PROCEEDINGS OF SCIENCE

SkyLLH - A generalized Python-based tool for log-likelihood analyses in multi-messenger astronomy

The IceCube Collaboration*
http://icecube.wisc.edu/collaboration/authors/icrc19_icecube
E-mail: martin.wolf@icecube.wisc.edu

Common analysis techniques in multi-messenger astronomy involve hypothesis tests with unbinned log-likelihood (LLH) functions using data recorded in celestial coordinates to identify sources of high-energy cosmic particles in the Universe. We present the new Python-based tool "SkyLLH" to develop such analyses in a telescope-independent framework. The main goal of the software is to provide an easy-to-use and modularized concept to implement and to execute such LLH functions efficiently on the computer with high-performance. SkyLLH can be applied on different multi-messenger data like neutrino and gamma-ray events from experiments such as the IceCube Neutrino Observatory and the Fermi-LAT. In this contribution we highlight SkyLLH's various design goals, current development status, and prospects for its wider application in multi-messenger astronomy.

Corresponding authors: Martin Wolf[†1]
[1] *Physik-department, Technische Universität München, D-85748 Garching, Germany*

36th International Cosmic Ray Conference -ICRC2019-
July 24th - August 1st, 2019
Madison, WI, U.S.A.

*For collaboration list, see PoS(ICRC2019) 1177.
†Speaker.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

arXiv:1908.05181v1 [astro-ph.IM] 14 Aug 2019

1. Introduction

In multi-messenger astronomy (MMA) the maximum log-likelihood (LLH) hypothesis ratio test (LRT) is a common analysis technique. Let Θ denote the entire allowed parameter space of the model with model parameters $\vec{\theta}$, the LRT is constituted of the LLH ratio test statistic, $\log\Lambda(\vec{D})$, for a null- and alternative hypothesis, $\mathscr{H}_0 : \vec{\theta} \in \Theta_0$, and $\mathscr{H}_1 : \vec{\theta} \in \Theta_0^c$, respectively:

$$\log\Lambda(\vec{D}) = \sup_{\vec{\theta}\in\Theta_0}\left\{\log\mathscr{L}(\vec{\theta}|\vec{D})\right\} - \sup_{\vec{\theta}\in\Theta}\left\{\log\mathscr{L}(\vec{\theta}|\vec{D})\right\}, \tag{1.1}$$

where $\log\mathscr{L}(\vec{\theta}|\vec{D})$ is the logarithm of the likelihood function with parameter set $\vec{\theta}$ for the given (observed) data $\vec{D}$. In the limit of large statistics the probability density function (PDF) of the test-statistic $\lambda(\vec{D}) = -2\log\Lambda(\vec{D})$, *i.e.* $f(\lambda)$, follows a χ^2_k-distribution with the number of degrees of freedom, k, equal to the number of free parameters in the test, if the parameter values are sufficiently far away from their bounds [1].

In MMA a usual task is to analyze data composed of independent events, so-called event data, with contributions from a signal and a background component. Hence, a two-component, *i.e.* signal and background, likelihood model is usually chosen. In such a model a likelihood function $\mathscr{L}(\vec{\theta}|\vec{D})$ can be constructed, that evaluates the observable PDFs for each component, *i.e.* signal and background, of each of the N data events, D_i:

$$\mathscr{L}(\vec{\theta}|\vec{D}) \equiv \mathscr{L}(n_\mathrm{s}, \vec{\theta}_\mathrm{s}|\vec{D}) = \prod_{i=1}^{N}\left[\frac{n_\mathrm{s}}{N}\mathscr{S}(D_i|\vec{\theta}_\mathrm{s}) + (1-\frac{n_\mathrm{s}}{N})\mathscr{B}(D_i)\right]. \tag{1.2}$$

Here, the parameter set $\vec{\theta}$ consists of the mean number of signal events in the data, n_s, and the parameters $\vec{\theta}_\mathrm{s}$ describing the signal source. $\mathscr{S}$ and $\mathscr{B}$ are the signal and background probability density functions for the N data events D_i, respectively.

