Density Estimation has vast application, such as in understanding the structure of the multivariate data, finding outliers, estimating the likelihood of seeing a datapoint in the data and many more.

First, we understand the Discrete Data Density Estimation. In case of multivariate categorical data, a column having three color categories viz Red, Blue and Green. Then the sum of individual color density should be 1. That is,<br>

$P(R)+P(B)+P(G)=1$

<br>

Here we are learning three parameters that we need to estimate from the data, but the number of free parameters = Number of parameters - Constraint ( probability adds up to 1) = 2. In the mentioned case, we are estimating the density along each column i.e. one-dimensional density estimation.

If we want to estimate joint estimation, for example, probability of red colored and triangular toy, then the probability would be:<br>

$P(R, T)=\frac{n(R, T)}{\N}$

<br>

where N = total number of toys and n(R,T) = count of common observations where toy is red and triangular

<br>

In the joint space, how many parameters are we learning?<br>

Let’s say that column “Color” takes 3 unique values and column “Type” takes 3 different values. Then the combined view has 9 cells i.e., number of parameters to learn= 9 and there is one constraint that sum of all possible groups is 1. So, we have 8 free parameters. <br><br>

These are called Counting-based Density Estimation Functions.

One important point to note:

If we increase the number of datapoints, counts or number of observations increases per class, but the parameters(probability) of seeing a possible combination can still remain same. <br>

![image]([https://raw.githubusercontent.com/niteshjindal170988/unsupervised-learning/main/density-estimation /.scrap/1.JPG](https://raw.githubusercontent.com/niteshjindal170988/unsupervised-learning/main/density-estimation%20/.scrap/1.JPG))

With more data, we get more confidence in the data however parameter value remains unchanged. As the data size increases, we tend to get more biased outcomes unlike prior estimates (i.e., unbiased estimates or equal probability estimates to each class).

<br>

# Non Parameteric Density Estimation

<br>

Assume that each data point has a field of influence(i.e., a gaussian) and as we move away from the datapoint, the field of influence decreases. To calculate the density of any new point, we compute the distance of each datapoints from the new point which will provide us with the overall influence of all the points on the new point.Each gaussian has mean and variance and each gaussian sits on each datapoint.

<br>

Let $K\_{\sigma}\left(x, x\_{n}\right)$ be the Kernel function where $x\_{n}$ is the nth datapoint or the mean of the gaussian around nth datapoint. The bigger is distance between nth datapoint and datapoint “x”, smaller would be the overall influence i.e. smaller kernel function value.

<br>

The density of the point ‘x’ (i.e., the influence of all the datapoints on the datapoint ‘x’ ) is:<br>

$P(x)=\frac{1}{N} \sum\_{n=1}^{N} K\_{\sigma}\left(x, x\_{n}\right)$

<br><br> where <br>

$K\_{\sigma}\left(x, x\_{n}\right)=\frac{1}{\sigma \sqrt{2 \pi}} \exp \left(-\frac{\left(x-x\_{n}\right)^{2}}{2 \sigma^{2}}\right)$

<br><br> This is also known as Kernel Density Function or Parzen Window Function. Few points to know are:<br>

* In this process, we are not learning any parameters and just storing the datapoints. When a new datapoint comes, we estimate the probability of the new datapoint by using the given kernal function. That’s why this is called non-parameteric methods.
* Only the scoring time is involved, but no training takes places therefore, there is no training time associated with non-parameteric approach.
* $/sigma$ is the hyperparameter and there are no parameters involved.

# Parameteric Density Estimation