Title: Kernel density estimation

URL: https://en.wikipedia.org/wiki/Kernel_density_estimation

PageID: 2090057

Categories: Category:Estimation of densities, Category:Machine learning, Category:Nonparametric statistics

Source: Wikipedia (CC BY-SA 4.0).

In statistics , kernel density estimation (KDE) is the application of kernel smoothing for probability density estimation , i.e., a non-parametric method to estimate the probability density function of a random variable based on kernels as weights . KDE answers a fundamental data smoothing problem where inferences about the population are made based on a finite data sample . In some fields such as signal processing and econometrics it is also termed the Parzen–Rosenblatt window method, after Emanuel Parzen and Murray Rosenblatt , who are usually credited with independently creating it in its current form. [1] [2] One of the famous applications of kernel density estimation is in estimating the class-conditional marginal densities of data when using a naive Bayes classifier , which can improve its prediction accuracy. [3]

Definition

Let (x 1 , x 2 , ..., x n) be independent and identically distributed samples drawn from some univariate distribution with an unknown density f at any given point x . We are interested in estimating the shape of this function f . Its kernel density estimator is f ^ h (x) = 1 n ∑ i = 1 n K h (x – x i) = 1 n h ∑ i = 1 n K (x – x i h) , {\displaystyle {\hat {f}}_{h}(x)={\frac {1}{n}}\sum _{i=1}^{n}K_{h}(x-x_{i})={\frac {1}{nh}}\sum _{i=1}^{n}K{\left({\frac {x-x_{i}}{h}}\right)},} where K is the kernel — a non-negative function — and h > 0 is a smoothing parameter called the bandwidth or simply width. [3] A kernel with subscript h is called the scaled kernel and defined as K h (x) = ■ 1 / h ■ K (■ x / h ■) . Intuitively one wants to choose h as small as the data will allow; however, there is always a trade-off between the bias of the estimator and its variance. The choice of bandwidth is discussed in more detail below.

A range of kernel functions are commonly used: uniform, triangular, biweight, triweight, Epanechnikov (parabolic), normal, and others. The Epanechnikov kernel is optimal in a mean square error sense, [4] though the loss of efficiency is small for the kernels listed previously. [5] Due to its convenient mathematical properties, the normal kernel is often used, which means K (x) = φ (x) , where φ is the standard normal density function. The kernel density estimator then becomes f ^ h (x) = 1 n h σ 1 2 π ∑ i = 1 n exp ■ (– (x – x i) 2 2 h 2 σ 2) , {\displaystyle {\hat {f}}_{h}(x)={\frac {1}{nh\sigma }}{\frac {1}{\sqrt {2\pi }}}\sum _{i=1}^{n}\exp \left({\frac {-(x-x_{i})^{2}}{2h^{2}\sigma ^{2}}}\right),} where σ {\displaystyle \sigma } is the standard deviation of the sample x → {\displaystyle {\vec {x}}} .

The construction of a kernel density estimate finds interpretations in fields outside of density estimation. [6] For example, in thermodynamics , this is equivalent to the amount of heat generated when heat kernels (the fundamental solution to the heat equation) are placed at each data point locations x i . Similar methods are used to construct discrete Laplace operators on point clouds for manifold learning (e.g. diffusion map).

Example

Kernel density estimates are closely related to histograms , but can be endowed with properties such as smoothness or continuity by using a suitable kernel. The diagram below based on these 6 data points illustrates this relationship:

For the histogram, first, the horizontal axis is divided into sub-intervals or bins which cover the range of the data: In this case, six bins each of width 2. Whenever a data point falls inside this interval, a box of height 1/12 is placed there. If more than one data point falls inside the same bin, the boxes are stacked on top of each other.

For the kernel density estimate, normal kernels with a standard deviation of 1.5 (indicated by the red dashed lines) are placed on each of the data points x i . The kernels are summed to make the kernel density estimate (solid blue curve). The smoothness of the kernel density estimate (compared to the discreteness of the histogram) illustrates how kernel density estimates converge faster to the true underlying density for continuous random variables. [7]

Bandwidth selection

The bandwidth of the kernel is a free parameter which exhibits a strong influence on the resulting estimate. To illustrate its effect, we take a simulated random sample from the standard normal distribution (plotted at the blue spikes in the rug plot on the horizontal axis). The grey curve is the true density (a normal density with mean 0 and variance 1). In comparison, the red curve is undersmoothed since it contains too many spurious data artifacts arising from using a bandwidth h = 0.05, which is too small. The green curve is oversmoothed since using the bandwidth h = 2 obscures much of the underlying structure. The black curve with a bandwidth of h = 0.337 is considered to be optimally smoothed since its density estimate is close to the true density. An extreme situation is encountered in the limit h $\to$ 0 {\displaystyle h\to 0} (no smoothing), where the estimate is a sum of n delta functions centered at the coordinates of analyzed samples. In the other extreme limit h $\to$ ∞ {\displaystyle h\to \infty } the estimate retains the shape of the used kernel, centered on the mean of the samples (completely smooth).