In the special case, when assuming a negligible signal contribution, *e.g.* in a background-dominated experiment, the null-hypothesis can be defined as the data without any signal events, *i.e.* $\mathscr{H}_0 : n_{\mathrm{s},0} = 0$. Thus, the negative of the log-likelihood ratio test statistic for the two-component likelihood function (1.2) can be written as

$$-\log\Lambda(\vec{D}) = \sup_{n_\mathrm{s},\vec{\theta}_\mathrm{s}}\left\{\log\frac{\mathscr{L}(n_\mathrm{s},\vec{\theta}_\mathrm{s}|\vec{D})}{\mathscr{L}(n_\mathrm{s}=n_{\mathrm{s},0}=0|\vec{D})}\right\} = \sup_{n_\mathrm{s},\vec{\theta}_\mathrm{s}}\left\{\sum_{i=1}^{N}\log\left[\frac{n_\mathrm{s}}{N}\left(\frac{\mathscr{S}(D_i|\vec{\theta}_\mathrm{s})}{\mathscr{B}(D_i)}-1\right)+1\right]\right\}. \tag{1.3}$$

It should be noted that the PDF $f(\lambda)$ of the test statistic function $\lambda(\vec{D})$ does not follow a χ^2-distribution function under this null-hypothesis, because the parameter value for n_s lies on the boundary of the allowed parameter value space $[0,N]$. Instead, $f(\lambda)$ follows the super-position of a δ-function and a χ^2_k-distribution function [2]:

$$f(\lambda) = \frac{1}{2}\delta(\lambda) + \frac{1}{2}\chi^2_k(\lambda) \tag{1.4}$$

The SkyLLH software provides a framework for defining maximum LLH hypothesis ratio tests, to construct, and to evaluate LLH ratios as given in equation (1.1). It allows the user to define a statistical LRT analysis for a given set of data. Furthermore, the user can easily generate pseudo data trials to determine statistical properties of the analysis, *e.g.* sensitivity and discovery potential, in a frequentist approach.

2. The SkyLLH Framework

The SkyLLH framework is implemented entirely in the programming language Python[1] and is an open-source package licensed under the GPLv3 license[2]. It is available on the open source github repository[3] of the IceCube collaboration. The software package dependencies for SkyLLH are kept to an absolute minimum: *numpy*, *scipy*, and *astropy*. These packages are usually already available in most Python installations. In cases where they are not, they are at least easily available for installation on most platforms.

SkyLLH utilizes object-oriented-programming (OOP) techniques. The class structure is tied to the mathematical objects of the LLH ratio formalism. A fundamentally used mathematical object is the PDF. Thus, the Python abstract base class `core.PDF` exists and defines the interface of a PDF object. The most important method of this class is `PDF.get_prob`, which is supposed to return the probability for each data event D_i.

The code of the SkyLLH framework is structured into four distinct main Python modules: core, physics, plotting, and detector specific implementations and specializations. The `core` module contains all Python classes defining the base framework of SkyLLH. This includes the classes for all mathematical objects like PDFs, LLH ratio functions, and test statistic functions. The `physics` module provides classes defining the physics models of interest. Since the actual sources of high-energy cosmic rays, including photon and neutrino emission, remain unknown to-date, commonly used physics models are generic flux models for the messenger particle of interest that creates a signal in the particular detector. The `plotting` module contains auxiliary functionality for plotting particular objects of the framework, *e.g.* PDF objects. This is meant mainly for integrity checks of the analysis. Finally, the fourth part of modules provide detector specific implementations and specializations based on the classes defined in the `core` module. Currently this includes the `i3` module for the IceCube Neutrino Observatory [3], which is a cubic-kilometer neutrino detector installed in the ice at the geographic South Pole between depths of 1450 m and 2450 m, and was completed in 2010.

3. Source Hypothesis Definition

The first step of a LRT in MMA is the definition of the source hypothesis. Because the actual sources of high-energy cosmic rays in the Universe are still unknown, a common approach is to define a generic differential particle flux at Earth, Φ_S, from a source. Such a generic differential flux can be parameterized as

$$\frac{\mathrm{d}^4\Phi_\mathrm{S}(\alpha,\delta,E,t|\vec{\theta}_\mathrm{s})}{\mathrm{d}E\mathrm{d}\Omega\mathrm{d}A\mathrm{d}t}, \tag{3.1}$$

which is a function of the celestial coordinates right-ascention, α, and declination, δ, as well as the energy and time of the source (signal) particle, given the source parameters $\vec{\theta}_\mathrm{s}$. Ω and A denote the sky's solid-angle covered by the source, and the surface area of Earth covered by the flux,

[1] Both, Python2 and Python3 environments are supported. However, Python3 based developments of analyses, that utilize SkyLLH, is preferred.

[2] `https://www.gnu.org/licenses/gpl-3.0.txt`

[3] `https://github.com/IceCubeOpenSource/SkyLLH/`

respectively. Φ_S can describe a point-like source, like a Blazar, or an extended source, like the galactic plane of the Milky Way.

