kalepy: a python package for kernel density estimation, sampling and plotting

2020 October 11

Summary

‘Kernel Density Estimation’ or ‘KDE’ (Rosenblatt 1956; Parzen 1962) is a type of non-parametric density estimation (David W. Scott 2015) that improves upon the traditional ‘histogram’ approach by, for example, i) utilizing the exact location of each data point (instead of ‘binning’), ii) being able to produce smooth distributions with continuous and meaningful derivatives, and iii) removing the arbitrary offset of an initial bin edge. The `kalepy` package presents a python KDE implementation designed for broad applicability by including numerous features absent in other packages presented in a class-based structure designed for extensibility. `kalepy` provides optional weightings, reflecting boundary conditions, an arbitrary number of dimensions, numerous (and extensible) kernel (i.e. window) functions, built-in plotting, and built-in resampling.

Statement of need

Numerous python KDE implementations exist, for example in `scipy` (`scipy.stats.gaussian_kde`) (Virtanen et al. 2020), `seaborn` (`seaborn.kdeplot`) (Waskom and team 2020), `GetDist` (Lewis 2019) and `KDEpy` (Odland 2018). The `scipy` and `seaborn` tools are simple and accessible. The `KDEpy` package provides excellent performance on large numbers of data points and dimensions, but does not include resampling, boundary conditions, or plotting tools. The `GetDist` package offers extensive methods for plotting samples and utilizes numerous boundary treatments (Lewis 2019), but lacks a standalone KDE interface or resampling functionality. `kalepy` provides convenient access to both plotting and numerical results in the same package, including multiple kernel functions, built-in resampling, boundary conditions, and numerous plotting tools for 1D, 2D, and N-dimensional ‘corner’ plots. `kalepy` is entirely class-based, and while focusing on ease of use, provides a highly extensible framework for modification and expansion in a range of possible applications.

While `kalepy` has no features specific to any particular field, it was designed for resampling from weighted astronomical datasets. Consider a population of binaries derived from cosmological simulations. If the initial population is costly to produce (e.g. requiring tens of millions of CPU hours), and as long as it accurately samples the parameter space of interest, it may be sufficiently accurate to produce larger populations by ‘resampling with variation,’ e.g. using a KDE approach. Depending on the details of the population, many of the parameters may be highly correlated and often abut a boundary: for example, the mass-ratio defined as the lower-mass component divided by the more massive component, is often highly correlated with the total mass of the binary, and is bounded to the unit interval i.e. $q \equiv M_2/M_1 \leq 1$. Faithfully resampling from the population requires handling this discontinuity, while also preserving accurate covariances which may be distorted when transforming the variable, performing the KDE, and transforming back.

Methods

Consider a d dimensional parameter space, with N data points given by $x_i = (x_{i1}, x_{i2}, ..., x_{id})$, with $i = \{1, ..., N\}$. Each data points may have an associated ‘weight’ that is appropriately normalized, $\sum_i^N w_i = 1$.

The kernel density estimate at a general position $x = (x_1, x_2, ..., x_N)$ can be written as,

$$\hat{f}_H(x) = \sum_{i=1}^{N} w_i K_H(x - x_i),$$

where the kernel can typically be expressed as,

$$K_H(x) = \|H\|^{-1/2} K\left(H^{-1/2} x\right).$$

Here H is the 'bandwidth' (or covariance) matrix. Choosing the kernel and bandwidth matrix produces most of the nuance and art of KDE. The most common choice of kernel is likely the Gaussian, i.e.,

$$\hat{f}_H(x) = \sum_{i=1}^{N} \frac{w_i}{(2\pi)^{-d/2} \|H\|^{1/2}} \exp\{(x_j - x_{ij}) H^j{}_k (x^k - x_i{}^k)\}.$$

