}_j = \tilde{f}(g_j, h, M) = \frac{1}{n}\sum_{l=1}^{M} K_h(g_j - g_l)c_l j = 1, ..., M$$

The number of kernel evaluations now is O(M^2). It is also halved as the kernel function is symmetric and is only required to be computed once. To reduce this value further to O($Mlog_2 M$), the FFT-based technique is used. The equation is now rewitten as:

$$\begin{aligned}\tilde{f}_j &= \frac{1}{n}\sum_{l=1}^{M} K_h(g_j - g_l)c_l \\ &= \sum_{l=1}^{M} k_{j-l}c_l = c * k\end{aligned}$$

where,

$$k_{j-l} = \frac{1}{n}K_h\frac{(g_M - g_1)}{(M-1)}(j-l)$$

The last operation is called the convolution and this is what helps us in implementing the FFT-based technique. The actual maths to reduce this equation is bit more complex and can be found in the book by Artur Gramacki which I have mentioned in the references below. Fortunately, this FFT-based Wand's algorithm is implemented in-built in various packages (statsmodel KDE in python and standard density function in R).

5 Models

Next I build the models on real dataset using KDE classifier. I am using three common datasets for model building: Iris Dataset, Digit Dataset and Wine quality dataset. I have also compared the performance with other benchmark model like SVM with rbf kernel, simple logistic regression and naive Bayes classfiier, all built with hyperparameter tuning to give the best performance.

5.1 IRIS Dataset

Let us begin by building our model on the simplest of the dataset. This is perhaps the best known dataset to be used for classification. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.It contains two four features (sepal length, sepal width, petal length and petal width). One class is linearly separable from the other 2; the latter are not linearly separable from each other.

Here I have built a model using both the Multivariate KDE classifier and one using the Naive KDE classifier. The latter makes assumption that each of the features are conditionally independent to ease our computation. The model built using both this classifier gives a accuracy of 0.99 and 0.97 respectively (Table 2) which is not alot given the computational cost. And even this outperforms the naive Bayes approach and is comparable to other benchmark models such as SVM and simple logistic regression. Figure 7 shows the bandwidth parameter selection using cross-validation whereas Figure 8 gives us the classification report for iris dataset using KDE classifier.

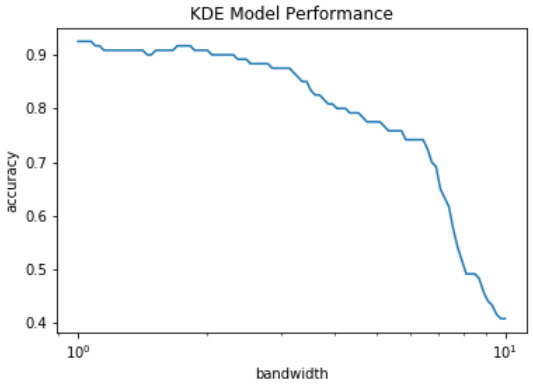

Figure 7: Bandwidth selection for Iris Dataset

Model	Testing Accuracy
Naive KDE Classifier	0.97
Multivariate KDE	0.99
SVM	0.99
Logistic Regression	0.99
NaiveBayes	0.95

Table 2: Model Performance on Iris Dataset

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	0.89	1.00	0.94	8
Iris-virginica	1.00	0.91	0.95	11
accuracy			0.97	30
macro avg	0.96	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Figure 8: Classification Report: KDE classifier on Iris Dataset

5.2 Digit Dataset

Next we use a digit dataset to build our classifier. This dataset is made up of 1797 8x8 images. Each image is of a hand-written digit. In order to utilize an 8x8 figure like this, we first transform it into a feature vector with length 64. Since this involves calculating a 64 dimensional data, building a multivariate KDE classifier is computationally infeasible. So, we resort to building the naive KDE classifier. As we can see from the results (Table 3) that this is able to learn quite well and the model performance is on par with other benchmark classifiers like SVM and even better than the logistic regression. It outperforms the naiveBayes classifier by quite a margin. Figure 9 shows the bandwidth parameter selection using cross-validation whereas Figure 10 gives us the classification report for digit dataset using KDE classifier.

