KDE - Kernel Density Estimation
Term Project Research Report

CptS 437

Team: Project 4

Gabriel Gray
Jay Woo
Connor Hill
Julia Messegee

11/19/2024

Abstract

This project explores the application of Kernel Density Estimation (KDE), a non-parametric technique for estimating probability density functions (PDF) without reliance on predefined distributions, making it ideal for analyzing multimodal datasets. Our research aims to enhance the understanding of KDE's principles and practical applications in machine learning, primarily for improving probabilistic models like Bayesian networks by providing more accurate probability estimations. Using datasets such as London weather data and global air and water quality metrics, the team will implement and evaluate KDE algorithms, refining bandwidth parameters and kernel functions to strike a balance between smoothness and detail in density estimates. The project covers data cleaning, algorithm development, data visualization and analysis, before finally culminating in a comparison of kernel functions to identify optimal configurations for real-world data modeling. By focusing on KDE, we seek to explain and contribute to the development of statistical tools decision making in machine learning.

Table of Contents

Introduction

Estimating the distribution of data is important for statistical modeling, since incorrect probability may cause incorrect prediction of models. A problem is how it can be calculated. Since the real world data is collected raw, the volume is finite, and there may be noise in it, so the estimation of the distribution itself may be very difficult.

One of the solutions to find the underlying probabilistic distribution is KDE. KDE is a nonparametric method that estimates the probability density function of a dataset without assuming any specific underlying distribution, so it can be a flexible way to figure the data distribution.

KDE works by placing a kernel-a smoothing function-around each data point and summing their contributions. The bandwidth of the kernel controls the level of smoothing, allowing users to balance detail and generalization in the density estimate. For example, smaller bandwidths capture local features but may overfit to noise, while larger bandwidths smooth out variations but might miss critical details.

The ability to flexibly model distributions without underlying assumptions makes KDE a valuable tool in modern data analysis. It has been applied successfully in various ways, such as anomaly detection, probabilistic modeling, and feature generation for ML algorithms. KDE can complement these machine learning models by improving probability-based components, where accurate density estimation enhances model performance.

The objective of this project is to delve into the theoretical and practical aspects of KDE, exploring its capabilities to improve machine learning models. By applying KDE to diverse datasets, the project will demonstrate its utility in analyzing data distributions.

Kernel Density Estimation Methodology

Suppose a finite real dataset is collected. Each data point would lie on some certain PDF if the data points are not independent from the problem.
With the given dataset, a KDE assumption of a new data point is available.
Let x be a new data point that may be caused by the problem. The probability of getting x by the problem is dependent on the underlying PDF of the problem. Therefore, KDE would estimate the probability of x based on the given dataset.

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)$$

$\hat{f}(x)$ will return the $p(x)$.

How it works

An applicator may choose kernel K and hyper-parameter h. K is usually a gaussian function since many practical problems cause normal distribution or polynomial distribution. Hyper-parameter h works as a smoother. If h is too high, the estimated PDF will not distinguish local density. If h is too low, the estimated PDF is vulnerable to the noise. Basically, h is the variance of each kernel.
With a given data set X, each data point x_n has its own kernel $K(x_n)$ around it, and the breadth of the kernels will be depending on h.

Let $p(y)$ be a probability that the applicator wants to figure.
To get $p(y)$, KDE calculates how y is distanced from each point x. Which means, KDE checks if y is in the right range based on how much each kernel $K(x)$ covers y.

How KDE Forms Distribution

Let Y be an input domain of KDE ranged in finite interval. By giving every single input y in Y, KDE will form the distribution with $f(Y)$.
Even if the domain of Y is infinite, $f(Y)$ can be reduced into finite range when the dataset X is finite. There will be a certain point a that for any $y<a$, $f(y)$=0, and b that for any $y>b$, $f(y)$=0. Which means, $f(y)$ approaches zero for y far outside the data range, and those y will be null.

Compare to Classifier

KDE can be a classifier.
Let q be a support vector machine (SVM). X and Y are datasets that were used for the learning, where X is input in R and Y is output in (0,1).
Given x' is an input for the prediction. $q(x')$ will return 0 or 1.

Two KDEs trained by the data X and Y will have two PDFs over X. Let those KDE be A and B. A is trained over $X|Y$=0, and B is trained over $X|Y$=1.
If $q(x')$=0, the $p(x')$ given by $\mathrm{A}(x')$ will be higher than $p(x')$ given by $\mathrm{B}(x')$. Else, $\mathrm{B}(x')$ will return a higher probability.
Therefore, KDE can work as a classifier for problems, and it can cover multi-classification as well.

