pyuds 1.0 User's Guide

A Python library for measuring uncertainty in Dempster-Shafer theory of evidence

Sari Haj Hussein (`http://www.sarihh.info`)

2011-11-07

Contents

1 Introduction

1.1 What is pyuds?

pyuds is a Python library for measuring uncertainty in Dempster-Shafer theory of evidence. The functionals supported are Generalized Hartley (GH) uncertainty functional, Generalized Shannon (GS) uncertainty functional, and Aggregate Uncertainty (AU) functional. The library can be utilized either through its API, or through a user-friendly web interface.

1.2 Some Intended Purposes

Here are some example usages of pyuds:

- Experts, who are familiar with Dempster-Shafer theory, use it to measure uncertainty while applying this theory to their fields of study.
- People, who are unfamiliar with Dempster-Shafer theory, use it to reach firm understanding while learning this theory.

1.3 Features

Following are the features provided by pyuds:

- pyuds supports computing all the renown uncertainty functionals in Dempster-Shafer theory. This includes Generalized Hartley (GH) uncertainty functional, Generalized Shannon (GS) uncertainty functional, and Aggregate Uncertainty (AU) functional.
- Since computing Aggregate Uncertainty (AU) involves a recursive algorithm, pyuds facilitates a verbose mode that enables users to view the output of each recursive call.
- The pyuds API is available for any operating system with Python 3.0 or higher installed.
- The pyuds web interface is sent in undecorated HTML to the client and therefore accessible by any web browser. The server-side CGI script is written in

pure Python and therefore executable by any web server with an embedded Python 3.0 or higher interpreter, for instance, Apache HTTP Server with the module mod_python (`http://www.modpython.org/`) installed. For testing purposes, one can also rely on the most basic CGI web server that comes with Python 3.0 or higher.

1.4 Open Source Software

pyuds is an open source software project, and is released under the GNU General Public License (GPL). You can freely use pyuds on any number of computers you like, without worrying about license keys or fees or such. In addition, all the source code is freely available under the GPL. Because of that, it is very easy for people to contribute to the project.

1.5 What pyuds is Not?

- pyuds does not implement all the concepts in Dempster-Shafer theory.
- pyuds supports computing uncertainty functionals only in Dempster-Shafer theory, not in any other parallel theory.

1.6 Supported Operating Systems

The pyuds API should run on any operating system with Python 3.0 or higher installed. The pyuds web interface should run if served by any web server with an embedded Python 3.0 or higher interpreter.

1.7 Where to Get pyuds?

You can get the latest copy of pyuds from its website (`http://sourceforge.net/projects/pyuds/`).

1.8 Development and Maintenance of pyuds

pyuds was developed by Sari Haj Hussein. Ongoing development and maintenance of pyuds are also handled by Sari Haj Hussein. pyuds is an open source software project, and is released under the GNU General Public License (GPL). All the source

code is freely available under the GPL. You are welcome to modify pyuds to suit your own needs, and it would be appreciated if you contribute your improvements back to us. The pyuds source code is available from pyuds website (`http://sourceforge.net/projects/pyuds/`).

1.9 Reporting Problems and Getting Help

If you have problems, or need help with pyuds, please send an email describing the issue to angyjoe@gmail.com.

2 Installing pyuds

2.1 Introduction

To use pyuds, you must obtain its binary package, then install it to its final destinations.

2.2 Obtaining the Binary Package

You can obtain the binary package from pyuds website (`http://sourceforge.net/projects/pyuds/`). It should be named something like: pyuds<version>.zip.

2.3 Installing pyuds on Windows/Linux

Unpack pyuds binary package into a directory of choice.

2.4 Uninstalling pyuds

Remove the directory where you unpacked pyuds binary package.

3 Uncertainty in Dempster-Shafer Theory

3.1 Introduction

This chapter includes a primer on representing and measuring uncertainty in Dempster-Shafer theory.