In the current implementation, the Gaussian, tri-weight, and box-car kernels are implemented, in addition to the Epanechnikov kernel (Epanechnikov 1969) which in some cases has been shown to be statistically optimal but has discontinuous derivatives that can produce both numerical and aesthetic problems. Often the bandwidth is chosen to be diagonal, and different rules-of-thumb are typically used to approximate a bandwidth that minimizes typical measures of error and/or bias. For example, the so-called 'Silverman factor' (Silverman 1978) bandwidth,

$$H_{ij} = \delta_{ij} \sigma_i \left[\frac{4}{(d+2)n}\right]^{1/(d+4)} \quad \text{(summation not implied)},$$

where δ_{ij} is the Kronecker delta, and σ_i is the standard deviation (or its estimate) for the ith parameter. In the current implementation, both the Silverman and Scott factor (David W. Scott 1979) bandwidth estimators are included.

Reflecting boundary conditions can be used to improve reconstruction accuracy. For example, with data drawn from a log-normal distribution, a standard KDE will produce 'leakage' outside of the domain. To enforce the restriction that $f(x < 0) = 0$ (which must be known {*a priori*}), the kernel is redefined such that $K_H(x < 0) = 0$, and re-normalized to preserve unitarity[1]. This example is shown in Figure 1, with histograms in the upper panel and KDEs on the bottom.

Resampling from the derived PDF can be done much more efficiently in the KDE framework than by the standard method of CDF inversion. In particular, we can see that sampling from the PDF is identical to re-sampling with replacement from the weighted data points, while shifting each point based on the PDF of the Kernel at that location.

`kalepy` has recently been used in astronomy and astrophysics, particularly in Siwek, Kelley, and Hernquist (2020), Kelley (2020), Andrews (2020).

Acknowledgements

We acknowledge very helpful consultations on statistical nuances from Christopher Berry and Diego Muñoz. The support and 'beta-testing' performed by Magda Siwek and Jeff Andrews is also much appreciated. `kalepy` utilizes tools and functionality from `numpy` (Harris et al. 2020), `matplotlib` (Hunter 2007), `scipy` (Virtanen et al. 2020), `jupyter` notebooks (ipython-2007; Kluyver et al. 2016), and `corner` (Foreman-Mackey 2016).

[1]Note that some implementations instead truncate and renormalize the resulting $\hat{f}_H$ which which incorrectly redistributes probability from near the boundaries to the whole domain.

