CSE/ECE-471: Statistical Methods in AI **Spring 2020**

Non Parametric Density Estimation

Prepared by: Ankush Mitra,Amit Jindal,Mowlancia Billa

1 Probability Density Function

The probability density function (pdf) is a fundamental concept in statistics. Given the pdf f(x) of a random variable X, probabilities associated with X can be easily computed as

$$P(a \leq X < b) = \int_a^b f(x)dx.$$

Many problems in statistics can be described as using a (random) sample to infer the underlying (unknown) population. Given a sample x_1 , . . . , x_n - f , the problem of density estimation is to contruct an estimate $\hat{\text{f}}$ of f based on the sample.

2 Parametric vs. Nonparametric

Density estimation methods can be roughly categorized into parametric density estimation and nonparametric density estimation.

In parametric density estimation, f is assumed to be a member of a parametric family (such as normal with unknown mean and variance), and the density estimation problem is then transformed into a simpler one: find estimates of mean and variance and plug into the normal density formula.

In nonparametric density estimation, we do not restrict the form of f with any parametric assumptions (We still need some smoothness assumptions in order to analyze theoretical properties). It is not easy to give a clear definition of what is "nonparametric", but sometimes it is useful to think nonparametric as "infinite parametric".

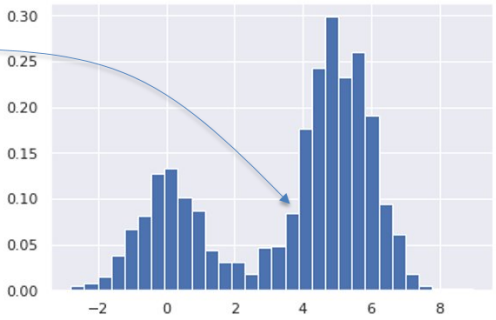
3 Histogram Density estimator

A density estimator is an algorithm which seeks to model the probability distribution that generated a dataset. For one dimensional data, we are probably already familiar with one simple density estimator: the histogram.

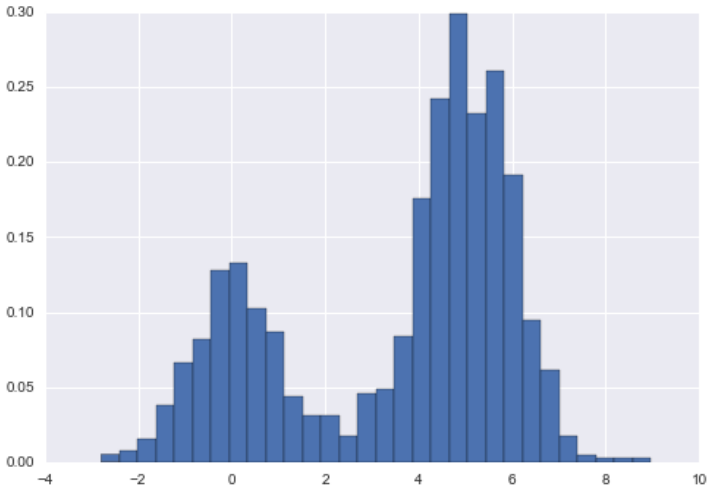
A histogram divides the data into discrete bins, counts the number of points that fall in each bin, and then visualizes the results in an intuitive manner. Histogram is probably the oldest and simplest density estimator.

Probability that a random sample x will fall in this bin

$$p_H(x) = \frac{1}{N}\frac{[\#\ of\ x^{(k}\ in\ same\ bin\ as\ x]}{[width\ of\ bin]}$$



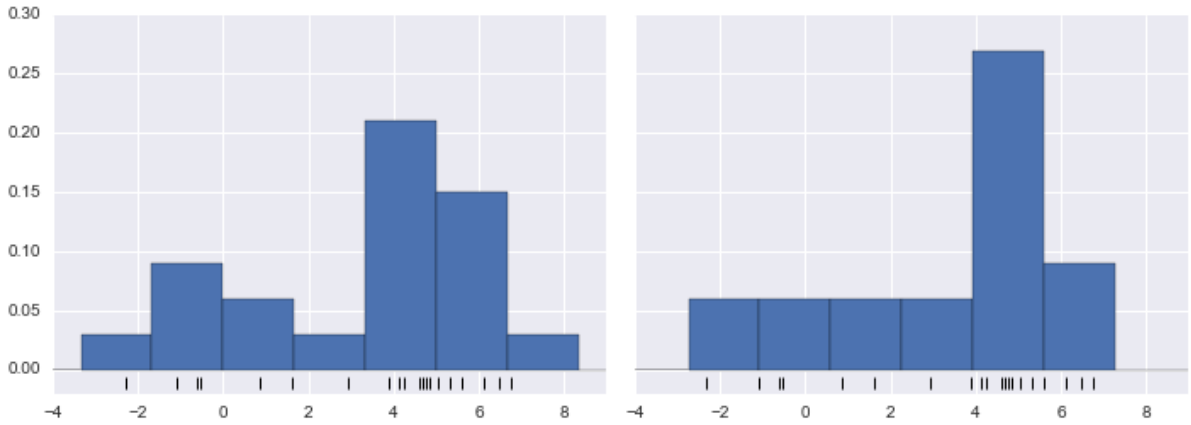
For example, let's create some data that is drawn from two normal distributions: We have a normalized histogram where the height of the bins does not reflect counts, but instead reflects probability density:



Notice that for equal binning, this normalization simply changes the scale on the y-axis, leaving the relative heights essentially the same as in a histogram built from counts. This normalization is chosen so that the total area under the histogram is equal to 1.

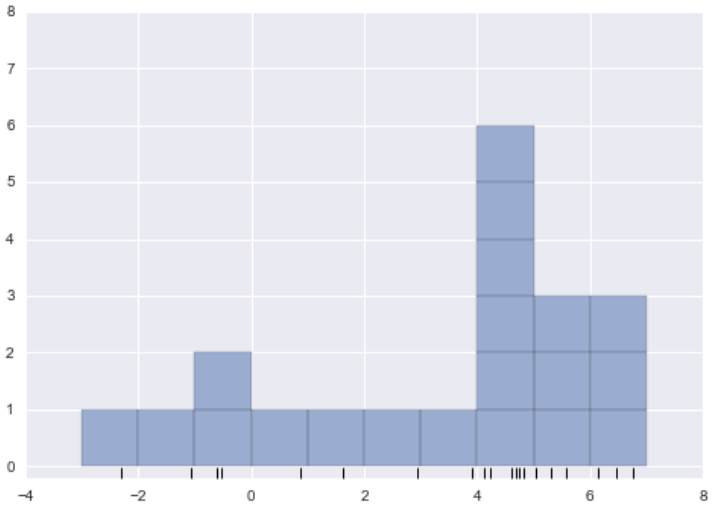
4 Issues with histogram and their solution

One of the issues with using a histogram as a density estimator is that the choice of bin size and location can lead to representations that have qualitatively different features. For example, if we look at a version of this data with only 20 points, the choice of how to draw the bins can lead to an entirely different interpretation of the data! Consider this example:

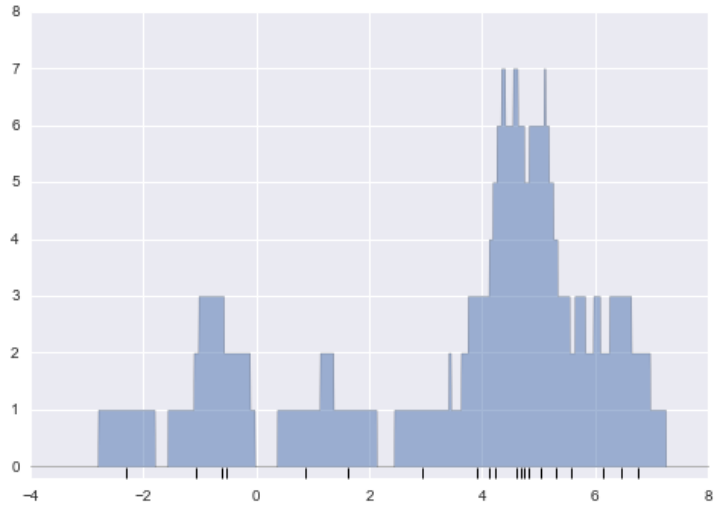


On the left, the histogram makes clear that this is a bimodal distribution. On the right, we see a unimodal distribution with a long tail. These two histograms were built from the same data: with that in mind, how can you trust the intuition that histograms confer? And how might we improve on this?

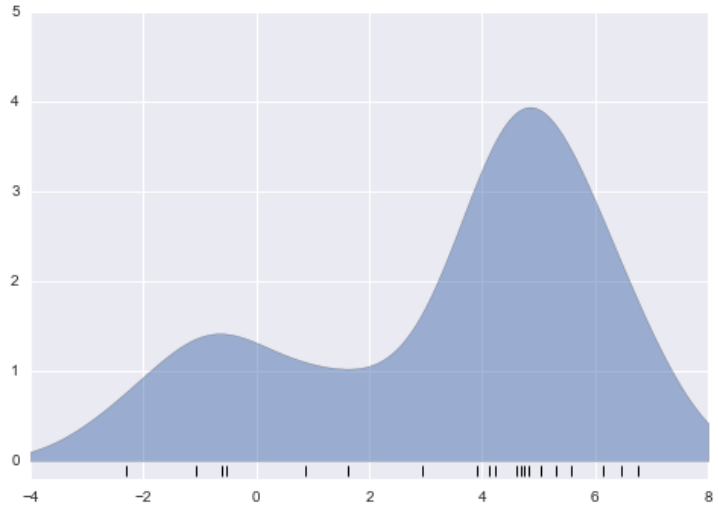
Stepping back, we can think of a histogram as a stack of blocks, where we stack one block within each bin on top of each point in the dataset. Let's view this directly:



The problem with our two binnings stems from the fact that the height of the block stack often reflects not on the actual density of points nearby, but on coincidences of how the bins align with the data points. This mis-alignment between points and their blocks is a potential cause of the poor histogram results seen here. But what if, instead of stacking the blocks aligned with the bins, we were to stack the blocks aligned with the points they represent? If we do this, the blocks won't be aligned, but we can add their contributions at each location along the x-axis to find the result. Let's try this:



The result looks a bit messy, but is a much more robust reflection of the actual data characteristics than is the standard histogram. Still, the rough edges are not aesthetically pleasing, nor are they reflective of any true properties of the data. In order to smooth them out, we might decide to replace the blocks at each location with a smooth function, like a Gaussian. Let's use a standard normal curve at each point instead of a block:



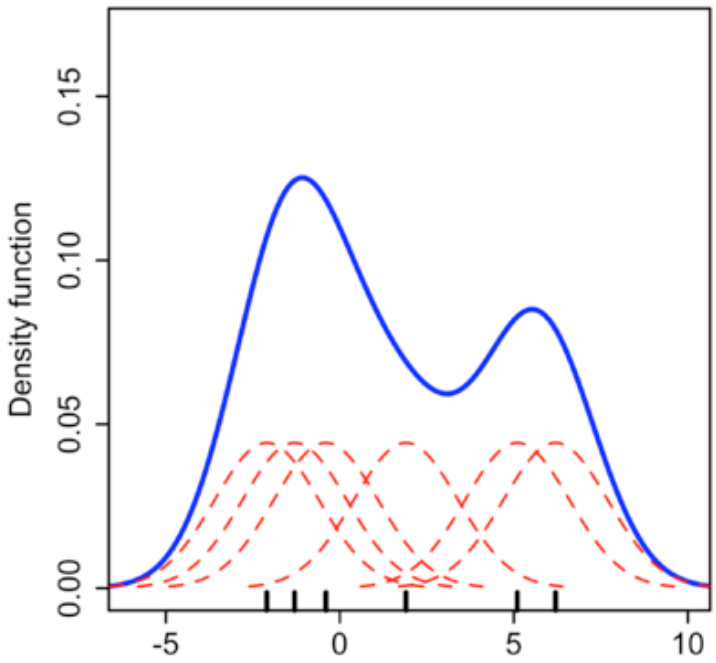
This smoothed-out plot, with a Gaussian distribution contributed at the location of each input point, gives a much more accurate idea of the shape of the data distribution, and one which has much less variance (i.e., changes much less in response to differences in sampling).

These last two plots are examples of kernel density estimation in one dimension: the first uses a so-called "tophat" kernel and the second uses a Gaussian kernel. We'll now look at kernel density estimation in more detail.

5 Kernel Density Estimation

The most common form of estimation is known as kernel density estimation. In this method, a continuous curve (the kernel) is drawn at every individual data point and all of these curves are then added together

to make a single smooth density estimation. The kernel most often used is a Gaussian (which produces a Gaussian bell curve at each data point). Lets look at the following plot

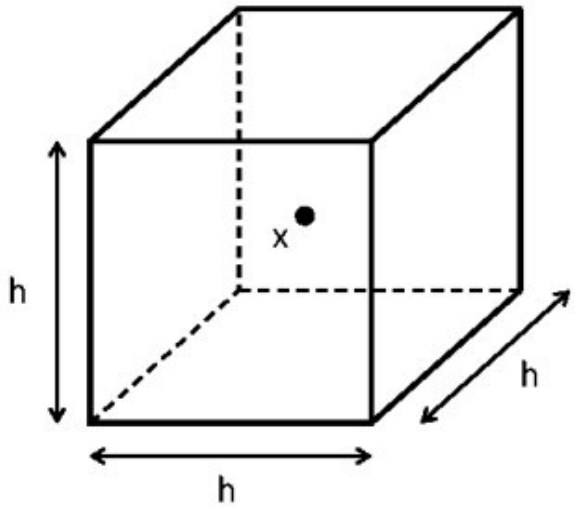


Here, each small black vertical line on the x-axis represents a data point. The individual kernels (Gaussians in this example) are shown drawn in dashed red lines above each point. The solid blue curve is created by summing the individual Gaussians and forms the overall density plot.

