Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Fall 2012, C. Faloutsos
Homework 2, Due Date: Tue Nov 6th, at noon

Prepared by: Kai Ren

Reminders

- Deposit your work as a single tar-file, in Blackboard - see the instructions below, as well as the '**What to turn in**' instructions for each question.
- **Before you start: Download** all three of these files:
  1. the *handin* one, a tar-file that has place-holders for your answers, makefiles etc
  2. the *handout* one, that has the datasets for questions 1,2 and 3, as well as a useful plotting script.
  3. the large data file with the patent information, for question 4: *data4-tar-file-link*.
- Please follow the same structure as the above `handin` tar-file *handin*. That is, the tar-file you deposit, should contain
  - a "**report.pdf**" with all your drawings, plots and typed answers.
  - **two folders** named "q1" and "q2", with your code and a `makefile`, for questions 1 and 2, respectively
  - Please **delete** all the un-needed/derived files from your archive.

FAQ

Q: Can we do this homework in groups?
A: NO - all homework are to be done **INDIVIDUALLY**.
Q: How long will it take to solve?
A: Approximate estimates:

- Q1: 2-5 hours
- Q2: 2-4 hours
- Q3: 1-5 hours
- Q4: 2-4 hours

Q: Who is the contact TA?
A: Mr. Kai Ren, `kair@cs.cmu.edu`

1 Q1 – Dynamic time warping [25 pts]

Problem Description: The goal is to implement the so-called *dynamic time warping* (DTW) algorithm to measure the distance between two time sequences. It is similar to the string editing distance algorithm (see lecture slide *210_text1.pdf*, page 6) Intuitively, it measures the editing operations that are needed to transform the first time sequence into the second time sequence, allowing:

- substitutions (at the cost of the absolute value of the difference).
- stutterings (ie, we can repeat the last value of a sequence), at no additional cost

More formally, the definition of DTW is as follows:
Suppose there are two time sequences, the query sequence $Q = q_1, \ldots, q_n$ and the data sequence $C = c_1, \ldots, c_m$. As with the string editing distance, we compute the cost $f(i, j)$ of matching the length-i prefix of Q with the length-j prefix of C.

$$f(i,j) = |q_i - c_j| + \min \begin{cases} f(i-1, j-1) & // \text{ substitution} \\ f(i, j-1) & // \text{ stutter first seq. Q} \\ f(i-1, j) & // \text{ stutter second seq. C} \end{cases} \tag{1}$$

and the initial conditions are the obvious ones: For $i = 1$, we 'stutter' the first value q_1 of the first sequence (Q), enough times:

$$f(1, j) = \sum_{1 \le k \le j} |q_1 - c_k| \tag{2}$$

and, symmetrically, for $j = 1$, we 'stutter' the value of c_1, enough times:

$$f(i, 1) = \sum_{1 \le k \le 1} |q_i - c_1| \tag{3}$$

The distance between the two sequences is then:

$$D(Q, C) = f(n, m) \tag{4}$$

Optionally, for more details, see [1] or click *DOI-link*.

Dataset description You are given a small subset of the UCR insect dataset (see the *data123-tar-file-link* we mentioned earlier, `handout/q1/data`). It contains 10 files `00*.dat`, each with a time sequence. The format is one integer per line, with 'unix/linux' end-of-line termination; the first line is the duration, $m = n$=15,883 time-ticks for all of them.

Implement the DWT algorithm to compute the similarity between any two time sequences, and report (a) the most similar pair from the datasets and (b) their distance.

Implementation Instructions:

1. [**20 pt**] Write a program `timewarp.cc` (or *.py *.pl etc), that
   - takes two file-names of time-sequences as the input parameters and
   - prints the filenames, and their DTW distance.
2. [**5 pt**] Find out the most similar pair in the given UCR insect dataset sample, and their distance. In case of a tie, print all qualifying pairs (omit mirror pairs; sort file names in ascending lexicographic order).

What to turn in:

- **Code:** The source code of your program "`timewarp.cc`". (or `timewarp.py` etc.).

Details on what to turn in

- *Packaging instructions - overview:* See our template in `/handin/q1/timewarp.cc` in the given `handin` example, and replace things accordingly.
- *Packaging instructions - gory details:* Replace our place-holder `timewarp.cc` with your solution. Adjust the included `makefile` so that "`make`" generates the correct answers into the file "`hw2.q1.output.txt`". This text file should contain one line per qualifying pair, with three blank-separated entries:

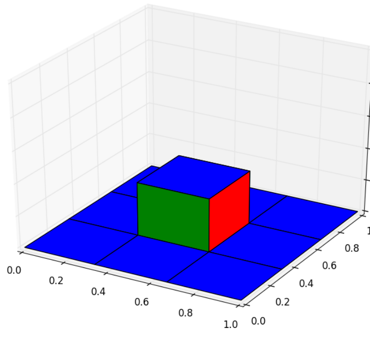
file1 file2 DWT_distance

2 Q2 – Fractals [25 pts]

Problem Description: We want to generate the *Koch surface* K_n of order n, (for any n), in three dimensions. Figure 1 shows the Koch surfaces K_1 and K_2, respectively.
In each iteration, each square surface follows the transformation shown in Figure 2: The transformation divides the square into a 3-by-3 grid of 9 square cells, and it replaces the middle cell with a cube.

