Data Mining

Homework 3

Due: 1/12/2024, 23:59

Instructions

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

The first homework has to be done by each **person individually**.

Handing in: You must hand in the homeworks by the due date and time by an email to Gianluca (`decarlo@diag.uniroma1.it`) that will contain as attachment **(not links to some file-uploading server!)** a .zip file with your answers. The filename of the attachment should be `DM_Homework_3__StudentID_StudentName_StudentLastname.zip`;
for example:
`DM_Homework_3__1235711_Robert_Anthony_De_Niro.zip`.
The email subject should be
`[Data Mining] Homework_3 StudentID StudentName StudentLastname`;
For example:
`[Data Mining] Homework_3 1235711 Robert Anthony De Niro`.
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after the deadline then contact Gianluca.

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

If you have any questions, feel free to first email Gianluca: `decarlo@diag.uniroma1.it`

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

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 work on the Amazon products description already downloaded for HW2.

We want to find products that are near duplicates. We will say that two products 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:

To apply the algorithm you have the following tasks:

1. Find the near-duplicates among all the Amazon products, using LSH.
2. Find the near-duplicates among all the Amazon products 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.

Hand in the code, properly commented, with the instructions to run it.

Problem 2. For this exercise you will perform some clustering and use some feature engineering techniques. The lecture by Pablo Duboue[1] gave us a lot of interesting insights on the topic of feature engineering. It is a procedure that is sometimes underestimated but that, as you saw, can be really game changing in machine learning applications; the correct way to preprocess and model the features can increase the accuracy of many ML models. For this exercise we will use the famous dataset California Housing Prices: `https://www.kaggle.com/camnugent/california-housing-prices`. It is a very simple dataset widely used for ML models benchmarking. Even though being a simple and small dataset, it presents different kinds of features, from longitude and latitude coordinates to continuous and categorical values. The non homogeneity of the features strongly suggests us to use some engineering techniques to better work with such data. What you have to do is the following:

1. Choose one clustering algorithm (k-means++, DBSCAN, etc.) to cluster the samples in the dataset **without** any feature engineering technique applied, just on raw data. You can use library implementations. Use the elbow method (`https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/`) to find the most suitable number of clusters. You are also free to use a different method to find the optimal number of clusters, if you want (like Silhouette, etc).

[1] `http://duboue.net`

2. Inspect and understand your data and choose the best feature engineering technique (or techniques) that you would apply. Do whatever you think may be necessary (normalization, scaling, one-hot encoding and more). After that, perform the clustering again using the same clustering method used with the raw data.

Do you see any differences in the clusters obtained? What about running times? Make a plot of the clusters in both cases and comment what you see. Did your feature engineering help in creating more distinct clusters? Did it help with the shape of the elbow curve (or with the different method chosen)? Write a very short report (max 3 pages) in which you describe the clustering algorithm employed and the feature engineering techniques used (and why you think they were the proper techniques to use) along with the plots, comments and any observation on the results obtained that you think to be important. Hand in the report along with the code and instructions to run it.