The x-axis is the value of the variable just like in a histogram, but what exactly does the y-axis represent? The y-axis in a density plot is the probability density function for the kernel density estimation. However, we need to be careful to specify this is a probability density and not a probability. The difference is the probability density is the probability per unit on the x-axis. To convert to an actual probability, we need to find the area under the curve for a specific interval on the x-axis. Somewhat confusingly, because this is a probability density and not a probability, the y-axis can take values greater than one. The only requirement of the density plot is that the total area under the curve integrates to one.

$$\int K\left(\frac{x-x^{(2)}}{h}\right) = 1$$

6 Parzen Window Density Estimation



The histogram approach to nonparametric estimation inspires the more general,tunable Parzen window method. This approach is defined by a kernel function g(u) satisfying

$$g(u) \geq 0, \quad \int g(u)du = 1.$$

In other words, g(u) satisfies the properties of a probability density function. One example of a kernel for a D-dimensional sample space is

$$g(u) = \begin{cases} 1 & |u_i| \leq \frac{1}{2} \text{ for all } i \in \{1, \ldots, D\} \\ 0 & \text{otherwise.} \end{cases}$$

The kernel g(u) is 1 whenever u falls within the D-dimensional unit hypercube centered about the origin. If we have a sample x1,...,xN, then

$$K = \sum_{i=1}^{N} g\left(\frac{x - x_i}{h}\right)$$

is the number of sample points falling within the the hypercube with side length h centered at the point x.

If we have some region R about a point x in our sample space, the density at x can be approximated by

$$p(x) \approx \frac{K}{NV},$$

where K is the number of examples (from some training data set) in R, N is thetotal number of examples, and V is the volume of the region R

Substituting the value of K and other variables into above equation we obtain the following estimate for the density at x

$$p(x) = \frac{1}{Nh^D} \sum_{i=1}^{N} g\left(\frac{x - x_i}{h}\right),$$

where the volume of the hypercube h^D has replaced V

Note that the right-hand side of above equation will always integrate to one. In other words, the hypercube kernel is just one of many possible choices for g(u). This equation is general and holds for any kernel function.

7 Smooth Kernels

The parzen window has several drawbacks

1. It yields density estimates that have discontinuities
2. It weights equally all points x_i , regardless of their distance to the estimation point x

For these reasons, the Parzen window is commonly replaced with a smooth kernel function K(u)

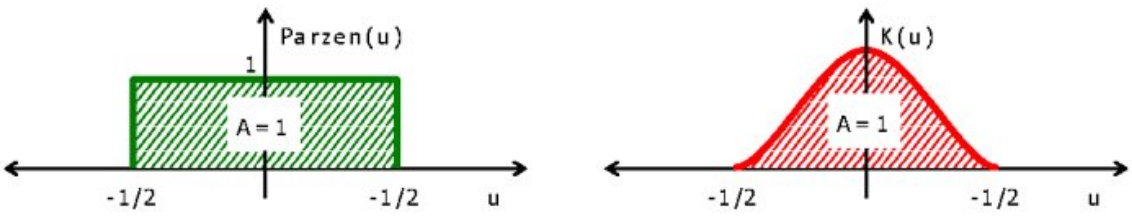
$$\int_{R^D} K(x)dx = 1$$

Usually, but not always, K(u) will be radially symmetric and unimodal pdf, such as the Gaussian

$$K(x) = (2\pi)^{-D/2} e^{-\frac{1}{2}x^T x}$$

Which leads to the density estimate

$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(k}}{h}\right)$$

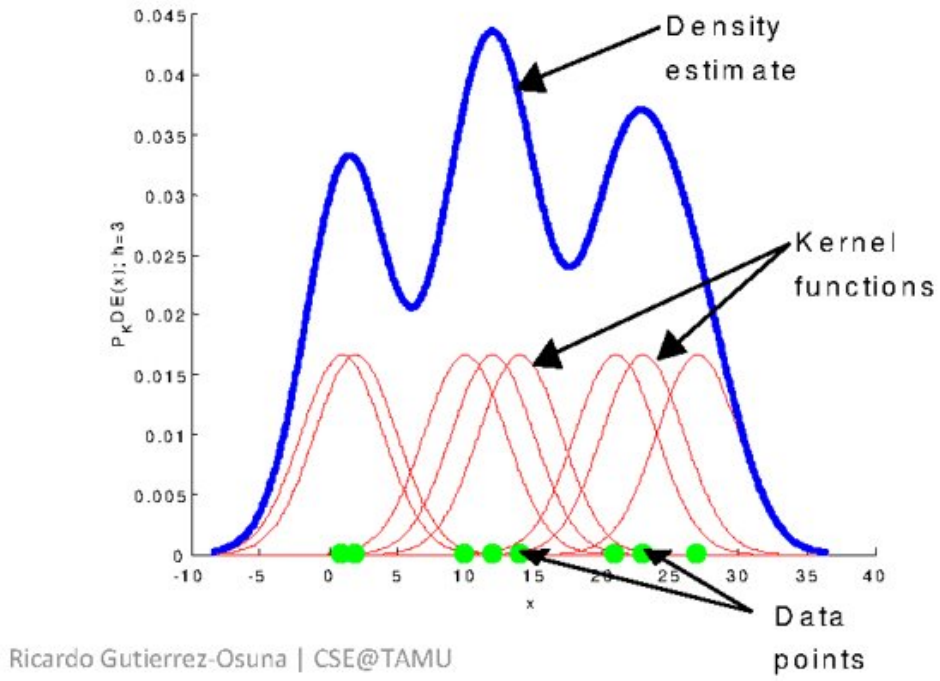


7.1 Interpretation

Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of "bumps".

