Carnegie Mellon University
Department of Computer Science
15-826 Multimedia Databases and Data Mining
C. Faloutsos, Spring 2016

Homework 3 - Solutions

Due: **Soft + hard copy, 3:00pm, on 03/02/2016**

VERY IMPORTANT:

1. On Blackboard, deposit a `tar-file` with your code, as described on page 2.
2. Deposit a **hard copy** of your answers, in class. As before:
   - **Separate** your answers, and
   - type your full info on each page (andrewId, course#, hw#, q# etc).

Reminders:

- *Plagiarism*: Homework is to be completed *individually*.
- *Typeset* all your answers.
- *Late homeworks*: email it
  - to all TAs
  - with the subject line exactly: `15-826 Homework Submission (HW 3)`
  - and the count of slip-days you are using.

For your information:

- Graded out of **100** points; **4** questions total
- Rough time estimate: ≈ *4 -16 hours*

Revision : 2016/03/22 23:42

Question	Points	Score
Correlation Integral	30	
Breaking the Law of Large Numbers	20	
String Editing Distance	25	
Text: Compression, brevity, and Elias codes	25	
Total:	100	

Preliminaries: Code-packaging info

These instructions are the same as in earlier homeworks, and repeated for your convenience.

Summary

Submit your code to blackboard, in a single `tar` file, called

andrewId`-hw3.tar.gz`

We will refer to that as *"your-tar-file"*.
As before, for your convenience, we provide a `template-tar-file package`, at

`http://www.cs.cmu.edu/~christos/courses/826.S16/HOMEWORKS/HW3/template-hw3.tar.gz`

It has 4 directories `/Q1, /Q2, /Q3, /Q4`, the first of which has the 'mystery' dataset of Q1.

To Do:

- `tar xvfz template-hw3.tar.gz`; `make`;
- Delivery:
  - In each directory `Q1-Q4`, replace the place-holder code `q*.py` with your solutions,
  - package the necessary files (`make package`)
  - and submit to blackboard.

Hints:

Please explore the `makefile`s we have created for your convenience. For example, from the top directory:

- `make hw3` will run the code for all 4 questions
- `make package` will try create the tar-file, for submission
- `make spotless` will/should clean up all the derived files
- `make sanity-check` for Q3, Q4 only, to verify you have the correct format

FAQs

Q: May I change the `makefile`s?
A: 'yes', for Q1/Q2; 'yes', for Q3/Q4, as long as `make autograde` works
Q: What language should we use?
A: **Python** - it is one of the most readable scripting languages; it leads to short solutions for this homework, and it is popular in the industry (eg., click here, or here).

Question 1: Correlation Integral . . . . . [30 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Grading info: Graded by: Varshaa Naganathan

Problem Description: The purpose is to show the power of the correlation integral, to help us find patterns in a dataset with d-dimensional points. Suppose a physicist colleague has T= 50 snapshots of sub-atomic particles, as 3-d coordinates plus timestamp. Or, a medical doctor has T snapshots of cancer-cells, progressing over time inside the body of a lab rat. Or, an astrophycist colleague runs a simulation of the universe, and, again, she has several galaxy coordinates, changing over time.

Your goal is to help your colleague understand how the objects (particles/cells/galaxies), are moving, if at all. Do they form clusters? If yes, how many? Are the clusters moving, merging, spliting?

Setup: The `template-tar-file package` contains a *mystery* dataset: Each row has blank-separated quadruplets

$$x \; y \; z \; t$$

The timestamp t is an integer ($t = 1 \ldots 50$), and the rest are floats. Notice that:

- We don't know the *identity* of each object - thus, the (1,1,1) object at time $t = 1$, may or may not, be the object (0.9, 1.2, 1.3) at time $t = 2$.
- Some objects may be missing (i.e., not recorded, or disappeared) in the middle of the experiment; some other objects, may appear or re-appear, at a later time.

Implementation Details: Implement an algorithm for calculating the correlation integral of a cloud of points and use it on time-slices of the *mystery* dataset. We will compare your code against the approximate, but faster, box-counting method is at `http://www.cs.cmu.edu/~christos/SRC/fdnq_h.zip` We shall refer to that package as *"box-counting package"*.