3.2 Representing Uncertainty

There is a good number of methods of representing uncertainty, and the starting point for most of them is a set of possible worlds, states, or elementary outcomes that an agent considers possible. This set is called a frame of discernment (a frame for short). For instance, in the crude guessing of commonly used passwords, as agent could consider the following set possible:

$$\mathcal{W} = \{\text{password, 123456, qwerty, abc123, letmein, monkey, 696969, blink182}\}$$

A proposition an agent supports (considers possible) is just a set of possible worlds. For instance, a proposition could be "the password is numeric", and it would correspond to the set {123456, 696969}. This set qualitatively measures the agent's uncertainty. Intuitively, the more worlds an agent considers possible, the more uncertain she is, and the less she knows. A frame is a very coarse-grained representation of uncertainty, since we do not have any means of comparing the likelihood of two worlds. Many fine-grained representations of uncertainty that allow such comparison have been considered in the literature, and they all have relative advantages and disadvantages. Of these representations, we are concerned with Dempster-Shafer theory.

3.3 Dempster-Shafer Theory

Dempster-Shafer theory is a mathematical theory of evidence that allows a numeric representation of uncertainty. This representation is capable of modeling the evolution (or regression) of agent's knowledge about a system as more and more pieces

of evidence become available. This modeling is facilitated by the ability to combine and condition pieces of evidence in this theory.

3.3.1 Capturing Agent's Belief

An agent's belief is based on a piece of evidence that substantiates some degree of support (possibly 0) to each subset of a frame. The total amount of support is 1. In the framework of Dempster-Shafer theory, this belief is captured using a mass function, which is defined as follows.

Definition 1 (Mass Function) Let $\mathcal{W}$ be a frame. A mass function on $\mathcal{W}$ is a function of the form $m : \mathcal{P}(\mathcal{W}) \rightarrow [0, 1]$ where $\mathcal{P}(\mathcal{W})$ is the first-order power set of $\mathcal{W}$ defined as $\mathcal{P}(\mathcal{W}) = \{X | X \subseteq \mathcal{W}\}$. This function satisfies the following two properties:

1. $m(\emptyset) = 0$
2. $\sum_{A \in \mathcal{P}(\mathcal{W})} m(A) = 1$

For all $A \in \mathcal{P}(\mathcal{W})$, the value $m(A)$ has the following meaning; it characterizes the degree of belief that the true world is in the set A, but it does not take into account any additional evidence for the various subsets of A. Each set $X \in \mathcal{P}(\mathcal{W})$ such that $m(X) > 0$ is called a focal set of m. We denote the set of all focal sets induced by m as $\mathcal{F}_m$, and write:

$$\mathcal{F}_m = \{X \in \mathcal{P}(\mathcal{W}) | m(X) > 0\}$$

In Dempster-Shafer theory, it is possible to bind the pieces of evidence together and obtain a belief function $Bel : \mathcal{P}(\mathcal{W}) \rightarrow [0, 1]$ from the mass function m using the formula:

$$Bel(A) = \sum_{B \subseteq A} m(B)$$

This transformation is reversible using the formula:

$$m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} Bel(B)$$

It can be shown that a one-to-one correspondence (bijectivity) exists between mass and belief functions.

3.3.2 Measuring Agent's Uncertainty

In Dempster-Shafer theory, two types of uncertainty coexist:

1. The nonspecificity in our prediction about the true world in a frame.
2. The conflict between the pieces of evidence expressed by each mass value.

To measure nonspecificity in Dempster-Shafer theory, we use Generalized Hartley (GH) uncertainty functional, which is given in Definition 2.

Definition 2 (Generalized Hartley (GH) Uncertainty Functional) Let $\mathcal{W}$ be a frame, $m : \mathcal{P}(\mathcal{W}) \rightarrow [0, 1]$ be a mass function on $\mathcal{W}$, and $\mathcal{F}_m$ be the set of all focal sets induced by m. The nonspecificity uncertainty about the true world in $\mathcal{W}$ is given by the functional:

$$GH(m) = \sum_{A \in \mathcal{F}_m} m(A) log|A|$$

To aggregatedly measure both nonspecificity and conflict in Dempster-Shafer theory, we use the Aggregate Uncertainty (AU) functional, which is given in Definition 3.

