Data Mining

Homework 2

Due: 12/4/2015, 23:59.

Instructions

You must hand in the homeworks electronically and before the due date and time.

Handing in: You must hand in the homeworks by the due date and time by an email to the instructor that will contain as attachment (not links!) a .zip or .tar.gz file with all your answers and subject
`[Data Mining class] Homework #`
where `#` is the homework number. After you submit, you will receive an acknowledgement email that your homework has been received and at what date and time. If you have not received an acknowledgement email within 1 day after you submit then contact the instructor.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

The solutions for the programming assignments must contain the source code, instructions to run it, and the output generated (to the screen or to files).

For information about collaboration, and about being late check the web page.

Most of the questions are not very hard but require time and thought. **You are advised to start as early as possible, to work in groups, and to ask the instructor in case of questions.**

Problem 1. Here we are asking to implement nearest-neighbor search for text documents. You have to implement shingling, minwise hashing, and locality-sensitive hashing. We split it into several parts:

1. Implement a class that, given a document, creates its set of character shingles of some length k. Then represent the document as the set of the hashes of the shingles, for some hash function.

2. Implement a class, that given a collection of sets of objects (e.g., strings, or numbers), creates a minwise hashing based signature for each set.

3. Implement a class that implements the locally sensitive hashing (LSH) technique, so that, given a collection of minwise hash signatures of a set of documents, it finds the all the documents pairs that are near each other.

To test the LSH algorithm, also implement a class that given the shingles of each of the documents, finds the nearest neighbors by comparing all the shingle sets with each other.

We will apply the algorithm to solve the problem that companies such as kijiji face when companies or individuals post many copies of the same announcement—usully they want to block announcements that are near duplicates. We will work on the announcements for job positions of Problem 1.

We want to find announcements that are near duplicates. We will say that two announcements are near duplicates if the Jaccard coefficient of their shingle sets is at least 80%. We will use shingles of length 10 characters. Find values for r and b (see Section 3.4 in the book) that can give us the desired behavior. To plot the graph that gives the probability as a function of the similarity for different values of r and b you can use, for example, Wolfram Alpha.

To apply the algorithm you have the following tasks:

1. Find the near-duplicates among all the announcements of Problem 1 using LSH.
2. Find the near-duplicates among all the announcements of Problem 1 by comparing them with each other.
3. Report the number of duplicates found in both cases, and the size of the intersection.
4. Report the time required to compute the near duplicates in either case.

You will need a way to create a family of hash functions. One way is to use a hash function and a code similar to the following.

```
# Implement a family of hash functions. It hashes strings and takes an
# integer to define the member of the family.
# Return a hash function parametrized by i
import hashlib
def hashFamily(i):
  resultSize = 8         # how many bytes we want back
  maxLen = 20            # how long can our i be (in decimal)
  salt = str(i).zfill(maxLen)[-maxLen:]
  def hashMember(x):
    return hashlib.sha1(x + salt).digest()[-resultSize:]
  return hashMember
```

Note that this code is an overkill because we use a cryptographic hash function, which can be very slow, even though it is not needed to be as secure. However, for the necessities of the homework we will use it to avoid having to install some external hash library.

After you implement LSH apply it on the kijiji data that you have downloaded. If you have not downloaded the full description ad (the optional part of the first homework) you will need to do it now. Report all the sets of ads that where duplicate, and the Jaccard similarity of their representation as sets.

Problem 2. We are given a set of points $V \subset \mathbb{R}^d$, with $|V| = N$, and a clustering $\mathcal{C} = C_1, \ldots, C_K$ in K clusters, that is, a partition of the points into sets C_k such that $\cup_{k=1}^{K} C_k = V$ and $C_k \cap C_\ell = \emptyset$ for $k \neq \ell$. Recall that the k-means cost function for clustering $\mathcal{C}$ is the

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_2^2 ,$$

where μ_k is the average of the points in C_k.

1. Assume instead that we use the objective function

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_1 ,$$

that is, we try to minimize the ℓ_1 distance between the points and the cluster center. How should we modify the k-means algorithm?

2. Assume that the optimal solution for the k-means cost function has cost C. You are now asked to cluster the points, but under the constraint that the cluster representative (i.e., the point corresponding to μ_k) has to be one of the input points in V. Prove that there exists a solution with cost at most $4C$.