(a) **[4 points]** Implement the naive $O(N^2)$ algorithm for computing the correlation integral. Include self-pairs and mirror pairs.

(b) Plot and submit the correlation integral for the *mystery* dataset at $t = 1$ using

i. **[2 points]** (a) your method and the $L1$(City-Block) distance and report the slope(s) and range(s). Superimpose the plot of the *"box-counting package"*.

Solution: Range: [0.00898, 3.46174], with outliers - [0.00898, 30000]
Grading info: -1 if range given with outliers, deduct only once, -1 for slope or range not mentioned, if range values were given in log scale marks have been given iff base was mentioned

Question 1 continues. . .

ii. **[2 points]** (b) your method and the $L2$(Euclidean) distance and and report the slope(s) and range(s). Superimpose the plot of the *"box-counting package"*.

Solution: Range: [0.00687, 2], with outliers - [0.00687, 17320]
Grading info: -1 for slope/range not mentioned

iii. **[2 points]** Also plot the correlation integral at $t = 50$ using your method and $L2$(Euclidean) distance and report the slope(s) and range(s). Superimpose the plot of the *"box-counting package"*.

Solution: Range: [0.343, 100], with outliers - [0.343, 17320]
Grading info: -1 for slope/range not mentioned

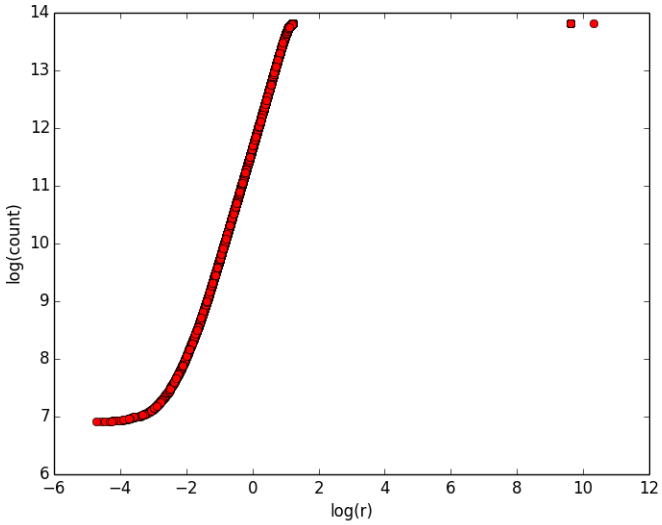

Figure 1: Correlation integral at t = 1 using L1 distance

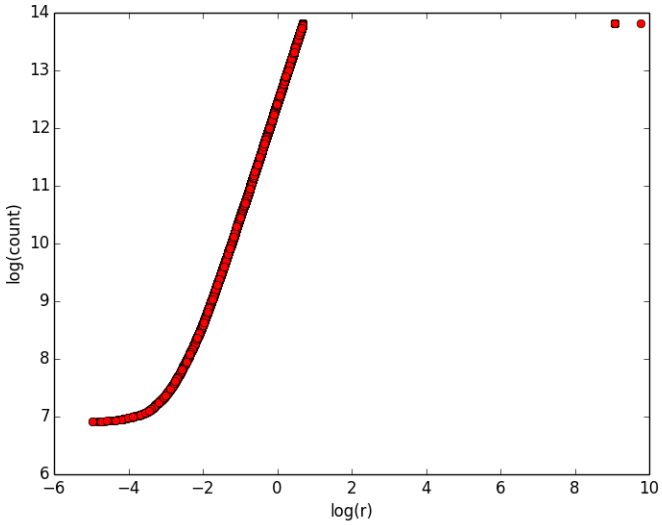

Figure 2: Correlation integral at t = 1 using L2 distance

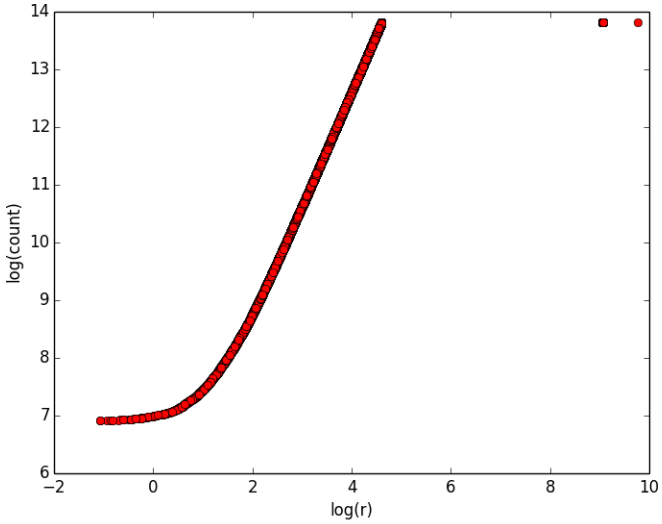

Figure 3: Correlation integral at t = 50 using L2 distance

(c) For the given dataset, answer the following questions:

i. **[2 points]** What is the fractal dimension of the given data at $t = 1$? If it is *not* self-similar, say "*not fractal*."

ii. **[2 points]** What is the fractal dimension at $t = 50$?

Solution: All time-ticks have fractal dimension about 2

Grading info: 2+2

iii. **[2 points]** How many clusters (corresponding to plateaus in the correlation integral) are present in the data at $t = 1$? (ignore isolated points, if any).
Grading info: -1 if answer is 0 clusters, without proper explanation that student considers the fact that all points belong to a single entity to mean 0 clusters

iv. **[2 points]** Does the number of clusters change with time? (that is, do plateaus appear, disappear, or change height/width, over time?)

Solution: All time-ticks have only 1 cluster. No change in number of clusters.

Grading info: 2+2+2