Input Range Null

However, since KDE is based on the PDF, it would not be able to classify the input domain that goes to null. Which means, since the dataset is finite, it will have certain inputs that can not be decided.
On the other hand, SVM is based on a linear determining function, so it has the ability to answer the input that is far away from the trained inputs.

Also, since it has a problem of degrading performance with high-dimension data, and the requirement of more time consumption for the bigger size of dataset, it would be better used as a supplementary machine.

How to Choose the Best Bandwidth (h)?

The choice of the bandwidth parameter h is one of the most defining aspects of KDE. The bandwidth controls the smoothness of the estimated probability density function, and balances bias (under-smoothing) and variance (over-smoothing). Selecting the optimal h is crucial for ensuring that KDE represents the underlying data distribution effectively.

Since h is a hyper-parameter, there are many approaches to determine its value. One of them is adjusting h through epochs. There are loss functions to determine how much loss can be reduced for each h. Through the epoch, the applicator can find the best h that has minimum loss. Which is just the same method for the other machine learning methods.

Mean Integrated Square Error

MISE is one of the loss functions that can evaluate the performance of h.

$$\mathrm{E}\,\|f_n - f\|_2^2 = \mathrm{E}\int (f_n(x) - f(x))^2\,dx$$

For each epoch, KDE will produce a distribution with a given h, which is $f\_n(x)$.
The produced distribution will be compared with the “true distribution”, and the difference will be summed up. MISE loss will indicate how much the produced distribution is different from the true distribution.

One problem is that there is no way to estimate the true distribution since the data is collected raw. To solve this problem, plug-in selection can be used.

Plug-In Selector

Let L be a loss calculated with MISE. In each epoch, L can be minimized by adjusting h with $(d/dL)*h\rightarrow 0$. Which means, every epoch will produce $f_0(x)$ such that L becomes minimum.
Now, the produced PDF $f_0(x)$ can be used to form a true distribution for the next epoch. Though it is not the real true PDF , it is closer to the local minimum of loss, so the next d/dL will be also in the direction toward the local maximum. Therefore, the PDF will eventually reach the local minimum.

Problem Regarding KDE

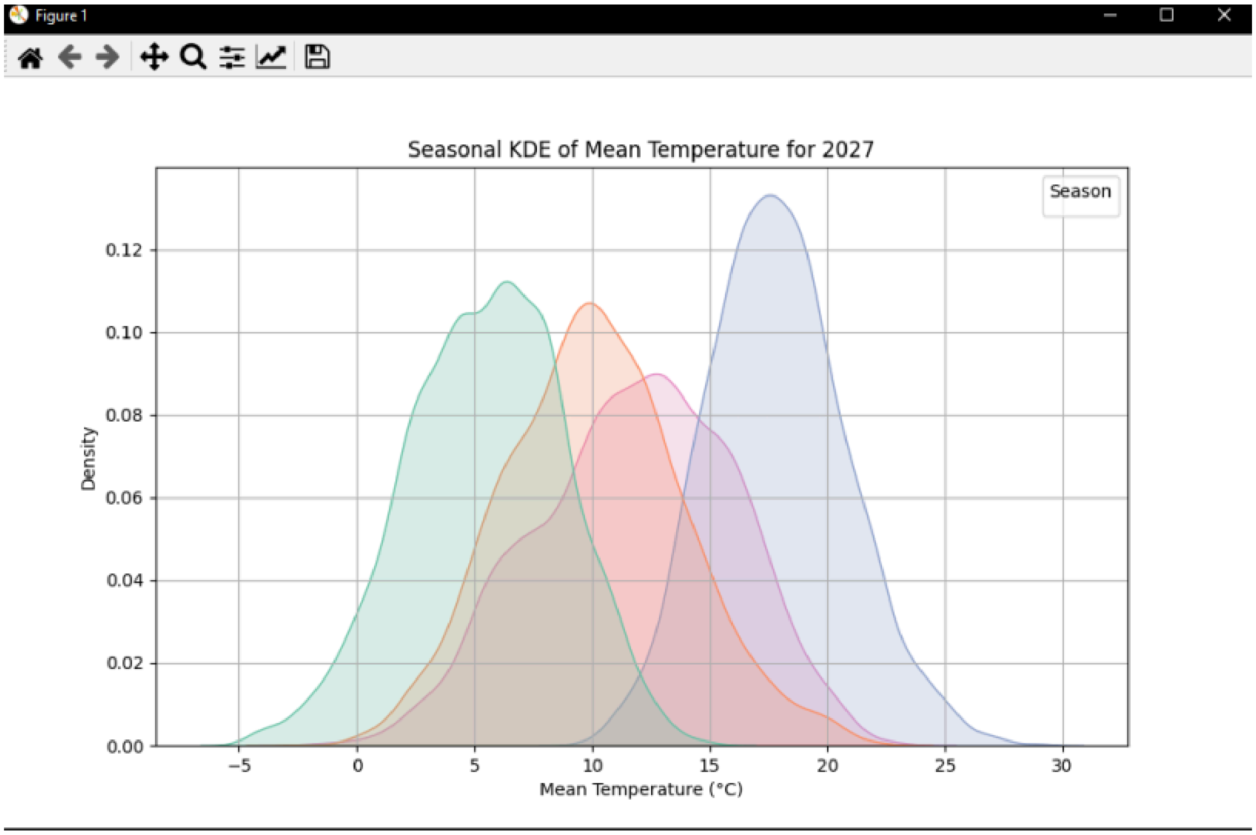
The problem with KDE is that it does not guarantee searching the global minimum of loss. As described in the Plug-In Selector section, KDE will only produce a reasonable PDF but no real true PDF. When an epoch reaches the local minimum, the direction will be 0, so the loss of the final PDF will be stuck in the local minimum.
Therefore, KDE is - like many other ML methods - not a solution, but it is only a method. Which means, the performance of KDE will be depending on the use of the applicator and some randomness.

