FLASH

Generated by Doxygen 1.8.14

Contents

Chapter 1

FLASH

FLASH (Fast LSH Algorithm for Similarity Search Accelerated with HPC) is a library for large scale approximate nearest neighbor search of sparse vectors. It is currently available in C++ for CPU parallel computing and supports OpenCL enabled GPGPU computing. See `our paper` for theoretical and benchmarking details.

License

This library is licensed under Apache-2.0. See LICENSE for more details.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 LSH Class Reference

Public Member Functions

- void getHash (cl_mem ∗hashIndices_obj, cl_mem ∗identity_obj, cl_mem ∗dataIdx_obj, cl_mem ∗dataVal_↩ obj, cl_mem ∗dataMarker_obj, int numInputEntries, int numProbes)
- void getHash (unsigned int ∗hashIndices, unsigned int ∗identity, int ∗dataIdx, float ∗dataVal, int ∗dataMarker, int numInputEntries, int numProbes)
- void getHash (cl_mem ∗hashIndices_obj, cl_mem ∗identity_obj, cl_mem ∗input_obj, int numInputEntries, int numProbes)
- void getHash (unsigned int ∗hashIndices, unsigned int ∗identity, float ∗input, int numInputEntries, int num↩ Probes)
- LSH (int hashType, int numHashPerFamily, int numHashFamilies, int dimension, int samFactor)
- LSH (int hashType, int _K_in, int _L_in, int _rangePow_in)
- void clLSH (cl_platform_id ∗platforms_lsh, cl_device_id ∗devices_lsh, cl_context context_lsh, cl_program program_lsh, cl_command_queue command_queue_lsh)
- ∼LSH ()

3.1.1 Constructor & Destructor Documentation

3.1.1.1 LSH() `[1/2]`

```
LSH::LSH (
            int hashType,
            int numHashPerFamily,
            int numHashFamilies,
            int dimension,
            int samFactor )
```

Constructor.

Construct an LSH class for signed random projection.

Parameters

hashType	For SRP, use 1.
numHashPerFamily	Number of hash (bits) per hashfamily (hash table).
numHashFamilies	Number of hash families (hash tables).
dimension	Dimensionality of input data.
samFactor	samFactor = dimension / samSize, have to be an integer.

3.1.1.2 LSH() [2/2]

```
LSH::LSH (
            int hashType,
            int _K_in,
            int _L_in,
            int _rangePow_in )
```

Constructor.

Construct an LSH class for optimal densified min-hash (for more details refer to Anshumali Shrivastava, anshu@rice.edu). This hashing scheme is for very sparse and high dimensional data stored in sparse format.

Parameters

hashType	For optimal densified min-hash, use 2.

3.1.1.3 ∼LSH()

```
LSH::∼LSH ( )
```

Destructor.

Does not uninitialize the OpenCL environment (if applicable).

3.1.2 Member Function Documentation

3.1.2.1 clLSH()

```
void LSH::clLSH (
            cl_platform_id * platforms_lsh,
            cl_device_id * devices_lsh,
            cl_context context_lsh,
            cl_program program_lsh,
            cl_command_queue command_queue_lsh )
```

Initializes LSH hashing with OpenCl environment.

Takes in initialized OpenCL objects to support OpenCL hash functions, given that the particular type of hashing instantiated have an OpenCL implementation. Otherwise an error will be triggered during hashing.

Parameters

platforms_lsh	Reference to OpenCL cl_platform_id.
devices_lsh	Reference to OpenCL cl_device_id.
contect_lsh	Reference to OpenCl context_lsh.
program_lsh	Reference to OpenCL program_lsh.
command_queue_lsh	Reference to OpenCL cl_command_queue.

3.1.2.2 getHash() [1/4]

```
void LSH::getHash (
            cl_mem * hashIndices_obj,
            cl_mem * identity_obj,
            cl_mem * dataIdx_obj,
            cl_mem * dataVal_obj,
            cl_mem * dataMarker_obj,
            int numInputEntries,
            int numProbes )
```

Obtain hash indice given the (sparse) input vector, using OpenCL.

Hash indice refer to the corresponding "row number" in a hash table, in the form of unsigned integer. This function will only be valid when an OpenCL implementation exists for that type of hashing, and when OpenCL is initialized (clLSH) for that hashing. The outputs indexing is defined as hashIndicesOutputIdx(numHashFamilies, numProbes, numInputs, inputIdx, probeIdx, tb) (unsigned)(numInputs * numProbes * tb + inputIdx * numProbes + probeIdx).

Parameters

hashIndices_obj	Hash indice for the batch of input vectors.
identity_obj	Hash indice's corresponding identifications (sequential number starting 0) for this batch of inputs.
dataIdx_obj	Non-zero indice of the sparse format.
dataVal_obj	Non-zero values of the sparse format.
dataMarker_obj	Marks the start index of each vector in dataIdx and dataVal.
numInputEntries	Number of input vectors.
numProbes	Number of probes per input.