v. **[3 points]** What are the characteristic scales, if any, at $t = 1$? List them *all*, along with an explanation. We expect answers of the form, e.g., "$r_1 = 0.3$*: is the smallest distance between two points*".

Solution: Using L2 distance: $r_1 = 0.00687, r_2 = 2$
Grading info: -1 for one wrong value, if r_2 without outliers not mentioned, only one mark deducted for this and next question together

vi. **[3 points]** Repeat, for $t = 50$ - that is, give all characteristic scales, if any, at t=50, along with explanations.

Solution: Using L2 distance: $r_1 = 0.343, r_2 = 100$
Grading info: -1 for one wrong value

vii. **[2 points]** Submit a plot of the set diameter $d(t)$ (= largest distance) as it changes over time (t=1, ..., 50). Ignore isolated, remote points (if any).

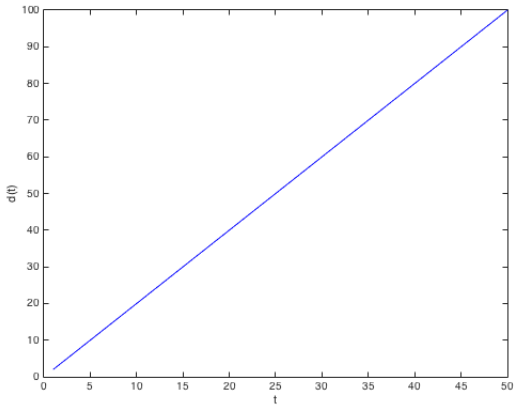

Figure 4: Plot of set diameter vs. time

viii. **[1 point]** Is the cloud of points moving? Choose one of the options below.

A. Marching: all or most of the objects/points move in unison, parallel to some line.

B. Imploding: all or most of the objects/points, move towards some center

C. Exploding (reverse)

D. Stop-and-go: the objects move in unison, stop in unison, then move again

E. Splitting: The objects are splitting into groups that move away from each other

F. Merging: the objects initially form several clusters, which merge into one cluster

G. Hovering: The objects stay more or less in their place (possibly, wth small oscillations)

H. None of the above - Explain briefly.

ix. **[3 points]** Give a brief explanation on how you arrived at your conclusion about the movement of the dataset

Solution: All time-ticks have fractal dimension about 2; the radius seems growing over time; the 2-d plots seem like a sphere

What to turn in:

- **Code:** in *"your-tar-file"*, your code `Q1/q1.py` along with the `Q1/makefile`
- **Answers:** On hard copy, please submit
  1. the code for Q1(a) see the solution tar-file solution tar-file
  2. the plots for Q1(b)
  3. the answers for Q1(c)

Question 2: Breaking the Law of Large Numbers .... [20 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Grading info: Graded by: Wan-shu Lai

Problem Description: The goal is to become familiar with the Pareto distribution, which is a Power Law, heavy-tailed distribution, and to observe that the law of large numbers only holds for random variables with finite variance. The law of large number states that, if we have n samples $x_1, \ldots x_n$ that all come from the same distribution, then the sample average $s(n)$

$$s(n) = 1/n \sum_{i=1}^{n} x_n$$

tends to the average of the distribution, as n grows.

