Data Mining: Learning from Large Data Sets - Fall Semester 2015

member1@student.ethz.ch
member2@student.ethz.ch
member3@student.ethz.ch

October 9, 2015

Approximate near-duplicate search using Locality Sensitive Hashing

In this project we applied Locality Sensitive Hashing (LSH) to select pair of near-duplicates from a set of videos. The input consists of a long list of lines, where every line contains the video ID and a set of shingles for that video.

1 Mapper

In order to implement LSH we need to compute a signature for each video. The calculation of the signature was made using a set of 100 hash functions. Every hash function is of the form:

$$h_i(r) = a_i r + b_i \tag{1}$$

where a_i and b_i are random numbers. Therefore before starting to read the input we computed these two random values for every hash function, i.e. a random matrix of size 100×2.

```
hash_functions = np.random.randint(MAX_INT, size=(HASH_FUNC_NUM, 2))
```

A signature for video v is then computed using the following algorithm:

for *i in 1:100* **do**
  signature[i] $\leftarrow \infty$
end
for *r in shingle* **do**
  for *j in 1:100* **do**
    hash $\leftarrow$ h(r) % MAX_INT signature[i] $\leftarrow$ min(signature[i], hash)
  end
end

MAX_INT is the maximum 16 bits integer. Each signature will be split into 10 segments; therefore every segment is a vector of length 10. Every segment will be then mapped into a bucket; so the number of

buckets is also 10. Hence before reading the input we need to initialize the hash function that maps a segment into a bucket. These hash functions have the following form:

$$h(\mathbf{s}) = \sum_{i=1}^{10} c_i s_i + b_i \tag{2}$$

The hash functions are initialized using the following command:

```
bucket_hash_functions = np.random.randint(MAX_INT, size=(BUCKET_SIZE + 1)).
```