Statistical Methods in the NPStat Package

Igor Volobouev

Texas Tech University

i.volobouev@ttu.edu

PyHEP 2019 Workshop, October 18 2019

Introduction into NPStat

- NPStat is an acronym for "nonparametric statistics".
- Developed in C++, with python API created mainly by SWIG.
- Implements a substantial number of nonparametric data analysis algorithms. Some of these are not available anywhere else.
  - Arbitrary-dimensional histogramming.
  - Density estimation by OSDE, KDE, LOrPE, kNN. Estimation of comparison densities and copula-based techniques.
  - Local regression (polynomial, logistic, quantile, least trimmed squares).
  - Univariate and multivariate nonparametric density interpolation (a.k.a. template morphing).
  - Unfolding (i.e., solution of inverse statistical problems) with regularization by smoothing.
  - Various supporting code: sample characterization, parametric density modeling and fitting, copula modeling, generation of random numbers (including QMC), numerical integration, root finding, persistence, etc.
- Useful for HEP, as most of our distributions can only be derived by simulations (GEANT). We want distributions but only get samples.
- The package is hosted on Hepforge. See npstat.hepforge.org.

Statistical Concepts Pervading NPStat

- Empirical density function (EDF): $\hat{f}_{\mathrm{EMP}}(x) = \frac{1}{n}\sum_{i=1}^{n}\delta(x - x_i)$
- Bandwidth (h)
- Filter degree (often maps one-to-one into kernel order)
- Plug-in bandwidth selector
- Cross-validation (LOO CV in particular)
- Akaike information criterion: $AIC = 2k - 2\ln(\hat{L})$
- Effective degrees of freedom: for linear regression, $\hat{\mathbf{y}} = H\mathbf{y}$. We can associate degrees of freedom with some measure of rank(H). For non-linear regression, replace H with the error propagation matrix.

Density Estimation

- Orthogonal Series Density Estimation (OSDE):
  $\hat{f}_{\mathrm{OSDE}}(x) = \frac{1}{b-a} + \sum_{j=1}^{J} w_j \hat{\theta}_j \phi_j(x), \qquad \hat{\theta}_j = \frac{1}{n} \sum_{i=1}^{n} \phi_j(x_i),$
  where $\{\phi_k\}$ is some basis orthonormal on $[a, b]$.

- Kernel Density Estimation (KDE): EDF is convolved with $\frac{1}{h} K\left(\frac{x-y}{h}\right)$
  obtaining $\hat{f}_{\mathrm{KDE}}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x-x_i}{h}\right)$.

- Local Orthogonal Polynomial Expansion (LOrPE):
  $\hat{f}_{\mathrm{LOrPE}}(x) = \sum_{k=0}^{M} c_k(x_{\mathrm{fit}}, h) P_k\left(\frac{x-x_{\mathrm{fit}}}{h}\right),$
  In this expansion, polynomials are normalized *locally*:
  $\frac{1}{h} \int_a^b P_j\left(\frac{x-x_{\mathrm{fit}}}{h}\right) P_k\left(\frac{x-x_{\mathrm{fit}}}{h}\right) K\left(\frac{x-x_{\mathrm{fit}}}{h}\right) dx = \delta_{jk}$. Then
  $c_k(x_{\mathrm{fit}}, h) = \frac{1}{h} \int_a^b \hat{f}_{\mathrm{EMP}}(x) P_k((x - x_{\mathrm{fit}})/h) K((x - x_{\mathrm{fit}})/h) dx.$

Local Regression

- Local least squares: the idea is to minimize $\sum_{i=1}^{n} \left(\frac{p(x_i|\mathbf{c}) - y_i}{\sigma_i} \right)^2 K \left(\frac{x_i - x_{\text{fit}}}{h} \right)$ over the set of coefficients $\mathbf{c}$ at every x_{fit}. Then $\hat{y}(x_{\text{fit}}) = p(x_{\text{fit}}|\mathbf{c})$.

- Other local regression techniques (logistic, quantile, least trimmed squares) are implemented in C++ but not yet ported to python.

Persistence

- NPStat persistence is based on "Geners": geners.hepforge.org.
- "Geners" is somewhat similar to boost.serialization but, like shelve, it supports random access to stored objects.
- You can use "Geners" as shelve on steroids:
  - "Geners" archives can grow as large as your disk space allows.
  - Compression and random access can be used simultaneously.
  - Archives are accessible from either python or C++. Dual API NPStat classes supporting "Geners" serialization mechanisms (arrays, histograms, probability distributions, interpolation tables, filters, etc) and various python objects that have direct C++ analogs (strings, floats, regular precision ints, numpy arrays of certain types) can be stored and retrieved using either python or C++ API.
  - Anything that can be pickled can also be stored in "Geners" archives for python-only use.
  - Items in the archives can be searched for by name and/or by category using regular expressions (C++ regex). Naturally, item metadata can be examined without retrieving the item from the archive.
- C++ streams under the hood, with a special compressible streambuf.

Development Status

- Currently, the python API is available for about 3/4 of the NPStat functionality. The code is known to work on various Linux systems (Ubuntu, CentOS, Scientific Linux) in combination with recent anaconda python distributions.
- Detailed API documentation is limited at the moment, but a substantial number (26 and growing) of well-commented example scripts illustrate various features and algorithms.
- The source tarball can be downloaded from Hepforge (use the latest version). Please read the INSTALL file, as you will need to install prerequisites and to include a special "make python" step in order to compile the python API. An example script which installs NPStat and its dependencies into a user-owned directory on a typical Linux box is posted on Indico.
- Looking for collaborators experienced in distributing mixed C++/python packages for Windows/macOS.

Summary

- NPStat python API provides access to a number of modern nonparametric statistical techniques including several unique algorithm implementations.
- You might also like NPStat it for its histogramming tools and persistence mechanism. Give them a spin. Request features and contribute.
- While we will never be able to compete with the R community in terms of breadth of statistical methods available, python has other advantages. A good collection of HEP-oriented python tools should elevate us as the field from the depths of using C++ as the scripting language for data analysis.