The IceCube Neutrino Observatory searches for point-like sources using a factorized flux model as source hypothesis:

$$\frac{\mathrm{d}^4\Phi_\mathrm{S}(\alpha,\delta,E,t|\vec{\theta}_\mathrm{s})}{\mathrm{d}E\mathrm{d}\Omega\mathrm{d}A\mathrm{d}t} = \Phi_0\Psi_\mathrm{S}(\alpha,\delta|\vec{\theta}_\mathrm{s})\varepsilon_\mathrm{S}(E|\vec{\theta}_\mathrm{s})T_\mathrm{S}(t|\vec{\theta}_\mathrm{s}), \tag{3.2}$$

where Φ_0 is the flux normalization carrying the differential flux units, and Ψ_S, ε_S, and T_S are the spatial, energy, and time profiles of the source, respectively. For a point-like source at the celestial location $\vec{r}_\mathrm{s} = (\alpha_\mathrm{s}, \delta_\mathrm{s})$ the spatial profile collapses to a Dirac-δ-function:

$$\Psi_\mathrm{S}(\vec{r}|\vec{\theta}_\mathrm{s}) = \delta(\vec{r} - \vec{r}_\mathrm{s}), \tag{3.3}$$

with $\vec{r} = (\alpha, \delta)$ being the celestial coordinate vector. As energy profile $\varepsilon_\mathrm{S}(E|\vec{\theta}_\mathrm{s})$ a common choice is the power-law function

$$\varepsilon_\mathrm{S}(E|\vec{\theta}_\mathrm{s}) = \left(\frac{E}{E_0}\right)^{-\gamma} \tag{3.4}$$

with spectrial index γ and reference energy E_0. The time profile $T_\mathrm{S}(t|\vec{\theta}_\mathrm{s})$ is usually choosen to be box-shaped or gaussian-shaped. For a steady emitting source the time profile is unity.

SkyLLH provides the abstract base class `physics.flux_models.FluxModel` for the generic differential flux given by (3.1). Utilizing the `units` module of the *astropy* package, this class supports the specification of individual energy, angle, length, and time units for the flux, as well as the their conversion to the internally used flux unit of GeV^{-1} sr^{-1} cm^{-2} s^{-1}. In addition `FactorizedFluxModel` provides a class for a factorized flux model as given in equation (3.2). Individual spatial, energy, and time flux profiles can be designed through the abstract base class `FluxProfile`.

In case the LRT consists of several sources, it is useful, from a computational point-of-view, to group sources with the same flux model into source hypothesis groups. Each group of sources will share the same implementation for the flux model, the mean number of expected signal events in the detector, *i.e.* the detector signal yield, and the signal event generation method. Calculations for those quantities can then be parallelized for all sources of a source hypothesis group. SkyLLH provides the `SourceHypoGroup` and the `SourceHypoGroupManager` classes to define and to mange those groups of source hypotheses.

4. Data Management Concepts

An important utility feature of SkyLLH is the ability to pre-define common data sets that can be used by different analyses. To accommodate this feature, the `core.dataset` module provides the `Dataset` class. An instance of the `Dataset` class describes a particular data set. The general enfolded properties of a data set are its name, the location on disk of the experimental and simulation data, as well as its live-time. Data can be stored on disk in various data formats. Depending on the file name extension of the data files, SkyLLH selects a specific file loader class for a particular data file format. The list of file loader classes is extendable by the user in order

to be able to support user-specific data file formats. Sometimes a data sample is split into several individual data sets. For instance when an event selection has to be performed on individual detector configurations or calibrations, resulting into separate detector responses for each data set. In SkyLLH such individual data sets can be grouped into a collection of data sets by using an instance of the `core.dataset.DatasetCollection` class. In addition to the data set properties listed above, the `Dataset` class supports the version and version qualifiers properties to allow for a documented evolution of data sets. This simplifies the reproduce-ability of the results of analyses.

After the pre-definition of data sets and collection of data sets, *i.e.* data samples, such data sets can be chosen by an analysis. For each data set the experimental, simulation, and possible auxiliary data can be loaded with a single call to the `load_and_prepare_data` method of the `Dataset` class. This class method applies possible data preparation functions to achieve uniformity in available data fields throughout the data sets. These data preparation functions also allow for analysis related data selection cuts. The loaded data is then stored in an instance of the data holder class `DatasetData` and can be used for creating PDF objects, background and signal event generator instances.