Application

Below is a link to a simple application utilizing KDE.
https://colab.research.google.com/drive/1zb27UlKucgJxrNX0YMUp_GCpSQtaZ9M_?usp=sharing

This code defines a simple weather prediction and analysis system using a ML model (random forest) and Kernel Density Estimation (KDE) techniques to handle historical weather data for London for the years 1979-2021. It incorporates data cleaning, predictive modeling, and synthetic data generation for a broader scope of viewing KDE’s applications.

The code utilizes KDE to estimate the probability density of temperature distributions for different seasons, allowing for the creation of synthetic weather datasets. Through the analysis of seasonal distribution of temperatures for this region with KDE, the model simulates realistic values for particular years. This is accomplished by sampling temperature values based on the probability density curves created by KDE, ensuring predicted data aligns with statistical properties of the London weather dataset. Key features like sunshine and snow depth are also preserved and utilized during generation.



Conclusion

Though KDE is not a solution that finds the true PDF underlying the problems, it can give visual and intuitive distribution of data. With the PDF given by KDE, the applicator may design a statistical prediction model with better performance. For example, if the data distribution is binomial, implementing an adversarial model may be better in performance, and the distribution is what KDE can illuminate.

Also, KDE can even be a great tool to evaluate the performance of ML models. Suppose there are several ML models for a specific problem. The user can feed the models with several test data sets, and get their precision scores as data points. With the precision scores, the user can form their performance distribution over test data sets to decide which model has the best performance.

Thus, KDE may not be a good stand-alone method, but it is a powerful tool to upgrade ML models.

Links and Demo Steps

Follow the link to our Google Colab for this project and clone it to your own Drive:
https://colab.research.google.com/drive/1zb27UlKucgJxrNX0YMUp_GCpSQtaZ9M_?usp=sharing

Before running, follow this link to download the necessary CSV file for the Historical London weather data from 1979 to 2021:
https://github.com/Nonchillant/TeamProject4-KDE/blob/main/london_weather.csv

Use the files menu on the left of your cloned colab to upload the dataset to the session storage.

You are now ready to run the colab notebook and follow the input prompts.

Link to presentation slides for this project:
https://docs.google.com/presentation/d/1njBGCJJeg4T0Kv_nSf1Ph1UScBQr4iF4kmwfzKOfTMc/edit?usp=sharing

Link to project repository:
https://github.com/Nonchillant/TeamProject4-KDE

Link to project colab notebook:
https://colab.research.google.com/drive/1zb27UlKucgJxrNX0YMUp_GCpSQtaZ9M_?usp=sharing

Works Cited

García-Portugués, E. (n.d.). *2.4 Bandwidth selection | Notes for Nonparametric Statistics*. https://bookdown.org/egarpor/NP-UC3M/kde-i-bwd.html

London Weather Data. (2022, May 16). Kaggle. https://www.kaggle.com/datasets/emmanuelfwerr/london-weather-data

ritvikmath. (2024, January 29). *Kernel Density Estimation : Data Science Concepts* [Video]. YouTube. https://www.youtube.com/watch?v=t1PEhjyzxLA

VanderPlas, J. (n.d.). *In-Depth: Kernel Density Estimation | Python Data Science Handbook*. https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html

Węglarczyk, S. (2018). Kernel density estimation and its application. In *ITM web of conferences* (Vol. 23, p. 00037). EDP Sciences.