Write the programs to do the following tasks.

(a) Generate N=1,000 Gaussian samples $(x_1, \ldots, x_N)$, with mean μ=100 and standard deviation σ=1. Delete negative ones, if any.

i. **[4 points]** Hand in your code (hard- and e-copy)

ii. **[2 points]** Plot the histogram for their PDF (in linear-linear scales - it should be a 'bell' curve)
Grading info: -1 Plot is not a histogram graph

iii. **[2 points]** Plot the partial average $s(n)$ versus n.

iv. **[2 points]** What do you observe? that is, does $s(n)$ grow/shrink/stabilize?

Solution: It is stabilizing, towards μ=100.

(b) Repeat the above task, with N=1,000 samples from the Pareto distribution $P(X > x) = 1/x \quad x \geq 1$

i. **[4 points]** Hand in your code (hard- and e-copy)

ii. **[2 points]** Plot their CCDF (in log-log scales - it should be a line), also plot a line with slope $= \quad -1$, for the reference.
Grading info: -1 the reference line does not plot properly
Grading info: -1 plot points smaller than 1
Grading info: -2 incorrect plot

iii. **[2 points]** Plot the partial average $s(n)$ versus n, as defined above.

Solution: See solution-tar-file - should be a rugged curve. Notice that you get different curve if you run it again
Grading info: -1 improper graph scale

iv. [**2 points**] What do you observe?

Solution: It is erratic - a few jumps, followed by smooth drops.
Grading info: -1 conclusion is slightly incorrect or missed the unstable trend
Grading info: -2 incorrect conclusion

What to turn in:

- **Code:** replace `Q2/q2.py` in *"your-tar-file"*
- **Answers:** Submit a hard copy for all your code, plots and answers.

Question 3: String Editing Distance . . . . . [25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Grading info: graded by: Varshaa Naganathan

Problem Description: The goal is to become familiar with the computation of string editing distance, and its dynamic-programming implementation.

Implementation Details: Write code using **Python** (a) to print out the string editing matrix, (b) to compute the string editing distance. See foils 32-36 (p. 8-9) in the class lecture slides, for an example of the string editing matrix: `http://www.cs.cmu.edu/~christos/courses/826.S16/FOILS-pdf/210_text1.pdf`

Use the following costs, that try to model the acoustic differences.

- $C_I = 1$ *# insertion cost*
- $C_D = 1$ *# deletion cost*
- $C_S = 0.1$ *# substitution: vowel by vowel*
- $C_S = 0.5$ *# substitution: consonant by consonant*
- $C_S = 0.9$ *# substitution: vowel by a consonant*
- $C_S = 0.9$ *# substitution: consonant by a vowel*

A vowel is one of {'a','e','i','o','u'}.

Output Format: In ./Q3, your code should give empty for `make sanity-check`

(a) **[5 points]** Turn in a hard-copy and e-copy of your algorithm

Solution: See solution tar-file.

(b) **[20 points]** For each of these word pairs give the string editing matrix and the string editing distance.

1. `abababab ddddddddd`
2. `receive recieve`
3. `blank bland`
4. `blank brandy`

Solution: See Tables below.