Internally, the data is stored as *numpy* arrays. A dedicated class named `core.storage.DataFieldArray` has been developed for SkyLLH to store the data fields of a data set as one-dimensional `numpy.ndarray` objects, whereas the interface of this class mimics the one of a structured ndarray object. Because numerical operations for the analysis are usually performed on a single data field for all events of the data set at once, data access is more efficient on one-dimensional ndarrays, where field data of consecutive events are stored continuously in memory, compared to structured ndarrays where entire event records are stored continuously in memory, resulting into a much larger memory footprint when traversing a single data field for all events. By using `DataFieldArray` as internal data holder, memory cache misses can be reduced, causing faster data access and shorter program execution times.

Another unique data management concept of SkyLLH is the management of trial data. For each analysis trial new pseudo event data has to get generated, which is based, in some analysis defined way, on the loaded data from the pre-defined data sets. The likelihood function $\mathscr{L}(\vec{\theta}|\vec{D})$ is always evaluated on the trial data. In the case of data unblinding, the trail data equals the experimental data. In SkyLLH the `core.trialdata.TrialDataManager` class provides a manager for the event data of a trial. The manager allows to define additional event data fields and their calculation based on the data fields of the data set and previously defined data fields for the trial data manager. `TrialDataManager` data fields have assigned possible dependencies on source properties and fit parameters. Hence, only those data fields have to be recalculated, whose dependencies were updated during the LLH function maximization process and the data $\vec{D}$ for the LLH function $\log\mathscr{L}(\vec{\theta}|\vec{D})$ is requested solely through the `TrialDataManager` instance of the data set.

5. Hypothesis Parameter Definition

For the LRT the definition of the parameter set Θ is essential. In general it can contain fixed and floating, *i.e.* fit, parameters $\vec{\theta}$. For the signal-and-background two-component likelihood model, a parameter θ belongs either to a signal, a background, or some other model, for instance a nuisance

model. We denote the parameter θ as a global parameter and define a local parameter, $\tilde{\theta}$, as a parameter of a specific model. In general the signal component can consist of multiple contributions, *e.g.* multiple individual point-like neutrino emitting sources in the Universe. Each signal model would represent a single source, modeled for instance by a neutrino flux as given in equation (3.2). Such a scenario is commonly referred to as "stacking of sources".

A specific model has a distinct set of parameters with their names. Hence, in the special case of a multi point-like source scenario the flux model, as given in equation (3.2) with the energy profile of a power-law (*c.f.* eq. (3.4)), has the spectral index (local) parameter $\tilde{\gamma}$, named "gamma", defined several times, once for each source. It raises the question how these local parameters should be handled in the global likelihood maximization process. Obviously, this question is of hypothesis nature. Depending on the source hypothesis these local spectral index parameters should all be independent global fit parameters, or should all share the same single global spectral index parameter, or should be grouped into groups of shared global spectral index parameters. SkyLLH provides the `ModelParameterMapper` class that takes a global parameter and assigns it to a local parameter of one or more models. Hence, global fit parameters can be assigned to local model parameters in a highly flexible manner.

6. Application of SkyLLH in Multi-Messenger Astronomy

SkyLLH is being developed within the IceCube collaboration as a standard tool to search for neutrino emitting sources in the Universe. The implementation of generalized concepts in terms of source hypothesis and hypothesis parameter definition makes it easy to use the SkyLLH framework also for searches of other messenger particles in other experiments. Whenever a LRT as given in equation (1.1) with celestial data has to be performed, SkyLLH is a suitable tool. Possible future applications of SkyLLH could be combined analyses of same-kind messenger particle data, for instance from different neutrino telescopes like IceCube and ANTARES [4] / KM3NeT [5], or of different messenger particle data of neutrinos and gamma-rays, for instance from IceCube and Fermi/LAT [6].

References

[1] S. S. Wilks, *Annals Math. Statist.* **9** (1938) 60–62.

[2] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, *Eur. Phys. J.* **C71** (2011) 1554. [Erratum: Eur. Phys. J.C73,2501(2013)].

[3] **IceCube** Collaboration, M. G. Aartsen et al., *JINST* **12** (2017) P03012.

[4] **ANTARES** Collaboration, M. Ageron et al., *Nucl. Instrum. Meth.* **A656** (2011) 11.

[5] **KM3Net** Collaboration, S. Adrian-Martinez et al., *J. Phys.* **G43** (2016) 084001.

[6] **Fermi/LAT** Collaboration, W. B. Atwood et al., *Astrophys. J.* **697** (2009) 1071.