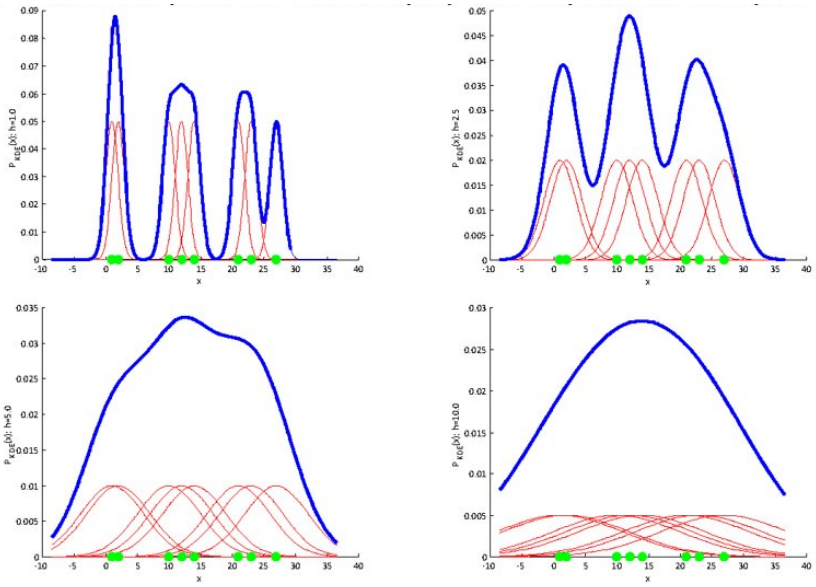
The kernel function determines the shape of the bumps.

The parameter h, also called the smoothing parameter or bandwidth determines their width.



8 Bandwidth Selection

The problem of choosing h is crucial in density estimation

1. A large h will over- smooth the DE and mask the structure of the data.
2. A small h will yield a DE that is spiky and very hard to interpre.t
3. 

9 Maximum likelihood cross validation

The ML estimate of h is degenerate since it yields h_{ML}=0, a density estimate with Dirac delta function at each training data point.

A practical alternative is to maximize the 'pseudo-likelihood' computed using leave one out cross validation

$$h^* = \arg\max\left\{\frac{1}{N}\sum_{n=1}^{N} log p_{-n}\left(x^{(n}\right)\right\}$$

$$where\ p_{-n}\left(x^{(n}\right) = \frac{1}{(N-1)h}\sum_{\substack{m=1 \\ m\neq n}}^{N} K\left(\frac{x^{(n} - x^{(m}}{h}\right)$$

10 Multivariate Density estimation

For the multivariate case, the KDE is

$$p_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

Notice that the bandwidth h is the same for all the axes, so this density estimate will be weigh all the axis equally.

If one or several of the features has larger spread than the others, we should use a vector of smoothing parameters or even a full co-variance matrix, which complicates the procedure.

There are two basic alternatives to solve the scaling problem without having to use a more general KDE

1. Pre-scaling each axis (normalize to unit variance, for instance).
2. Pre-whitening the data (linearly transform so Sigma = I) estimate the density and then transform back

11 Product Kernels

A good alternative for multivariate KDE is the product kernel

$$p_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(x, x^{(n}, h_1, \ldots h_D\right)$$

$$where\ K\left(x, x^{(n}, h_1, \ldots h_D\right) = \frac{1}{h_1 \ldots h_D}\prod_{d=1}^{D} K_d\left(\frac{x_d - x_d^{(n}}{h_d}\right)$$

The product kernel consists of the product of one dimensional kernels (Typically the same kernel function is used in each dimension (K_d(x)=K(x)), and only the bandwidth are allowed to differ.

Note that although K(x,$x^{(n)}$,h_1,...h_D) use kernel independence does not imply we assume the features are independent.