Implementation Instructions:

1. [**10 pt**] Write a program that generates the Koch surface. It should read n, the order of the desired curve, as the only input from the command line, and generate the corner-points of the K_n Koch surface, in the format described below.
   Output format details: Assume that the coordinates of initial four points are the corners of the unit square at altitude zero $((0,0,0),(0,1,0),(1,1,0),(1,0,0))$. Your program should first output a number m in the first line, that is the number of square surfaces


(a) K_1

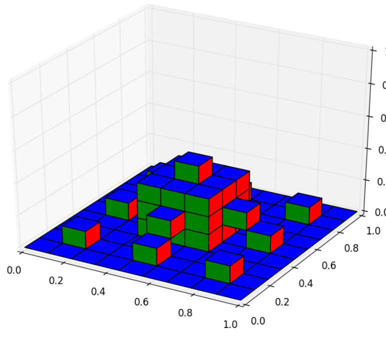
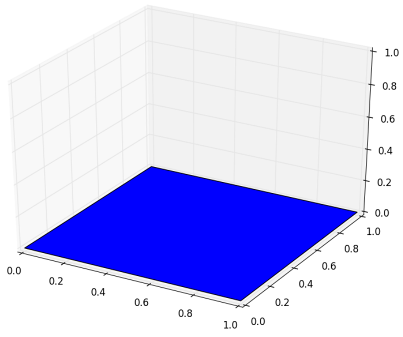

(b) K_2

Figure 1: Example Koch surface for K_1 and K_2


(a) Before transformation

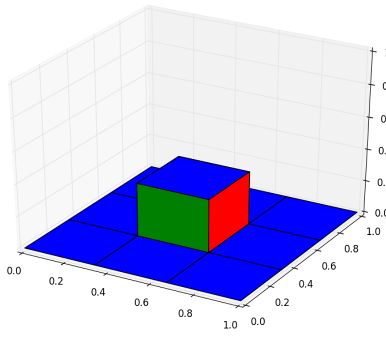

(b) After transformation

Figure 2: Transformation of Koch surface in each iteration

in K_n. It should then generate m lines, each containing float numbers separated by white space. These numbers are the coordinates of four points in each square surface of K_n (the first three numbers are the coordinates of the first point, and so on). For example, for K_0, your program should output the following lines:

```
1
0 0 0 0 1 0 1 1 0 1 0 0
```

2. **[5 pt]** Plot the next two iterations of the surface K_3 and K_4. For your convenience, the `handin` tar-file contains a python script `draw_surface.py`, that takes a file in the format described above, and plots the corresponding 3-D surface. [1]

3. **[10 pt]** Give the Hausdorff plot for the points of K_4, and report the corresponding Hausdorff fractal dimension. Use only the corner points of each surface. You may use

[1] You can also plot the surface using other tools such as Matlab. This python script requires the *matplotlib* library, which is available in many platforms, and it is already on the Andrew machines. For Ubuntu, do *sudo apt-get install python-matplotlib*.

the package `http://www.cs.cmu.edu/~christos/SRC/fdnq_h.zip`

What to turn in:

- **Code:** Hand in the source code named `fractal.cc` which takes n as the input parameter and generates all the surface coordinate points. Replace `/handin/q2/fractal.cc` in the given hand-in example, and adjust the `makefile` to generate the `png` figures of K_3 and K_4.
- **Answers:** In your `report.pdf`, give
  1. the two figures that are the images of the fractals K_3, K_4
  2. the Hausdorff plot for K_4
  3. a number: the Hausdorff fractal dimension of the K_4 surface.

3 Q3 – Separability [25 pts]

Problem Description: The goal is to analyze high dimensional datasets that cannot be easily visualized. Consider the following fictitious, but realistic setting: We have two datasets in a medical study, one from "healthy" ("H") participants, and one from "sick" ("S") ones. Each participant is modeled as a 4 dimensional point, e.g., body-height, body-weight, age, cholesterol-level. (The datasets are in the *data123-tar-file-link* handout archive mentioned above, in directory `q3`).