The most common optimality criterion used to select this parameter is the expected L 2 risk function , also termed the mean integrated squared error :

MISE ■ (h) = E [∫ (f ^ h (x) − f (x)) 2 d x] {\displaystyle \operatorname {MISE} (h)=\operatorname {E} \!\left[\int \!{\left({\hat {f}}\!_{h}(x)-f(x)\right)}^{2}dx\right]}

Under weak assumptions on f and K , (f is the, generally unknown, real density function), [1] [2]

MISE ■ (h) = AMISE ■ (h) + o ((n h) − 1 + h 4) {\displaystyle \operatorname {MISE} (h)=\operatorname {AMISE} (h)+{\mathcal {o}}{\left((nh)^{-1}+h^{4}\right)}}

where o is the little o notation , and n the sample size (as above). The AMISE is the asymptotic MISE, i. e. the two leading terms,

AMISE ■ (h) = R (K) n h + 1 4 m 2 (K) 2 h 4 R (f ″) {\displaystyle \operatorname {AMISE} (h)={\frac {R(K)}{nh}}+{\frac {1}{4}}m_{2}(K)^{2}h^{4}R(f'')}

where R (g) = ∫ g (x) 2 d x {\textstyle R(g)=\int g(x)^{2}\,dx} for a function g , m 2 (K) = ∫ x 2 K (x) d x {\textstyle m_{2}(K)=\int x^{2}K(x)\,dx} and f ″ {\displaystyle f''} is the second derivative of f {\displaystyle f} and K {\displaystyle K} is the kernel. The minimum of this AMISE is the solution to this differential equation

∂ ∂ h AMISE ■ (h) = − R (K) n h 2 + m 2 (K) 2 h 3 R (f ″) = 0 {\displaystyle {\frac {\partial }{\partial h}}\operatorname {AMISE} (h)=-{\frac {R(K)}{nh^{2}}}+m_{2}(K)^{2}h^{3}R(f'')=0}

or

h AMISE = R (K) 1 / 5 m 2 (K) 2 / 5 R (f ″) 1 / 5 n − 1 / 5 = C n − 1 / 5 {\displaystyle h_{\operatorname {AMISE} }={\frac {R(K)^{1/5}}{m_{2}(K)^{2/5}R(f'')^{1/5}}}n^{-1/5}=Cn^{-1/5}}

Neither the AMISE nor the h AMISE formulas can be used directly since they involve the unknown density function f {\displaystyle f} or its second derivative f ″ {\displaystyle f''} . To overcome that difficulty, a variety of automatic, data-based methods have been developed to select the bandwidth. Several review studies have been undertaken to compare their efficacies, [8] [9] [10] [11] [12] [13] [14] with the general consensus that the plug-in selectors [6] [15] [16] and cross validation selectors [17] [18] [19] are the most useful over a wide range of data sets.

Substituting any bandwidth h which has the same asymptotic order n −1/5 as h AMISE into the AMISE gives that AMISE(h) = O (n −4/5) , where O is the big O notation . It can be shown that, under weak assumptions, there cannot exist a non-parametric estimator that converges at a faster rate than the kernel estimator. [20] Note that the n −4/5 rate is slower than the typical n −1 convergence rate of parametric methods.

If the bandwidth is not held fixed, but is varied depending upon the location of either the estimate (balloon estimator) or the samples (pointwise estimator), this produces a particularly powerful method termed adaptive or variable bandwidth kernel density estimation .