5.3 Wine Quality Dataset

Finally I have built a classifier based on the Wine Quality Dataset. This dataset consist of 12 features and the target variable is a score (0 to 10) which tells us about the quality of wine. This can be considered both as a regression and classification problem. We have here treated it as multi-class classification problem. (NOTE: Here we have considered classes for score 3-9, as the dataset has no instance of score 0-2 and 10)

Building a classification algorithm has on this dataset has own set of challenges. As you can see from the results (Table 4), none of classifier is able to predict the scores accurately. As expected naive Bayes performs the worst and SVM classifier is the most accurate. But KDE classifier is not far behind and is able to outperform the logistic regression model. Figure 11 shows the bandwidth parameter selection using cross-validation for wine quality dataset.

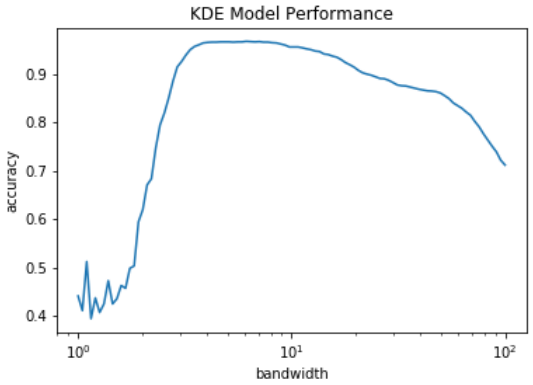

Figure 9: Bandwidth selection for Digit Dataset

Model	Testing Accuracy
KDE Classifier	0.99
SVM	0.99
Logistic Regression	0.96
NaiveBayes	0.81

Table 3: Model Performance on Digit Dataset

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31
1	1.00	1.00	1.00	37
2	1.00	0.97	0.99	38
3	0.98	1.00	0.99	44
4	1.00	1.00	1.00	41
5	0.94	1.00	0.97	31
6	1.00	1.00	1.00	42
7	1.00	1.00	1.00	32
8	0.97	1.00	0.99	33
9	1.00	0.90	0.95	31
accuracy			0.99	360
macro avg	0.99	0.99	0.99	360
weighted avg	0.99	0.99	0.99	360

Figure 10: Classification Report: KDE classifier on Digit Dataset

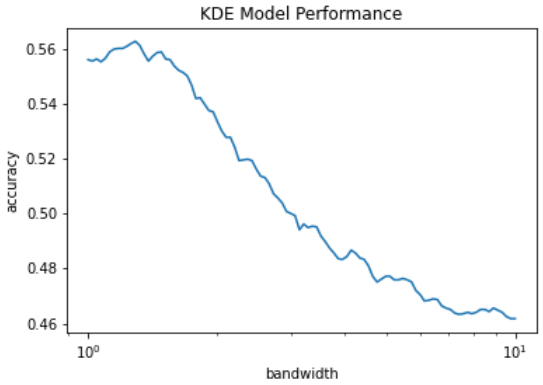

Figure 11: Bandwidth selection for Wine Quality Dataset

Model	Testing Accuracy
KDE Classifier	0.59
SVM	0.62
Logistic Regression	0.50
NaiveBayes	0.43

Table 4: Model Performance on Wine Quality Dataset

6 Conclusion

This is an interesting approach to solving a classification problem It is useful where a parametric model fails to give a meaningful inference.It builds a model without making many prior assumptions. As we can see from the results above, this methodology performs at par with other benchmark and state-of-the-art models such as SVM. It outperforms certain naive approaches such as the vanilla NaiveBayes classifier and in certain cases the logistic regression model. From this study we can see that despite its drawback,using this approach holds certain merit and can be confidently used for predictive modeling.

References

1. Nonparametric Kernel Density Estimation and Its Computational Aspects: Artur Gramacki, Springer.

2. The Elements of Statistical Learning:Trevor Hastie, Robert Tibshirani and Jerome Friedman.

3. Python Data Science Handbook by Jake VanderPlas.