3.1.2.3 getHash() [2/4]

```
void LSH::getHash (
            unsigned int * hashIndices,
            unsigned int * identity,
            int * dataIdx,
            float * dataVal,
            int * dataMarker,
            int numInputEntries,
            int numProbes )
```

Obtain hash indice given the (sparse) input vector, using CPU.

Hash indice refer to the corresponding "row number" in a hash table, in the form of unsigned integer. This function will only be valid when an CPU implementation exists for that type of hashing. The outputs indexing is defined as hashIndicesOutputIdx(numHashFamilies, numProbes, numInputs, inputIdx, probeIdx, tb) (unsigned)(numInputs ∗ numProbes ∗ tb + inputIdx ∗ numProbes + probeIdx).

Parameters

hashIndices	Hash indice for the batch of input vectors.
identity	Hash indice's corresponding identifications (sequential number starting 0) for this batch of inputs.
dataIdx	Non-zero indice of the sparse format.
dataVal	Non-zero values of the sparse format.
dataMarker	Marks the start index of each vector in dataIdx and dataVal.
numInputEntries	Number of input vectors.
numProbes	Number of probes per input.

3.1.2.4 getHash() [3/4]

```
void LSH::getHash (
            cl_mem * hashIndices_obj,
            cl_mem * identity_obj,
            cl_mem * input_obj,
            int numInputEntries,
            int numProbes )
```

Obtain hash indice given the (dense) input vector, using OpenCL.

Hash indice refer to the corresponding "row number" in a hash table, in the form of unsigned integer. This function will only be valid when an CPU implementation exists for that type of hashing. The outputs indexing is defined as hashIndicesOutputIdx(numHashFamilies, numProbes, numInputs, inputIdx, probeIdx, tb) (unsigned)(numInputs ∗ numProbes ∗ tb + inputIdx ∗ numProbes + probeIdx).

Parameters

hashIndices_obj	Hash indice for the batch of input vectors.

Parameters

identity_obj	Hash indice's corresponding identifications (sequential number starting 0) for this batch of inputs.
input_obj	Input vectors concatenated.
numInputEntries	Number of input vectors.
numProbes	Number of probes per input.

3.1.2.5 getHash() [4/4]

```
void LSH::getHash (
             unsigned int * hashIndices,
             unsigned int * identity,
             float * input,
             int numInputEntries,
             int numProbes )
```

Obtain hash indice given the (dense) input vector, using OpenCL.

Hash indice refer to the corresponding "row number" in a hash table, in the form of unsigned integer. This function will only be valid when an OpenCL implementation exists for that type of hashing, and when OpenCL is initialized (clLSH) for that hashing. The outputs indexing is defined as hashIndicesOutputIdx(numHashFamilies, numProbes, numInputs, inputIdx, probeIdx, tb) (unsigned)(numInputs ∗ numProbes ∗ tb + inputIdx ∗ numProbes + probeIdx).

Parameters

hashIndices	Hash indice for the batch of input vectors.
identity	Hash indice's corresponding identifications (sequential number starting 0) for this batch of inputs.
input	Input vectors concatenated.
numInputEntries	Number of input vectors.
numProbes	Number of probes per input.

The documentation for this class was generated from the following files:

- LSH.h
- LSH.cpp
- LSH_helpers.cpp
- LSH_init.cpp

3.2 LSHReservoirSampler Class Reference

```
#include <LSHReservoirSampler.h>
```

Public Member Functions

- void **restart** (LSH ∗hashFamIn, unsigned int numHashPerFamily, unsigned int numHashFamilies, unsigned int reservoirSize, unsigned int dimension, unsigned int numSecHash, unsigned int maxSamples, unsigned int queryProbes, unsigned int hashingProbes, float tableAllocFraction)
- LSHReservoirSampler (LSH ∗hashFam, unsigned int numHashPerFamily, unsigned int numHashFamilies, unsigned int reservoirSize, unsigned int dimension, unsigned int numSecHash, unsigned int maxSamples, unsigned int queryProbes, unsigned int hashingProbes, float tableAllocFraction)
- void add (int numInputEntries, int ∗dataIdx, float ∗dataVal, int ∗dataMarker)
- void ann (int numQueryEntries, int ∗dataIdx, float ∗dataVal, int ∗dataMarker, unsigned int ∗outputs, int k)
- void add (int numInputEntries, float ∗input)
- void ann (int numQueryEntries, float ∗queries, unsigned int ∗outputs, int k)
- void showParams ()
- void checkTableMemLoad ()
- ∼LSHReservoirSampler ()

Public Attributes

- cl_platform_id ∗ **platforms**
- cl_device_id ∗ **devices_gpu**
- cl_context **context_gpu**
- cl_program **program_gpu**
- cl_command_queue **command_queue_gpu**
- cl_device_id ∗ **devices_cpu**
- cl_context **context_cpu**
- cl_program **program_cpu**
- cl_command_queue **command_queue_cpu**

3.2.1 Detailed Description

LSHReservoirSampler Class.