Definition 3 (Aggregate Uncertainty (AU) Functional) Let $\mathcal{W}$ be a frame, $Bel : \mathcal{P}(\mathcal{W}) \rightarrow [0, 1]$ be a belief function on $\mathcal{W}$. The aggregate uncertainty about the true world in $\mathcal{W}$ is given by the functional:

$$AU(Bel) = \underset{\mathcal{P}_{Bel}}{max} \left\{ - \sum_{x \in \mathcal{W}} p(x) \log p(x) \right\}$$

where $\mathcal{P}_{Bel}$ is the set of all probability distribution functions that dominate Bel by satisfying the following two properties:

1. $p(x) \in [0, 1]$ for all $x \in \mathcal{W}$ and $\sum_{x \in \mathcal{W}} p(x) = 1$
2. $Bel(A) \leq \sum_{x \in A} p(x)$ for all $A \in \mathcal{P}(\mathcal{W})$

A recursive algorithm for computing AU is given in Algorithm 1.

Algorithm 1 (Computing AU)

Input: A frame $\mathcal{W}$ and a belief function $Bel : \mathcal{P}(\mathcal{W}) \rightarrow [0, 1]$ on $\mathcal{W}$.

Output: $AU(Bel)$ as given in Definition 3.

1. Find a nonempty set $A \in \mathcal{P}(\mathcal{W})$ such that $Bel(A)/|A|$ is maximal. If more than one set achieve a maximal $Bel(A)/|A|$, assume the set that has the highest cardinality.
2. For all $x \in A$, put $p(x) = Bel(A)/|A|$.

3. For each $B \subseteq \mathcal{W} - A$, put $Bel(B) = Bel(B \cup A) - Bel(A)$.
4. Put $\mathcal{W} = \mathcal{W} - A$.
5. If $\mathcal{W} \neq \emptyset$ and $Bel(\mathcal{W}) > 0$, go to step 1.
6. If $\mathcal{W} \neq \emptyset$ and $Bel(\mathcal{W}) = 0$, put $p(x) = 0$ for all $x \in \mathcal{W}$.
7. Compute $AU(Bel) = -\sum_{x \in \mathcal{W}} p(x) \log p(x)$.

To measure conflict in Dempster-Shafer theory, we use Generalized Shannon (GS) uncertainty functional, which is given in Definition 4.

Definition 2 (Generalized Shannon (GS) Uncertainty Functional) Let $\mathcal{W}$ be a frame, $m : \mathcal{P}(\mathcal{W}) \to [0, 1]$ be a mass function on $\mathcal{W}$, and $Bel : \mathcal{P}(\mathcal{W}) \to [0, 1]$ be the corresponding belief function on $\mathcal{W}$. The conflict uncertainty about the true world in $\mathcal{W}$ is given by the functional:

$$GS(m) = AU(Bel) - GH(m)$$

where $GH(m)$ and $AU(Bel)$ are respectively given in Definitions 2 and 3.

4 pyuds API

4.1 Introduction

This chapter describes the pyuds API. The purpose of this description is to aid developers in incorporating pyuds into their (presumably larger) reasoning applications. pyuds API requires Python 3.0 or higher installed.

4.2 API functions

The pyuds API is comprised of a single class named 'pyuds', which is contained in the file named 'pyuds.py' in the pyuds binary package. This class has three static methods that are described in the following subsection.

4.2.1 GH(m)

Purpose: This method returns the value of Generalized Hartley (GH) uncertainty functional associated with the mass function 'm'. Please refer to Section 3.3.2 for more information on this functional.

Arguments: 'm' is the mass function for which GH will be computed.