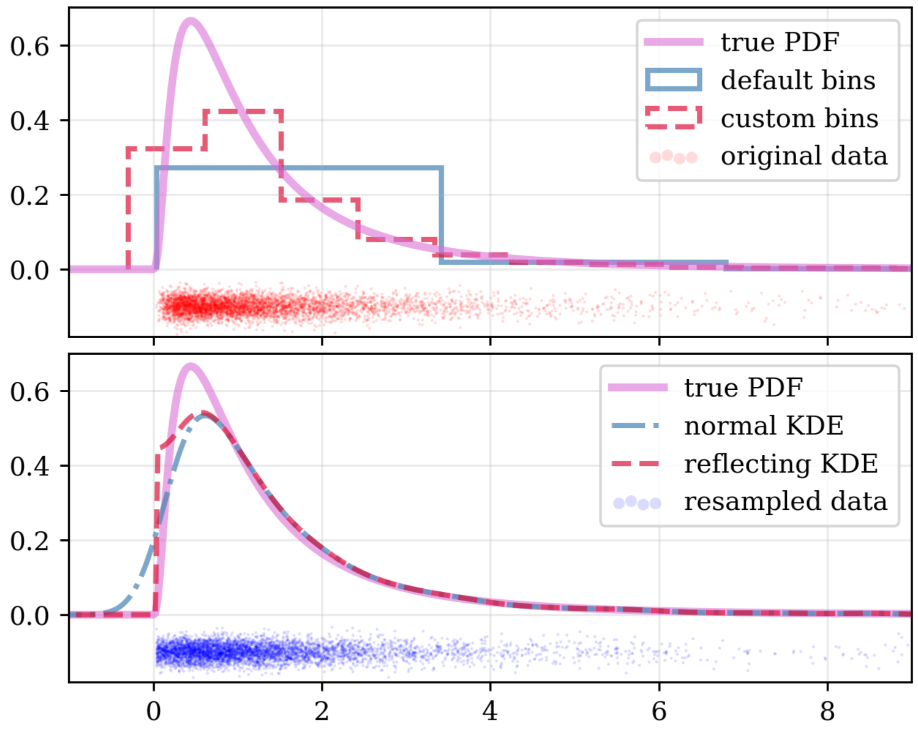

Figure 1: Data drawn from a log-normal distribution is used to estimate the underlying PDF using histgrams (upper) and KDEs (lower). The true distribution is shown in magenta. In the upper panel, the default bins chosen by `matplotlib` are especially uninsightful (blue), while custom bins misrepresent the distributions position when the initial edge is poorly chosen (red). The data is also included as a 'carpet' plot. In the lower panel, a Gaussian KDE with no reflection (blue) is compared to one with a reflection at $x = 0$, which better reproduces the true PDF. Data resampled from the reflecting-KDE PDF is shown as the blue 'carpet' points which closely resemble the input data.

References

Andrews, Jeff J. 2020. "Mass Ratios of Merging Double Neutron Stars as Implied by the Milky Way Population" 900 (2): L41. https://doi.org/10.3847/2041-8213/abb1bf.

Epanechnikov, V. A. 1969. "Non-Parametric Estimation of a Multivariate Probability Density." *Theory of Probability & Its Applications* 14 (1): 153–58. https://doi.org/10.1137/1114019.

Foreman-Mackey, Daniel. 2016. "Corner.py: Scatterplot Matrices in Python." *Journal of Open Source Software* 1 (2): 24. https://doi.org/10.21105/joss.00024.

Harris, Charles R., K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hunter, J. D. 2007. "Matplotlib: A 2d Graphics Environment." *Computing In Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/mcse.2007.55.

Kelley, Luke Zoltan. 2020. "Considerations for the Observability of Kinematically Offset Binary AGN." *arXiv e-Prints*, May, arXiv:2005.10255. http://arxiv.org/abs/2005.10255.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, et al. 2016. "Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows." Edited by F. Loizides and B. Schmidt. IOS Press.

Lewis, Antony. 2019. "GetDist: a Python package for analysing Monte Carlo samples." https://getdist.readthedocs.io.

Odland, Tommy. 2018. *Tommyod/KDEpy: Kernel Density Estimation in Python* (version v0.9.10). Zenodo. https://doi.org/10.5281/zenodo.2392268.

Parzen, Emanuel. 1962. "On Estimation of a Probability Density Function and Mode." *Ann. Math. Statist.* 33 (3): 1065–76. https://doi.org/10.1214/aoms/1177704472.

Rosenblatt, Murray. 1956. "Remarks on Some Nonparametric Estimates of a Density Function." *Ann. Math. Statist.* 27 (3): 832–37. https://doi.org/10.1214/aoms/1177728190.

Scott, David W. 1979. "On optimal and data-based histograms." *Biometrika* 66 (3): 605–10. https://doi.org/10.1093/biomet/66.3.605.

Scott, David W. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons.

Silverman, BW. 1978. "Choosing the Window Width When Estimating a Density." *Biometrika* 65 (1): 1–11.

Siwek, Magdalena S., Luke Zoltan Kelley, and Lars Hernquist. 2020. "The effect of differential accretion on the gravitational wave background and the present-day MBH binary population" 498 (1): 537–47. https://doi.org/10.1093/mnras/staa2361.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2.

Waskom, Michael, and the seaborn development team. 2020. *Mwaskom/Seaborn* (version latest). Zenodo. https://doi.org/10.5281/zenodo.592845.