If we assumed feature independence, the DE would have the expression

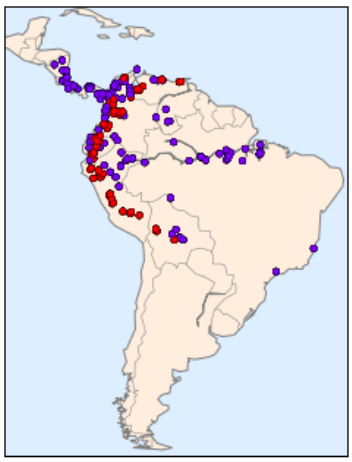
$$p_{FEAT-IND}(x) = \prod_{d=1}^{D} \frac{1}{Nh^D} \sum_{i=1}^{N} K_d \left(\frac{x_d - x_d^{(n}}{h_d} \right)$$

12 Using KDE for Visualization

For 1-D data, we use one simple density estimator: the histogram - One of the issues with using a histogram as a density estimator is that the choice of bin size and location can lead to representations that have qualitatively different features. For example, if we look at a version of data with only 20 points, the choice of how to draw the bins can lead to an entirely different interpretation of the data!.

The choice of bandwidth within KDE is extremely important to finding a suitable density estimate, and is the knob that controls the bias–variance trade-off in the estimate of density: too narrow a bandwidth leads to a high-variance estimate (i.e., over-fitting), where the presence or absence of a single point makes a large difference. Too wide a bandwidth leads to a high-bias estimate (i.e., under-fitting) where the structure in the data is washed out by the wide kernel. We can find a better choice of bandwidth via cross – validation approach.

KDE is built in and automatically used to help visualize points in one and two dimensions. On sophisticated data such as Geographical data where the distributions recorded are the observations of two South American mammals, Bradypus variegatus (the Brown-throated Sloth) and Microryzomys minutus (the Forest Small Rice Rat).

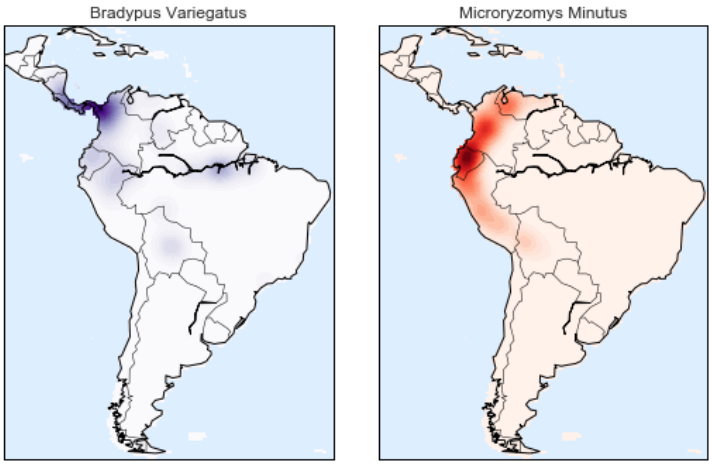


The above graph doesn't give a very good idea of the density of the species, because points in the species range may overlap one another.

Kernel density estimation can show this distribution in a more interpretable way: as a smooth indication of density on the map. Because the coordinate system here lies on a spherical surface rather than a flat

plane, Euclidean distance cannot be used, rather we will use the haversine distance metric(distance measured across the globe-great spherical distance)(https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwis

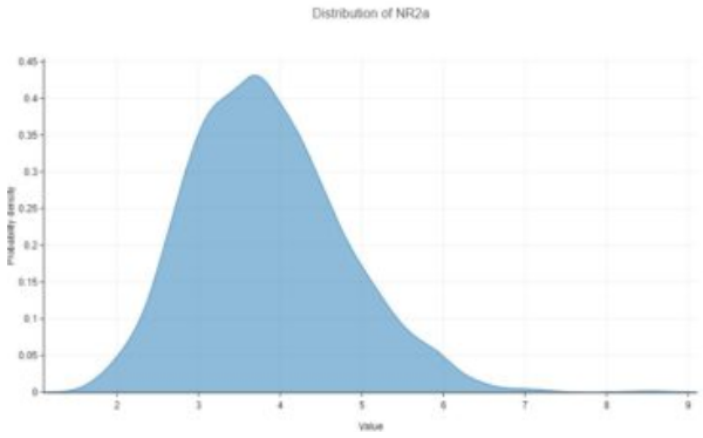
13 Spherical KDE



This graph gives a better understanding of the distribution and density of the data such as concentration of mammals depending on the area, outliers etc.