4.2.2 GS(m)

Purpose: This method returns the value of Generalized Shannon (GS) uncertainty functional associated with the mass function 'm'. Please refer to Section 3.3.2 for more information on this functional.

Arguments: 'm' is the mass function for which GS will be computed.

4.2.3 AU(m, verbose=False)

Purpose: This method returns the value of Aggregate Uncertainty (AU) functional associated with the mass function 'm'. Please refer to Section 3.3.2 for more information on this functional.

Arguments:

- 'm' is the mass function for which AU will be computed.
- 'verbose' enables/disables verbose mode (default False). If verbose mode is enabled, users can view the output of each recursive call in the recursive AU algorithm.

4.3 Supplying a Mass Function

A mass function (given in Definition 1 in Section 3.3.1) is supplied to the pyuds API either as a Python dictionary, or as a list of tuples. For instance, given a frame $\mathcal{W} = \{a, b, c, d\}$, the following two code blocks supply the same mass function to the pyuds API:

```
m = MassFunction({
    "a": 0.26,
    "b": 0.26,
    "c": 0.26,
    "ab": 0.07,
    "ac": 0.01,
    "ad": 0.01,
    "bc": 0.01,
    "bd": 0.01,
    "cd": 0.01,
    "abcd": 0.1
})
```

```
m = MassFunction([
    ({'a'}, 0.26),
    ({'b'}, 0.26),
    ({'c'}, 0.26),
    ({'a', 'b'}, 0.07),
    ({'a', 'c'}, 0.01),
    ({'a', 'd'}, 0.01),
    ({'b', 'c'}, 0.01),
    ({'b', 'd'}, 0.01),
    ({'c', 'd'}, 0.01),
    ({'a', 'b', 'c', 'd'}, 0.1)
])
```

4.4 Example API Usage

For an example on pyuds API usage, see the file 'example.py' in the pyuds binary package.

5 pyuds Web Interface

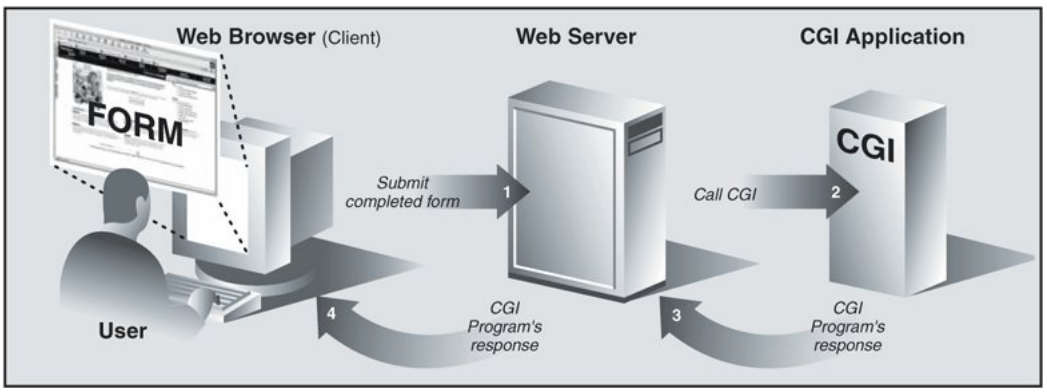
5.1 Introduction

This chapter describes the pyuds web interface. This interface is meant for end users who want to get their computations done without fussing much with the technical details of the library.

5.2 How Does it Work?

For an overview of how the pyuds web interface works, see Figure 1.