Bandwidth selection for kernel density estimation of heavy-tailed distributions is relatively difficult. [21]

A rule-of-thumb bandwidth estimator

If Gaussian basis functions are used to approximate univariate data, and the underlying density being estimated is Gaussian, the optimal choice for h (that is, the bandwidth that minimises the mean integrated squared error) is: [22]

h = (4 σ ^ 5 3 n) 1 / 5 ≈ 1.06 σ ^ n – 1 / 5 , {\displaystyle h={\left({\frac {4{\hat {\sigma }}^{5}}{3n}}\right)}^{1/5}\approx 1.06\,{\hat {\sigma }}\,n^{-1/5},}

An h {\displaystyle h} value is considered more robust when it improves the fit for long-tailed and skewed distributions or for bimodal mixture distributions. This is often done empirically by replacing the standard deviation σ ^ {\displaystyle {\hat {\sigma }}} by the parameter A {\displaystyle A} below:

A = min (σ ^ , I Q R 1.34) {\displaystyle A=\min \left({\hat {\sigma }},{\frac {\mathrm {IQR} }{1.34}}\right)} where IQR is the interquartile range.

Another modification that will improve the model is to reduce the factor from 1.06 to 0.9. Then the final formula would be:

h = 0.9 min (σ ^ , I Q R 1.34) n – 1 / 5 {\displaystyle h=0.9\,\min \left({\hat {\sigma }},{\frac {\mathrm {IQR} }{1.34}}\right)\,n^{-1/5}} where n {\displaystyle n} is the sample size.

This approximation is termed the normal distribution approximation , Gaussian approximation, or Silverman 's rule of thumb . [22] While this rule of thumb is easy to compute, it should be used with caution as it can yield widely inaccurate estimates when the density is not close to being normal.

For example, when estimating the bimodal Gaussian mixture model 1 2 2 π e – 1 2 (x – 10) 2 + 1 2 2 π e – 1 2 (x + 10) 2 {\displaystyle {\frac {1}{2{\sqrt {2\pi }}}}e^{-{\frac {1}{2}}(x-10)^{2}}+{\frac {1}{2{\sqrt {2\pi }}}}e^{-{\frac {1}{2}}(x+10)^{2}}} from a sample of 200 points, the figure on the right shows the true density and two kernel density estimates — one using the rule-of-thumb bandwidth, and the other using a solve-the-equation bandwidth. [6] [16] The estimate based on the rule-of-thumb bandwidth is significantly oversmoothed.

Relation to the characteristic function density estimator

Given the sample (x 1 , x 2 , ..., x n) , it is natural to estimate the characteristic function ϕ (t) = E[e itX] as ϕ ^ (t) = 1 n ∑ j = 1 n e i t x j {\displaystyle {\hat {\varphi }}(t)={\frac {1}{n}}\sum _{j=1}^{n}e^{itx_{j}}} Knowing the characteristic function, it is possible to find the corresponding probability density function through the Fourier transform formula. One difficulty with applying this inversion formula is that it leads to a diverging integral, since the estimate ϕ ^ (t) {\displaystyle {\hat {\varphi }}(t)} is unreliable for large t 's. To circumvent this problem, the estimator ϕ ^ (t) {\displaystyle {\hat {\varphi }}(t)} is multiplied by a damping function ψ h (t) = ψ (ht) , which is equal to 1 at the origin and then falls to 0 at infinity. The "bandwidth parameter" h controls how fast we try to dampen the function ϕ ^ (t) {\displaystyle {\hat {\varphi }}(t)} . In particular when h is small, then ψ h (t) will be approximately one for a large range of t 's, which means that ϕ ^ (t) {\displaystyle {\hat {\varphi }}(t)} remains practically unaltered in the most important region of t 's.

The most common choice for function ψ is either the uniform function ψ (t) = 1 {–1 ≤ t ≤ 1} , which effectively means truncating the interval of integration in the inversion formula to [–1/ h , 1/ h] , or the Gaussian function ψ (t) = e –πt 2 . Once the function ψ has been chosen, the inversion formula may be applied, and the density estimator will be f ^ (x) = 1 2 π ∫ – ∞ + ∞ ϕ ^ (t) ψ h (t) e – i t x d t = 1 2 π ∫ – ∞ + ∞ 1 n ∑ j = 1 n e i t (x j – x) ψ (h t) d t = 1 n h ∑ j = 1 n 1 2 π ∫ – ∞ + ∞ e – i (h t) x – x j h ψ (h t) d (h t) = 1 n h ∑ j = 1 n K (x – x j h) , {\displaystyle {\begin{aligned}{\hat {f}}(x)&={\frac {1}{2\pi }}\int _{-\infty }^{+\infty }{\hat {\varphi }}(t)\psi _{h}(t)e^{-itx}\,dt\\[1ex]&={\frac {1}{2\pi }}\int _{-\infty }^{+\infty }{\frac {1}{n}}\sum _{j=1}^{n}e^{it(x_{j}-x)}\psi (ht)\,dt\\[1ex]&={\frac {1}{nh}}\sum _{j=1}^{n}{\frac {1}{2\pi }}\int _{-\infty }^{+\infty }e^{-i(ht){\frac {x-x_{j}}{h}}}\psi

(ht)\,d(ht)\\[1ex]&={\frac {1}{nh}}\sum _{j=1}^{n}K{\left({\frac {x-x_{j}}{h}}\right)},\end{aligned}}}

where K is the Fourier transform of the damping function ψ . Thus the kernel density estimator coincides with the characteristic function density estimator.

