**Estimating Probability Density using a KDE**

Kernel Density Estimation is an alternative method of estimating the underlying PDF of a dataset.

A person looking at a math problem

Description automatically generated

We apply a kernel to each data point (imagine a rug plot and applying PDF to each stick) and sum them together. We then divide by the number of datapoints to normalise the KDE (this ensures that the total area under the KDE (i.e. integral) is 1 so that the KDE can represent a legitimate probability distribution).

**Explanation of the KDE formula:**

* **Xj** = an observation. The kernel is centred on this value
* S = the point at which we are estimating the probability density
* K(x) = this weighs the contribution of the observation to the overall probability density estimate. Larger bandwith means the observation has less contribution to overall PDF
* (s-xj/w) - this calculates the normalised distance of the observed value from the point where we are estimating the probability density. The output of this is used as the input for the Kernel function.
* W = the bandwith. This is the spread of the kernel. For a Uniform function, this would be the width of each uniform kernel.

f (S): This represents the estimated density function at the point s. It gives us an 
1. 
estimate of the probability density of the data at this specific point. 
1 
2. 
: This is a normalizing constant which ensures that the KDE integrates to 1 over the 
entire space, which is a requirement for any probability density function. Since we 
sum over 'n' data points, we divide by 'n' to average their contributions. 
E n •This is the summation notation indicating that we will sum the contributions of 
3. 
each data point to the density estimate at s. 
1 
: Another normalizing factor that is inversely proportional to the bandwidth 'w'. This 
4. 
w 
ensures that the contribution of each data point is scaled appropriately according to 
the bandwidth. 
: This is the kernel function. It weighs the contributions of each data point 
5. 
Tj to the density at point 's'. The kernel is typically a symmetric, positive function that 
integrates to 1, ensuring that it contributes a "bump" of area 1 to the density estimate. 
Common choices for K are Gaussian, Epanechnikov, and uniform kernels. 
6. 
s — Tj: This is the difference between the evaluation point 's' and a data point cj. It 
measures how far the data point is from the location where we are estimating the 
density. 
w: This is the bandwidth, which determines the width of the bumps created by the 
7. 
kernel function. A small bandwidth leads to a bumpier estimate, whereas a large 
bandwidth leads to a smoother estimate. 
, cn}: These are the observed data points used to estimate the density 
8. 
function. 

K(x) = a kernel function. Common kernel functions are Gaussian and Uniform.

Here is the form of the three kernels: 
Gaussian 
Uniform 
Epanechnikov 
KG) = 
KG) = 
KG) = 
1 
27T 
1 
2 
3 
max{l — $2, O}. 
4 

The kernel function K(x) provides the weighting for each datapoints contribution to the overall probability density estimate, based on the standardised distance of each datapoint from the point where your estimating the density (i.e. (s-xj/w)) --> this is the input for kernel function

* In a Gaussian distribution the closer the data point to the point at which you are estimating the probability density, the greater the input value into the normal kernel, thus the more weight the normal kernel will give it.
* In a uniform kernel, each datapoint that is within 1 unit distance from s will be given the same weight of 0.5.

**How to calculate a KDE using each kernel**

A smiley face with plus and a plus

Description automatically generated

Its important to note that the (s-xj/w) part of the function is not multiplied by the Kernel function but instead provides the input for it. That is why it is in brackets after the K i.e. K(X) - the input is placed within the brackets next to K to indicate it’s the input for it.

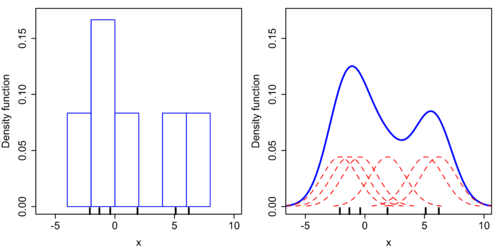
No, we do not multiply the output of the kernel function by (s 
use this value as the argument to the kernel function. 
The formula for the kernel density estimate is: 
Here's what happens step by step: 
w. Instead, we 
For each data point Ci, you calculate (s — Ci)/w. This is the standardized distance of 
1. 
the point s from each data point Xi, scaled by the bandwidth w. 
You then plug this standardized distance into the kernel function K. For a uniform 
2. 
1 
kernel, the function is K (u) — 
for lul land K (u) 0 for lul > 1. 
The kernel function K outputs a weight for each data point based on this 
3. 
standardized distance. If the distance is within the kernel's support (which for a 
uniform kernel is between -1 and 1), it contributes a weight of I ; otherwise, it 
contributes a weight of O. 
4. 
You sum up all the weights contributed by each data point. 
You divide this sum by the number of data points n and by the bandwidth w to get the 
5. 
kernel density estimate at point s. 
The multiplication by (s — ct•) w is part of the process of calculating the argument 
for the kernel function, not a multiplication applied to the output of the kernel function. 