Figure 1. An overview of how pyuds web interface works

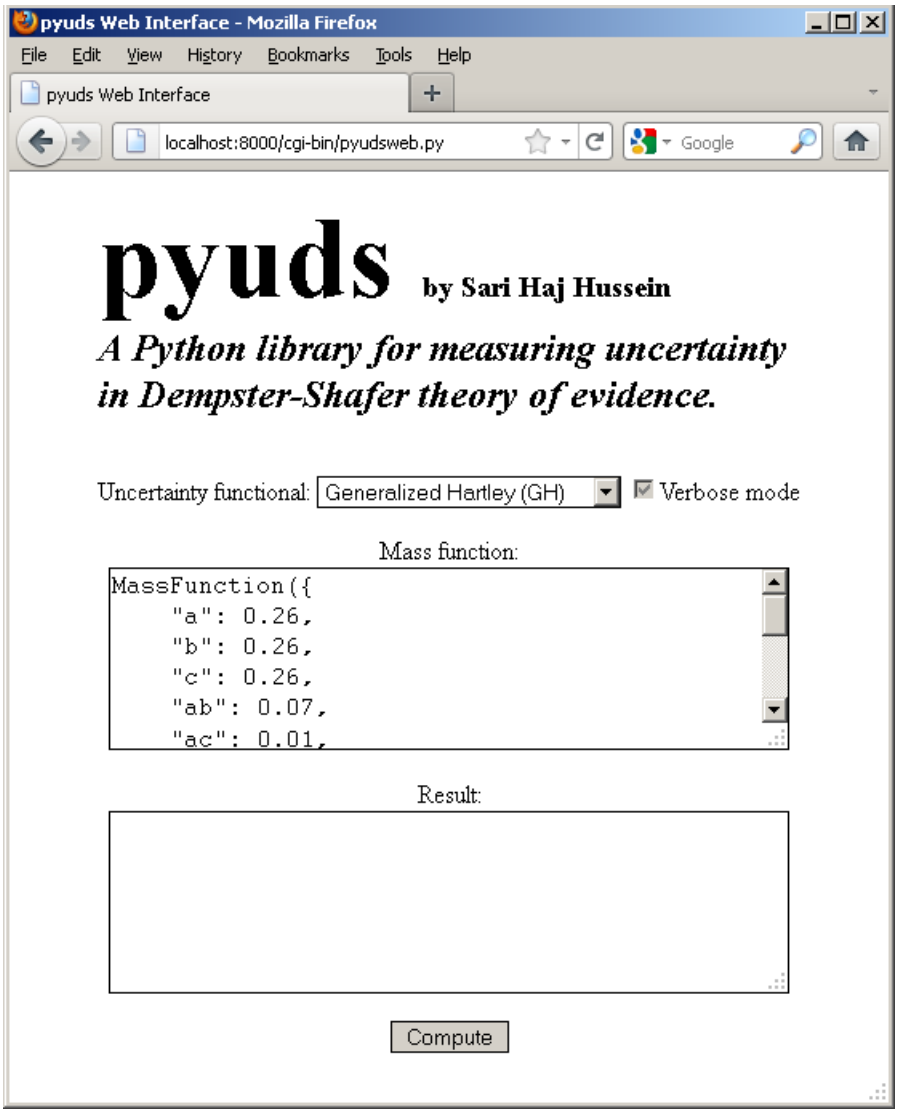


The pyuds web interface is sent in undecorated HTML to the client and therefore accessible by any web browser. The server-side CGI script is written in pure Python and therefore executable by any web server with an embedded Python 3.0 or higher interpreter, for instance, Apache HTTP Server with the module mod_python (`http://www.modpython.org/`) installed. For testing purposes, one can also rely on the most basic CGI web server that comes with Python 3.0 or higher.

5.3 Starting the pyuds Web Interface

First, you start the CGI web server that comes with Python 3.0 or higher using the file 'startCGIServer.bat' on Windows, or the file 'startCGIServer.sh' on Linux. Both files are included in the pyuds binary package. Next, you navigate to the link `http://localhost:8000/cgi-bin/pyudsweb.py`, either by typing it in your web browser address bar, or using the link file 'Start pyuds Web Interface' in the pyuds binary package. You should see the pyuds web interface as in Figure 2.