Geometric and topological features

We can extend the definition of the (global) mode to a local sense and define the local modes:

M = { x : g (x) = 0 ■ λ 1 (x) < 0 } {\displaystyle M=\{x:g(x)=0\mid \lambda _{1}(x)<0\}}

Namely, M {\displaystyle M} is the collection of points for which the density function is locally maximized. A natural estimator of M {\displaystyle M} is a plug-in from KDE, [23] [24] where g (x) {\displaystyle g(x)} and λ 1 (x) {\displaystyle \lambda _{1}(x)} are KDE version of g (x) {\displaystyle g(x)} and λ 1 (x) {\displaystyle \lambda _{1}(x)} . Under mild assumptions, M c {\displaystyle M_{c}} is a consistent estimator of M {\displaystyle M} . Note that one can use the mean shift algorithm [25] [26] [27] to compute the estimator M c {\displaystyle M_{c}} numerically.

Statistical implementation

A non-exhaustive list of software implementations of kernel density estimators includes:

In Analytica release 4.4, the Smoothing option for PDF results uses KDE, and from expressions it is available via the built-in Pdf function.

In C / C++ , FIGTree is a library that can be used to compute kernel density estimates using normal kernels. MATLAB interface available.

In C++ , libagf is a library for variable kernel density estimation .

In C++ , mlpack is a library that can compute KDE using many different kernels. It allows to set an error tolerance for faster computation. Python and R interfaces are available.

in C# and F# , Math.NET Numerics is an open source library for numerical computation which includes kernel density estimation

In CrimeStat , kernel density estimation is implemented using five different kernel functions – normal, uniform, quartic, negative exponential, and triangular. Both single- and dual-kernel density estimate routines are available. Kernel density estimation is also used in interpolating a Head Bang routine, in estimating a two-dimensional Journey-to-crime density function, and in estimating a three-dimensional Bayesian Journey-to-crime estimate.

In ELKI , kernel density functions can be found in the package de.lmu.ifi.dbs.elki.math.statistics.kernelfunctions

In ESRI products, kernel density mapping is managed out of the Spatial Analyst toolbox and uses the Quartic(biweight) kernel.

In Excel , the Royal Society of Chemistry has created an add-in to run kernel density estimation based on their Analytical Methods Committee Technical Brief 4 .

In gnuplot , kernel density estimation is implemented by the smooth kdensity option, the datafile can contain a weight and bandwidth for each point, or the bandwidth can be set automatically [28] according to "Silverman's rule of thumb" (see above).

In Haskell , kernel density is implemented in the statistics package.

In IGOR Pro , kernel density estimation is implemented by the StatsKDE operation (added in Igor Pro 7.00). Bandwidth can be user specified or estimated by means of Silverman, Scott or Bowmann and Azzalini . Kernel types are: Epanechnikov, Bi-weight, Tri-weight, Triangular, Gaussian and Rectangular.

In Java , the Weka machine learning package provides weka.estimators.KernelEstimator , among others.

In JavaScript , the visualization package D3.js offers a KDE package in its science.stats package.

In JMP , the Graph Builder platform utilizes kernel density estimation to provide contour plots and high density regions (HDRs) for bivariate densities, and violin plots and HDRs for univariate densities. Sliders allow the user to vary the bandwidth. Bivariate and univariate kernel density estimates are also provided by the Fit Y by X and Distribution platforms, respectively.

In Julia , kernel density estimation is implemented in the KernelDensity.jl package.

In KNIME , 1D and 2D Kernel Density distributions can be generated and plotted using nodes from the Vernalis community contribution, e.g. 1D Kernel Density Plot , among others. The underlying implementation is written in Java .