**Specific Example of calculating a KDE using a uniform kernel:**

Uniform K(c) 
2 

* **For the uniform kernel to provide weight to an observation (s-xj/w) must be between -1 and 1.**

Here is a detailed explanation of the calculation: 
Given the data points Xl, c2, ..., which are —2, —1, 0, 1, 2, and our point of 
interest s 0, the bandwidth w 1, and using the uniform kernel function K (u) 
1 
- for lul 1, we calculate the kernel density estimate f (s) as follows: 
2 
The uniform kernel contributes equally within the window defined by the bandwidth. 
For each data point Ci, we calculate (s — Ci) wand apply the uniform kernel 
function: 
—2: (0 — (—2))/ 1 2. Since this is outside the range [—1, 1], the kernel 
1. 
For 
function contributes O. 
1. This is on the boundary of the uniform kernel's 
2. 
For 
1 
support, so the kernel function contributes 
For 0: (0 — 0)/ 1 0. This is within the kernel's support, so the kernel function 
3. 
1 
contributes 
— 1: (o - 1)/1 
—1. This is also on the boundary of the kernel's support, so 
4. For zu — 
1 
the kernel function contributes 
2: (o - 2)/1 
—2. Since this is outside the range [—1, 1], the kernel 
5. 
For 
function contributes O. 
We then sum up these contributions and divide by the number of data points n — 5 
and by the bandwidth w 1 to get the final density estimate: 
1 
This is the estimated density at point s 0 using the uniform kernel. C>-) 



**Example 2: summing up the kernels to get the KDE**

A person looking at a graph

Description automatically generated

**Things to note:**

The K(x) represents the kernel value of a specific datapoint (i.e. a tiny gaussian PDF around each datapoint).

* The mean of the kernel is the datapoint for which the kernel has been placed on top of. The kernel is centred on the observed value xj

**Calculating the Gaussian kernel for a single value:**

Summing the kernel’s for all of the datapoints together and dividing by the bandwidth and sample size.

A smiley face with plus and a plus

Description automatically generated

The ‘w’ (or ‘h’) smoothening parameters = the bandwidth of the kernel. The bandwidth of the kernel represents its spread. The spread for each kernel is the same.

* The larger the bandwidth, the smoother the KDE. This increase the bias of the model (remember, the smoother a PDF, the more likely it is to underfit and thus shows bias). Larger bandwidth reduces the weight given to each observations
* The smaller the bandwidth, the more jagged the KDE. This increases the spread of the model (remember, the more complex a PDF, the more likely it is to overfit and thus show variance). Smaller bandwidth increases the weight given to each observation. The increased sensitivty can lead to more fluctuations, increasing the chance the and KDE may seem multimodal.
* **Undersmoothing =** bandwidth too small. Our PDF is too jagged and thus captures the randomness of the data too much
* **Over smoothing =**  the bandwidth is too big. Our PDF is too smooth leading which ends up hiding the real shape of the PDf of the dataset.
* Choosing the optimal bandwidth is very important for the accuracy of your KDE.

**Choosing the optimal bandwidth parameter:**

We can use the maximum likelihood cross validation technique to find the optimal bandwidth parameter for our dataset:

<https://medium.com/analytics-vidhya/kernel-density-estimation-kernel-construction-and-bandwidth-optimization-using-maximum-b1dfce127073>

**Convergence of the KDE to the true PDF of the population:**

As the sample size increases, the KDE increasingly converges with the true PDF of the population from which the sample is drawn. This can mean that for small sample size the KDE may not be a useful means for estimating the true PDF of the population. **TLDR –** larger sample size increases the accuracy of the KDE estimate.

A person in a video conference

Description automatically generated

The rate at which the KDE converges with the true PDF of the population refers to how quick the convergence happens as the sample size increases. The rate of convergence is impacted mainly by the bandwidth of the KDE, the sample size and the smoothness of the true PDF of the population:

The relationship between the KDE and the sample size is represented by:

A person in a video conference

Description automatically generated

This formula says that ‘w’ (bandwidth) is **proportional (**∝**)** to the reciprocal of the N1/5. It means that as the sample size N increases, the KDE convergences to the true PDF at the rate of N4/5.

* Generally, as the sample size increases, the bandwidth required for convergence between KDE and the true PDF decreases, but at a decreasing rate. This ensures that the KDE is more accurate as sample size increases.

The correct bandwidth required for KDE is calculated during cross-validation.

**Choice of kernel**

The choice of the kernel will have a significant impact on the accuracy of the KDE estimate.

A screenshot of a computer

Description automatically generated

Here is the form of the three kernels: 
Gaussian 
Uniform 
Epanechnikov 
KG) = 
KG) = 
KG) = 
1 
27T 
1 
2 
3 
max{l — $2, O}. 
4 

**The smoothness of the kernel affects the smoothness of the KDE**

A Gaussian kernel is a smooth distribution and so will increase the amount of smoothness in the KDE – this decreases variance but increases bias.

A triangular Kernel is a more jagged kernel and thus will decrease the amount smoothness in the KDE – this will decrease the bias but increase the variance.

**Different kernels will yield different shaped KDE’s for the same dataset:**

In the graph below, the triangular kernel provides a KDE with a much higher peak than the Gaussian kernel. This is because the triangular kernel is much less smooth, which means the KDE it creates is always less smooth. We can counter this by using a larger bandwidth parameter.

A person looking at a graph

Description automatically generated