14 KDE for Visualization (Violin Plot)

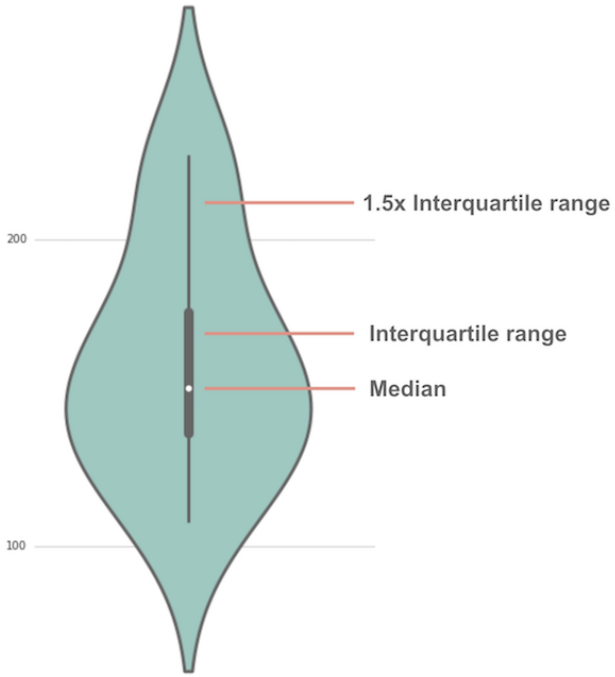
Standard Density Plot:



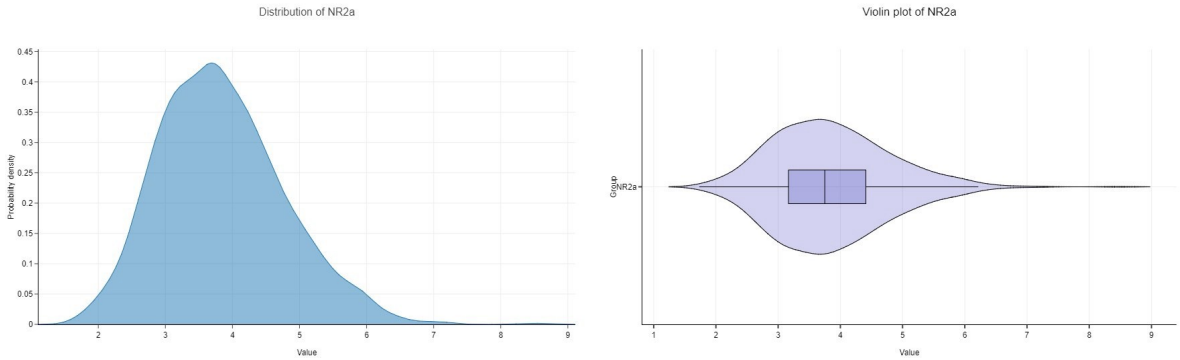
It does not represent frequency and spread across data points.

Violin plot:

1. Represents the outliers and frequency well
2. Uses KDE to produce the graph



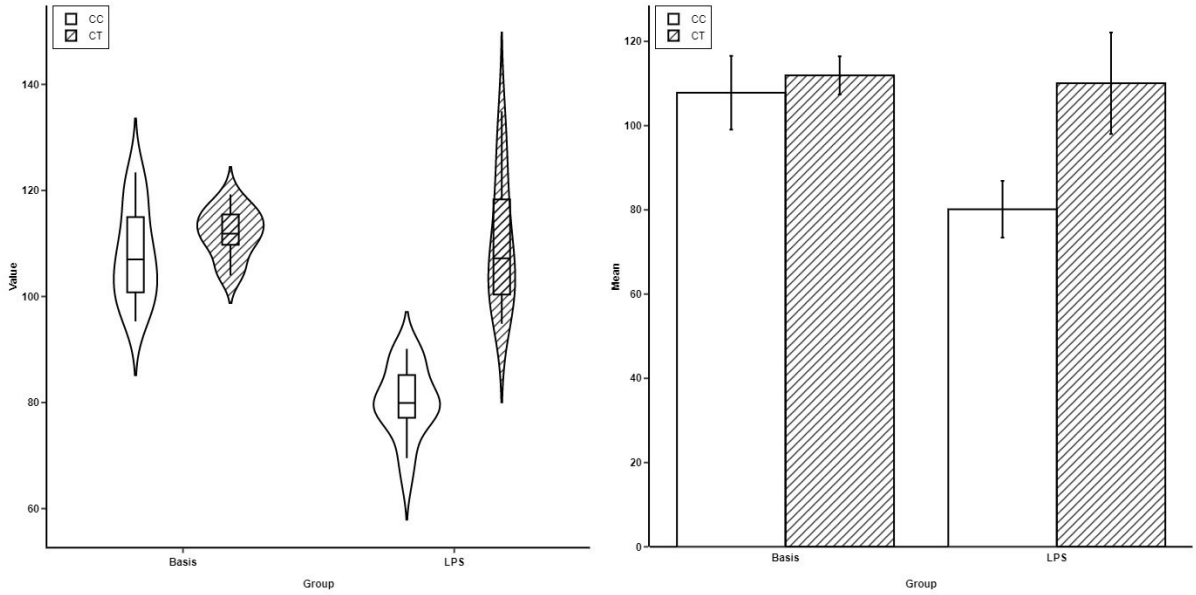
The "violin" shape of a violin plot comes from the data's density plot. You just turn that density plot sideway and put it on both sides of the box plot, mirroring each other



Reading the violin shape is same as reading a density plot(also mentioned in above diagram): the thicker part means the values in that section of the violin has higher frequency, and the thinner part implies lower frequency.