Providing hashtable data-structure and k-select algorithm. An LSH class instantiation is pre-required.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 LSHReservoirSampler()

```
LSHReservoirSampler::LSHReservoirSampler (
            LSH * hashFam,
            unsigned int numHashPerFamily,
            unsigned int numHashFamilies,
            unsigned int reservoirSize,
            unsigned int dimension,
            unsigned int numSecHash,
            unsigned int maxSamples,
            unsigned int queryProbes,
            unsigned int hashingProbes,
            float tableAllocFraction )
```

Constructor.

Creates an instance of LSHReservoirSampler.

Parameters

hashFam	An LSH class, a family of hash functions.
numHashPerFamily	Number of hashes (bits) per hash table, have to be the same as that of the hashFam.
numHashFamilies	Number of hash families (tables), have to be the same as that of the hashFam.
reservoirSize	Size of each hash rows (reservoir).
dimension	For dense vectors, this is the dimensionality of each vector. For sparse format data, this number is not used. (TBD)
numSecHash	The number of secondary hash bits. A secondary (universal) hashing is used to shrink the original range of the LSH for better table occupancy. Only a number <= numHashPerFamily makes sense.
maxSamples	The maximum number incoming data points to be hashed and added.
queryProbes	Number of probes per query per table.
hashingProbes	Number of probes per data point per table.
tableAllocFraction	Fraction of reservoirs to allocate for each table, will share with other table if overflows.

3.2.2.2 ∼LSHReservoirSampler()

```
LSHReservoirSampler::∼LSHReservoirSampler ( )
```

Destructor. Frees memory allocations and OpenCL environments.

3.2.3 Member Function Documentation

3.2.3.1 add() [1/2]

```
void LSHReservoirSampler::add (
             int numInputEntries,
             int * dataIdx,
             float * dataVal,
             int * dataMarker )
```

Adds input vectors (in sparse format) to the hash table.

Each vector is assigned ascending identification starting 0. For numInputEntries $>$ 1, simply concatenate data vectors.

Parameters

numInputEntries	Number of input vectors.
dataIdx	Non-zero indice of the sparse format.
dataVal	Non-zero values of the sparse format.
dataMarker	Marks the start index of each vector in dataIdx and dataVal. Have an additional marker at the end to mark the (end+1) index.

3.2.3.2 add() [2/2]

```
void LSHReservoirSampler::add (
            int numInputEntries,
            float * input )
```

Adds input vectors (in dense format) to the hash table.

Each vector is assigned ascending identification starting 0. For numInputEntries $>$ 1, simply concatenate data vectors.

Parameters

numInputEntries	Number of input vectors.
input	Concatenated data vectors (fixed dimension).

3.2.3.3 ann() [1/2]

```
void LSHReservoirSampler::ann (
            int numQueryEntries,
            int * dataIdx,
            float * dataVal,
            int * dataMarker,
            unsigned int * outputs,
            int k )
```

Query vectors (in sparse format) and return top k neighbors for each.

Near-neighbors for each query will be returned in descending similarity. For numQueryEntries $>$ 1, simply concatenate data vectors.

Parameters

numQueryEntries	Number of query vectors.
dataIdx	Non-zero indice of the sparse format.
dataVal	Non-zero values of the sparse format.
dataMarker	Marks the start index of each vector in dataIdx and dataVal. Have an additional marker at the end to mark the (end+1) index.
outputs	Near-neighbor identifications. The i_th neighbor of the q_th query is outputs[q $*$ k + i]
k	number of near-neighbors to query for each query vector.

3.2.3.4 ann() [2/2]

```
void LSHReservoirSampler::ann (
            int numQueryEntries,
            float * queries,
            unsigned int * outputs,
            int k )
```

Query vectors (in dense format) and return top k neighbors for each.

Near-neighbors for each query will be returned in descending similarity. For numQueryEntries $>$ 1, simply concatenate data vectors.

Parameters

numQueryEntries	Number of query vectors.
queries	Concatenated data vectors (fixed dimension).
outputs	Near-neighbor identifications. The i_th neighbor of the q_th query is outputs[q $*$ k + i]
k	number of near-neighbors to query for each query vector.

3.2.3.5 checkTableMemLoad()

```
void LSHReservoirSampler::checkTableMemLoad ( )
```

Check the memory load of the hash table.

3.2.3.6 showParams()

```
void LSHReservoirSampler::showParams ( )
```

Print current parameter settings to the console.

The documentation for this class was generated from the following files:

- LSHReservoirSampler.h
- LSHReservoirSampler.cpp
- LSHReservoirSampler_helpers.cpp
- LSHReservoirSampler_init.cpp
- LSHReservoirSampler_misc.cpp
- LSHReservoirSampler_routines.cpp
- LSHReservoirSampler_segsort.cpp