Figure 2. The pyuds web interface



5.4 Using the pyuds Web Interface

Using the pyuds web interface should be straightforward. You need to choose the uncertainty functional you want to compute, and supply the mass function for which the computation will be done. Finally hit the 'Compute' button, and expect the result to show in the 'Result' textarea. Notice that the 'Verbose mode' checkbox is only enabled when computing AU.

5.5 Sample Results

Computing GH for the sample mass function given yields the following result:

```
Generalized Hartley (GH) = 0.32000000000000006
```

Computing GS for the sample mass function given yields the following result:

```
Generalized Shannon (GS) = 1.6159802491167672
```

Computing AU for the sample mass function given in non-verbose mode yields the following result:

```
Aggregate Uncertainty (AU) = 1.9359802491167672
```

Computing AU for the sample mass function given in verbose mode yields the following result:

```
Iteration 1
=============
Step 1
======
```

A	Bel(A)	Bel(A)/|A|
['a', 'b', 'c']	0.87	0.29
['d']	0.0	0.0
['b']	0.26	0.26
['a']	0.26	0.26
['b', 'c', 'd']	0.55	0.18333333333333335
['b', 'd']	0.27	0.135
['a', 'd']	0.27	0.135
['a', 'c', 'd']	0.55	0.18333333333333335
['a', 'b', 'c', 'd']	1.0	0.25
['a', 'b', 'd']	0.61	0.20333333333333334
['b', 'c']	0.53	0.265
['a', 'c']	0.53	0.265
['c', 'd']	0.27	0.135
['c']	0.26	0.26

```
['a', 'b']              0.5900000000000001   0.29500000000000004
Maximal = 0.29500000000000004
Highest cardinality = ['a', 'b']
Step 2
======
Probabilities list = [0.29500000000000004,
                      0.29500000000000004]
Step 3
======
A           Bel(A)                  Bel(A)/|A|
['d']       0.019999999999999907    0.0
['c', 'd']  0.4099999999999999      0.135
['c']       0.2799999999999999      0.26
Step 4
======
frame = ['c', 'd']
Step 5
======
Iterating the algorithm...

Iteration 2
===========
Step 1
======
A           Bel(A)   Bel(A)/|A|
['d']       0.019999999999999907   0.019999999999999907
['c', 'd']  0.4099999999999999     0.20499999999999996
['c']       0.2799999999999999     0.2799999999999999
Maximal = 0.2799999999999999
Highest cardinality = ['c']
Step 2
======
Probabilities list = [0.29500000000000004,
                      0.29500000000000004,
                      0.2799999999999999]
Step 3
======
A       Bel(A)   Bel(A)/|A|
['d']   0.13     0.019999999999999907
Step 4
======
frame = ['d']
Step 5
======
```

```
Iterating the algorithm...

Iteration 3
==========
Step 1
======
A       Bel(A)   Bel(A)/|A|
['d']   0.13     0.13
Maximal = 0.13
Highest cardinality = ['d']
Step 2
======
Probabilities list = [0.29500000000000004,
                      0.29500000000000004,
                      0.2799999999999999,
                      0.13]
Step 3
======
A       Bel(A)   Bel(A)/|A|
Step 4
======
frame = []

Aggregate Uncertainty (AU) = 1.9359802491167672
```

6 pyuds License

The GNU General Public License (GPL) Version 3 (`http://www.gnu.org/licenses/gpl-3.0.html`) explains all the things that you are allowed to do with pyuds code and documentation.

7 pyuds Code Disclaimer

The author of pyuds code has used his best efforts in preparing the code. These efforts include the development, research, testing, and optimization of the theories and programs to determine their effectiveness. pyuds code is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Author disclaims any express or implied warranty of fitness for such uses. The author makes no warranty of any kind, expressed or implied, with regard to pyuds code or to the documentation accompanying it. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption whatsoever) arising out of, the furnishing, performance, or use of pyuds code, even if advised of the possibilities of such damages.