

---

# LSHash Documentation

*Release latest*

August 04, 2015



<b>1</b>	<b>Highlights</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Quickstart</b>	<b>7</b>
<b>4</b>	<b>Main Interface</b>	<b>9</b>



**Version** 0.0.4dev

A fast Python implementation of locality sensitive hashing with persistence support.



---

### Highlights

---

- Fast hash calculation for large amount of high dimensional data through the use of *numpy* arrays.
- Built-in support for persistency through Redis.
- Multiple hash indexes support.
- Built-in support for common distance/objective functions for ranking outputs.





---

# Installation

---

LShash depends on the following libraries:

- numpy
- redis (if persistency through Redis is needed)
- bitarray (if hamming distance is used as distance function)

To install:

```
$ pip install lshash
```



---

## Quickstart

---

To create 6-bit hashes for input data of 8 dimensions:

```
>>> from lshash import LSHash

>>> lsh = LSHash(6, 8)
>>> lsh.index([1,2,3,4,5,6,7,8])
>>> lsh.index([2,3,4,5,6,7,8,9])
>>> lsh.index([10,12,99,1,5,31,2,3])
>>> lsh.query([1,2,3,4,5,6,7,7])
[((1, 2, 3, 4, 5, 6, 7, 8), 1.0),
 ((2, 3, 4, 5, 6, 7, 8, 9), 11)]
```



---

## Main Interface

---

- To initialize a LSHash instance:

```
LSHash(hash_size, input_dim, num_of_hashtables=1, storage=None, matrices_filename=None, overwrite=False)
```

parameters:

**hash\_size:** The length of the resulting binary hash.

**input\_dim:** The dimension of the input vector.

**num\_hashtables = 1:** (optional) The number of hash tables used for multiple lookups.

**storage = None:** (optional) Specify the name of the storage to be used for the index storage. Options include “redis”.

**matrices\_filename = None:** (optional) Specify the path to the .npz file random matrices are stored or to be stored if the file does not exist yet

**overwrite = False:** (optional) Whether to overwrite the matrices file if it already exist

- To index a data point of a given LSHash instance, e.g., `lsh`:

```
lsh.index(input_point, extra_data=None):
```

parameters:

**input\_point:** The input data point is an array or tuple of numbers of `input_dim`.

**extra\_data = None:** (optional) Extra data to be added along with the `input_point`.

- To query a data point against a given LSHash instance, e.g., `lsh`:

```
lsh.query(query_point, num_results=None, distance_func="euclidean"):
```

parameters:

**query\_point:** The query data point is an array or tuple of numbers of `input_dim`.

**num\_results = None:** (optional) The number of query results to return in ranked order. By default all results will be returned.

**distance\_func = "euclidean":** (optional) Distance function to use to rank the candidates. By default euclidean distance function will be used.