In MATLAB , kernel density estimation is implemented through the ksdensity function (Statistics Toolbox). As of the 2018a release of MATLAB, both the bandwidth and kernel smoother can be specified, including other options such as specifying the range of the kernel density. [29] Alternatively, a free MATLAB software package which implements an automatic bandwidth selection method [6] is available from the MATLAB Central File Exchange for 1-dimensional data 2-dimensional data n-dimensional data A free MATLAB toolbox with implementation of kernel regression, kernel density estimation, kernel estimation of hazard function and many others is available on these pages (this toolbox is a part of the book [30]).

1-dimensional data

2-dimensional data

n-dimensional data A free MATLAB toolbox with implementation of kernel regression, kernel density estimation, kernel estimation of hazard function and many others is available on these pages (this toolbox is a part of the book [30]).

In Mathematica , numeric kernel density estimation is implemented by the function SmoothKernelDistribution [31] and symbolic estimation is implemented using the function KernelMixtureDistribution [32] both of which provide data-driven bandwidths.

In Minitab , the Royal Society of Chemistry has created a macro to run kernel density estimation based on their Analytical Methods Committee Technical Brief 4. [33]

In the NAG Library , kernel density estimation is implemented via the g10ba routine (available in both the Fortran [34] and the C [35] versions of the Library).

In Nuklei , C++ kernel density methods focus on data from the Special Euclidean group $SE(3)$.

In Octave , kernel density estimation is implemented by the kernel_density option (econometrics package).

In Origin , 2D kernel density plot can be made from its user interface, and two functions, Ksdensity for 1D and Ks2density for 2D can be used from its LabTalk , Python , or C code.

In Perl , an implementation can be found in the Statistics-KernelEstimation module

In PHP , an implementation can be found in the MathPHP library

In Python , many implementations exist: pyqt_fit.kde Module in the PyQt-Fit package , SciPy (scipy.stats.gaussian_kde), Statsmodels (KDEUnivariate and KDEMultivariate), and scikit-learn (KernelDensity) (see comparison [36]). KDEpy supports weighted data and its FFT implementation is orders of magnitude faster than the other implementations. The commonly used pandas library [1] offers support for kde plotting through the plot method (df.plot(kind='kde') [2]). The getdist package for weighted and correlated MCMC samples supports optimized bandwidth, boundary correction and higher-order methods for 1D and 2D distributions. One newly used package for kernel density estimation is seaborn (import seaborn as sns , sns.kdeplot()). [37] A GPU implementation of KDE also exists. [38]

In R , it is implemented through density in the base distribution, and bw.nrd0 function is used in stats package, this function uses the optimized formula in Silverman's book. bkde in the KernSmooth library , ParetoDensityEstimation in the DataVisualizations library (for pareto distribution density estimation), kde in the ks library , dkden and dbckden in the evmix library (latter

for boundary corrected kernel density estimation for bounded support), npudens in the np library (numeric and categorical data), sm.density in the sm library . For an implementation of the kde.R function, which does not require installing any packages or libraries, see kde.R . The btb library , dedicated to urban analysis, implements kernel density estimation through kernel_smoothing .

In SAS , proc kde can be used to estimate univariate and bivariate kernel densities.

In Apache Spark , the KernelDensity() class [39]

In Stata , it is implemented through kdensity ; [40] for example histogram x, kdensity . Alternatively a free Stata module KDENS is available [41] allowing a user to estimate 1D or 2D density functions.

In Swift , it is implemented through SwiftStats.KernelDensityEstimation in the open-source statistics library SwiftStats .

See also

Kernel (statistics)

Kernel smoothing

Kernel regression

Density estimation (with presentation of other examples)

Mean-shift

Scale space : The triplets {(x , h , KDE with bandwidth h evaluated at x : all x , h > 0} form a scale space representation of the data.

Multivariate kernel density estimation

Variable kernel density estimation

Head/tail breaks

Further reading

Härdle, Wolfgang; Müller, Marlene; Sperlich, Stefan; Werwatz, Axel (2004). Nonparametric and Semiparametric Models . Springer Series in Statistics. Berlin Heidelberg: Springer-Verlag. pp. 39–83. ISBN 978-3-540-20722-1 .

References

External links

Introduction to kernel density estimation A short tutorial which motivates kernel density estimators as an improvement over histograms.

Kernel Bandwidth Optimization A free online tool that generates an optimized kernel density estimate.

Free Online Software (Calculator) computes the Kernel Density Estimation for a data series according to the following Kernels: Gaussian, Epanechnikov, Rectangular, Triangular, Biweight, Cosine, and Optcosine.

Kernel Density Estimation Applet An online interactive example of kernel density estimation. Requires .NET 3.0 or later.