The reason why we can't just use a density plot instead of violin plot: When there are too many groups (more than 3), their overlapping density plots become difficult to read.

15 Violin plots are non-parametric



Unlike bar graphs with means and error bars, violin plots contain all data points.This make them an excellent tool to visualize samples of small sizes. Violin plots are perfectly appropriate even if your data do not conform to normal distribution. They work well to visualize both quantitative and qualitative data.

16 Summary:

When compared to the commonly used histogram, the kernel density estimator shows several advantages.

1. It is a smooth curve and thus it better exhibits the details of the pdf, suggesting in some cases non-unimodality.
2. It uses all sample points' locations, so, therefore, it better reveal the information contained in the sample.
3. It more convincingly suggests multimodality.
4. The bias of the kernel estimator is of one order better than that of a histogram estimator
5. Compared with 1D application, 2D kernel applications are even more better as the 2D histogram requires additionally the specification of the orientation of the bins which enhances the subjectivity of histogram.

17 KDE

1. Non-parametric density estimation - generative .
2. Advantage: Data-driven, Data-adaptive
3. Disadvantage: Need to keep around all data samples to estimate the density, sensitive to bandwidth 'h' and choice of kernel

18 Application of KDE(Mean Shift clustering)

1. Describes a general non-parametric method that locates the maxima of density functions in application to Clustering.
2. Locates the density function maxima (mean shift algorithm) and then assign points to the nearest maxima similar to the centroid in K-means
3. the number of clusters is not required for its implementation and, as it's density based, it can detect clusters of any shape.
4. Instead, the algorithm relies on a bandwidth parameter, which simply determines the size of neighbourhood over which the density will be computed.
5. A small bandwidth could generate excessive clusters, while a high value could erroneously combine multiple clusters.
6. high dimensionality, mean shift may also converge to local optima rather than global optima
7. Computationally expensive

19 K-means:

• Algorithm:

1. Start with pre-specified number of cluster centres.
2. Each point is assigned to the nearest centre.
3. For each segment, the centres are moved to the centroid of clustered points i. Points are then reassigned to their nearest centre.
4. Process is repeated until moving the centres is stopped.

• Advantage:

Performs well on linearly separable data.

• Disadvantage:

Underperforms on globular(spherical) clusters and with clusters of different size and density.

Note : Algorithms in the k-means family such as K- median, K- medoids are sensitive to the starting position of the cluster centres, as each method converges to local optima, the frequency of which increase in higher dimensions.

20 Expectation Maximisation algorithm – GMM

This technique is an application of the general expectation maximisation (EM) algorithm to the task of clustering. It is conceptually related and visually similar to k-means Where k-means seeks to minimise

the distance between the observations and their assigned centroids, EM estimates some latent variables (typically the mean and covariance matrix of a multinomial normal distribution (called Gaussian Mixture Models (GMM))), so as to maximise the log-likelihood of the observed data. Similar to k-means, the algorithm converges to the final clustering by iteratively improving its performance (i.e. reducing the log-likelihood). However, again like k-means, there is no guarantee that the algorithm has settled on the global minimum rather than local minimum (a concern that increases in higher dimensions).

In contrast to Kmeans, observations are not explicitly assigned to clusters, but rather given probabilities of belonging to each distribution. If the underlying distribution is correctly identified, then the algorithm performs well. In practice, especially for large data sets, the underlying distribution may not be retrievable, so EM clustering may not be well suited to such tasks.

Features:

1. Performs well on normally disturbed data.
2. Under performs on circular or spherical data and large data sets .

21 Hierarchical Clustering

Unlike K-means, EM, this clustering doesn't require the user to specify the number of clusters beforehand instead it returns an output(Dendogram), from which the user can decide the appropriate number of clusters or just return specific number of clusters (similar to K-means)

Comes in 2 flavors:

1. Divisive: Starts with the entire dataset comprising one cluster that is iteratively split- one point at a time- until each point forms its own cluster.
2. Agglomerative: The agglomerative method in reverse- individual points are iteratively combined until all points belong to the same cluster.

HC has an important concept called the "linkage criterion". This defines the distance between clusters as a function of the points in each cluster and determines which clusters are merged/split at each step.

Features:

1. It captures the non-globular structures within the data set by imposing simple connectivity constraints like points can only cluster with their n(=5) nearest neighbours.
2. Doesn't perform well on noisy circles and large data sets.

22 References:

1. https://dashee87.github.io/data20science/general/Clustering-with-Scikit-with-GIFs/
2. https://medium.com/@bioturing/5-reasons-you-should-use-a-violin-graph-31a9cdf2d0c6
3. https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html
4. https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0