- Doctor Bob claims that the two groups seem *inseparable*, pointing out that all the scatterplots he tried, of all the 4-choose-2 and 4-choose-3 attributes, can not visually separate the two groups. See, e.g., Figure 3, where the "healthy" and "sick" points are on top of each other.
- Doctor Alice disagrees: *"They are separable under some method"*.

Who is right? Use the tools we covered in class, to settle the dispute.

Implementation Instructions:

1. [**10 pt**] Plot the correlation integral of dataset S, and of dataset H.
2. [**5 pt**] Based on the correlation integrals, list your conclusions about datasets H and S. Specifically, answer the following questions in your `report.pdf`: (a) Is dataset H uniformly distributed in 4D space? (b) Is dataset S uniformly distributed in 4D space?
3. [**5 pt**] Are H and S separable? (i.e., is Doctor Alice right?)
4. [**5 pt**] Please describe the tool(s) you use, give a brief justification for your answer, and give all necessary plots and figures that your justification is based on.

The last sub-question needs more details:

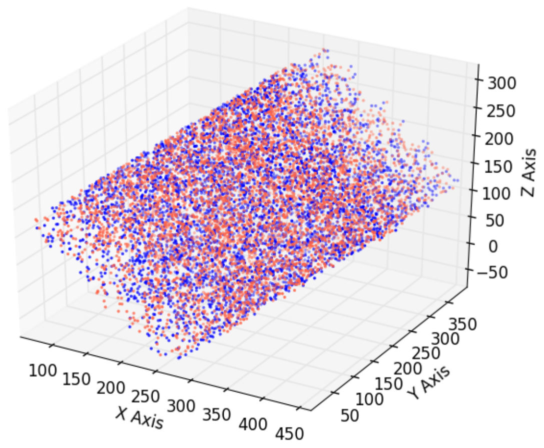


Figure 3: A 3D projection of the 4D patients dataset - red (sick) and blue (healthy) points seem inseparable.

- *Justification*: E.g., one (possibly incorrect) justification would be "*they are separable, because there is a plateau in the correlation integral of H, but not in the one of S*". Another (possibly incorrect) justification would be "*they are inseparable, because their correlation-integral plots are identical*".
- *Tools*: You may use any method we saw in class (but **no other methods**). For example,
  - the correlation integral plot (see lecture slides *160_fractals1.pdf* on page 45, with software at *link*),
  - *K-D* tree (software at *link*),
  - BOPS plot (also known as "triplots" - see lecture slides *160_fractals1.pdf* page 12, software at *link*),
  - R-trees (see our earlier *Homework 1* for algorithms and software)
  - z-ordering (foils at 120_SAMs2.pdf).

What to turn in:

- **Answers:** In your `report.pdf`, include
  1. Two figures, with the correlation integral plots of H and S.
  2. The text answers and justifications to the sub-questions.
  3. Any plot(s) or figure(s) that your justification is based on.

4 Q4 – Power Law in Graphs [25 pts]

Motivation In real graphs, we often have anomalies: botnets in computer networking traffic, link-farms for SEO (search engine optimization), account hijacking in ebay, facebook, etc. In most of these settings, the fraudsters induce abnormal connectivity structures. Here we try to mimic such a situation, and you have to spot the 'fraudsters'.

Problem Description: The goal is to see whether/how power laws can help us spot anomalies in large, real graphs.
Specifically, you are given the US patent dataset which contains the citations between patents. Many properties in the original dataset follow the power law, such as the in- and out-degree distributions. We injected a set of k fake nodes/patents, that all k point to many of the remaining $k-1$ ones, forming a near-perfect *clique*.
The goal is to spot the these fake patents, or, at least, to say as much as you can about them, eg., what is the value of k? which are the id's of the fake patents?

Implementation Instructions:

1. **[15 pt]** The original datasets is a *directed* graph with citation information, like patent "A" cites patent "B". Plot the distribution of the in-degree, the out-degree and the total (= undirected) degree. The latter is defined as the sum of in-degree and out-degree.
   In search of anomalies, plot
   - the rank-degree ('Zipf-like' plot), and
   - the degree-count (PDF)

   for each of the three cases: in-degree, out-degree, total-degree.
2. **[5 pt]** Based on previous plots, what is your guess for k, the count of the fake patents.
3. **[5 pt]** Justify briefly. For example, a (possibly wrong) justification would be, e.g., "*there is a plateau at degrees 20-through-40, which could be due to a clique of size 20*".
4. **[0 pt]** (Optional - very hard - gets zero points, but gets the admiration of the teaching staff). Give a list of patent ids, that you think are fake.

What to turn in:

- **Answers:** In your `report.pdf`, please give:
  1. the six plots for the degree distributions
  2. your estimate for the size k of the injected near-clique
  3. your justification why you think k patents are fake.
  4. (and optionally, the list with some/all of the ids of fake patents)

References

[1] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.