What to turn in:

- **Code:** Replace `Q3/q3.py`, and include it in *"your-tar-file"*. Make sure that your code passes: `make sanity-check`

Table 1: String Editing Matrix for abababab ddddddddd
String Editing Distance = 6.6

	ϕ	d	d	d	d	d	d	d	d	d
ϕ	0	1	2	3	4	5	6	7	8	9
a	1	0.9	1.9	2.9	3.9	4.9	5.9	6.9	7.9	8.9
b	2	1.5	1.4	2.4	3.4	4.4	5.4	6.4	7.4	8.4
a	3	2.5	2.4	2.3	3.3	4.3	5.3	6.3	7.3	8.3
b	4	3.5	3.0	2.9	2.8	3.8	4.8	5.8	6.8	7.8
a	5	4.5	4.0	3.9	3.8	3.69	4.69	5.69	6.69	7.69
b	6	5.5	5	4.5	4.4	4.3	4.19	5.19	6.19	7.19
a	7	6.5	6	5.5	5.4	5.3	5.19	5.1	6.1	7.1
b	8	7.5	7	6.5	6	5.9	5.8	5.69	5.6	6.6

Table 2: String Editing Matrix for receive recieve
String Editing Distance = 0.2

	ϕ	r	e	c	i	e	v	e
ϕ	0	1	2	3	4	5	6	7
r	1	0	1	2	3	4	5	6
e	2	1	0	1	2	3	4	5
c	3	2	1	0	1	2	3	4
e	4	3	2	1	0.1	1	2	3
i	5	4	3	2	1	0.2	1.2	2.1
v	6	5	4	3	2	1.2	0.2	1.2
e	7	6	5	4	3	2	1.2	0.2

- **Answers:** On hard copy, please submit
  1. your python code, and,
  2. your output for the above word-pairs (distance, and matrix)

For the code, see the solution tarfile at solution tar-file

Table 3: String Editing Matrix for blank bland
String Editing Distance = 0.5

	ϕ	b	l	a	n	d
ϕ	0	1	2	3	4	5
b	1	0	1	2	3	4
l	2	1	0	1	2	3
a	3	2	1	0	1	2
n	4	3	2	1	0	1
k	5	4	3	2	1	0.5

Table 4: String Editing Matrix for blank brandy
String Editing Distance = 2

	ϕ	b	r	a	n	d	y
ϕ	0	1	2	3	4	5	6
b	1	0	1	2	3	4	5
l	2	1	0.5	1.5	2.5	3.5	4.5
a	3	2	1.5	0.5	1.5	2.5	3.5
n	4	3	2.5	1.5	0.5	1.5	2.5
k	5	4	3.5	2.5	1.5	1	2

Question 4: Text: Compression, brevity, and Elias codes[25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Grading info: Zhoucheng Li

Problem Description: The goal is to become familiar with compression through Elias-gamma coding. Write code to derive the Elias-gamma coding of an integer, as well as code for the inverse.

Output Format:

- `eliasg.py` 3 *# should return '011'*
- `eliasg.py` 10 *# should return '0001010'*
- `eliasg.py` 17 *# should return '000010001'*

Similarly, the command-line syntax of the inverse should give:

- `ieliasg.py` 0001010011 *# should return '10 3'*
- `ieliasg.py` 000010001 *# should return '17'*
- `ieliasg.py` 10 *# should return:* `Illegal input.` (notice the final period.)

Implementation Details: Write the programs to do the encoding and decoding, `eliasg.py` and `ieliasg.py` respectively.

(a) **[5 points]** `eliasg.py` should return the encoding of the integer.

Solution: see the code in the solution tar-file.
Grading info: -2.5 points if program has bugs

(b) **[5 points]** `ieliasg.py` should give the decoding of the input codes.
Grading info: -2.5 points if program has bugs

(c) **[10 points]** Use your `eliasg.py` program to encode the following integers:
i). 15 $\rightarrow$ 0001111
ii). 32 $\rightarrow$ 00000100000
iii). 85 $\rightarrow$ 0000001010101
vi). 248 $\rightarrow$ 000000011111000
v). 4551 $\rightarrow$ 000000000000010001110001111
Grading info: 2 points for each answer

(d) **[5 points]** Use your `ieliasg.py` program to decode the following coding:
i). 000000000000100001000101000000100010001110000000100100001 $\rightarrow$ "4234 34 7 289"
ii). 000001001 *rightarrow* "Illegal input."
Grading info: 2.5 points for each answer

What to turn in:

- **Code:** Replace `./Q4/eliasg.py` and `./Q4/ieliasg.py` in *"your-tar-file"*. Remember, we will use `make autograde`, i.e, test, with `make sanity-check`.
- **Answers:** Submit a hard copy for your code, and for all your answers.