Data Mining

Homework 1

Due: 23/10/2017, 23:59.

Instructions—Read carefully!

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 Mara (`sorella@dis.uniroma1.it`) 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. In the text you must also mention the your name and student id (matricola). 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 2 days after you submit then contact Mara.

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 (but the writeup should be yours, as per the collaboration policy), and to ask the instructor in case of questions.**

Problem 1. A family has two kids, each being a boy or a girl with probability 1/2 and born in a random day of the week.

1. Define a sample space sufficient to answer the third question and define the probabilities to the points in the sample space.
2. If we know that one kid is a girl, what is the probablity that the other kid is a girl?
3. If we know that one kid is a girl born on Sunday, what is the probablity that the other kid is a girl?

Problem 2. You are in an airplane that falls in the jungle and you manage to survive. In the jungle there are are two tribes, the *Randomukee* and the *Bugiardukee*. The Randomukee are twice as many as the Bugiardukee. Each time you ask a question to a Randomukee he will say the truth with probability 3/4, whereas, each time you ask a question to a Bugiardukee he will lie. As you try to find your way out of the jungle, you find a random person from the two tribes. You ask him the question "To get out of the jungle, I have to go left or right?"

1. Define an appropriate sample space that can be used to answer the questions that follow, and define the probabilities for the points in the sample space.
2. Assume that the person gave you the answer "right." What is the probability that the answer is correct?

3. You ask the same person again, and he gives you the same answer. Show that the probability that the answer is correct is 1/2.

4. You ask a third time and you get again the same answer. What is now the probability that the answer is correct?

5. Finally, you ask a fourth time and you get again the answer "right." Show that the probability that the answer is correct is 27/70.

6. Assume that the first three times the answer was "right" but that the fourth one it was "left." Show that the probability that the correct answer is "right" is 9/10.

Hint: You need to define several events to answer the questions.

Problem 3. The Erdős-Rényi $G_{n,p}$ random-graph model, is a mathematical model for creating random graphs. Fix a positive integer n and a value $p \in [0,1]$. Then a graph created according to the $G_{n,p}$ model, has n nodes, and each pair of distinct nodes is connected with an edge with probability p, all pairs being independent from each other.

1. Define an appropriate probability space Ω to describe the $G_{n,p}$ model. What is its size?

2. What is the probability of each element of Ω?

3. What is the probability that a graph created according to $G_{n,p}$ contains exactly only one triangle (three nodes connected with each other and no other edges exist)?

4. What is the probability that all the n nodes are connected in a line (with no other edges present)?

5. Assume that we create a graph according to $G_{n,p}$, what is its expected number of edges?

6. Consider a random graph $G = (V,E)$ with $|V| = n$ nodes created according to the $G_{n,p}$ model. Let us define a *3-star* to be a subgraph $V' = \{v_0,v_1,v_2,v_3\} \subseteq V$ of V such that all the edges (v_0,v_i) for $i = 1,2,3$ exist in G, and none other of the edges (v_1,v_2), (v_1,v_3), and (v_2,v_3) exist in the graph (but other edges may exist, including edges between nodes in V' and other nodes in V). What is the expected number of 3-stars in G?

7. Define similarly a k-star. What is the expected number of k-stars?

Problem 4. In this question we will start getting our hands a bit dirty. We will practice a little bit with UNIX and then we will start playing with Python. At
`http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html`
you can find a dataset from Last.fm, containing songs that users have listened to. The initial file that we will work on is the
`userid-timestamp-artid-artname-traid-traname.tsv`,
which contains among other information, what song each user has listened to. Our goal is to find out what combinations of songs are the most common. But we want to restrict ourselves to songs that users listen to frequently.

- In the first part we will use UNIX commands to simplify and make the file shorter. Quite often you can analyze your data just by using simple UNIX tools and after one knows them he can use them for several simple tasks. Some usefull commands are the `grep`, `sort`, `uniq`, `cut`, `sed`, `awk`, `join`, `head`, `tail`, `wget`, `curl`. You can find more information using the `man` command or by checking the web. Shell scripting can help you even more.

  The file `userid-timestamp-artid-artname-traid-traname.tsv` contains several fields, some are empty, whereas some songs appear multiple times for each user. Our goal is to process it and create a file with the following format:

  $$\texttt{userid}_{\texttt{i}}\texttt{<TAB>track-id}_{\texttt{i1}}\ \ \texttt{track-id}_{\texttt{i2}}\ \ \ldots$$

  where $\texttt{userid}_{\texttt{i}}$ is the ith user and $\texttt{track-id}_{\texttt{ij}}$ is the jth track id that will contain only the songs that each user has listened at least 20 times. So every line must contain all the tracks that a user has listened to at least 20 times, and each track-id of those must appear just once. Note that in the initial file, some track ids are missing; ignore these songs.

  After downloading the file, use some of the above commands to convert the initial file to the format requested. You will need to use a bunch of the commands above. (**Hint:** You can do it with a single—but long, you'll need `awk`—command line, by chaining commands through pipes!)

- For the second part you need to write a Python program that finds what are pairs of songs that a lot of users like. From Part 1 we know for each user what tracks he has listened for at least 20 times. Write a Python program that outputs the top-10 pairs of track ids, that is, the track ids that correspond to the pairs of songs that have been listened at least 20 times by the most number of users.

Problem 5. We continue getting familiar with Python by starting playing with some APIs. There are several libraries that we will use often, and eventually we should learn them: `requests` is used to download web pages, `beautiful soup` is used to parse them, if they are sufficiently small `lxml` or regular expressions (package `re`) are used for larger pages, and so on.

Here instead we will play a bit with Twitter and Google maps. Our goal is to download a stream of tweets in Rome as they are created and create a web page that displays their location on Google maps. For that we need to access two APIs, the Twitter streaming API and the Google maps API.

1. First we will obtain the stream of tweets as they are generated. There are various ways to access the Twitter API. The most direct is to use the package `python-twitter`, which gives direct access to the API. However, there are several high-level packages, which hide several of the details. One of the easiest ones is the `twython` package. You are encouraged to use that one, but feel free to use whichever you prefer.

   To access the Twitter streaming API, you should register as a user, create a Twitter application, and generate four keys, which allow you to authenticate from your application. After you do that, you can use the `twython` library to set a query and listen for tweets that are being created in Rome (for that you need to define a bounding box to filter tweets and keep only tweets that are created in Rome). Thus the goal of this part is to grab the tweets in Rome as they are generated. In particular, we are interested in the location of the tweet, the screen name of the user, and the text of the tweet.

To learn more, you need to consult the documentation for `twython` and that of the Twitter API.

2. For the second part you are asked to plot the point locations on Google maps. A library for that is `pygmaps`, which although is far from begin complete, suffices for our purpose. Use this (or some other, if you prefer) library to save the points in an html file as points on a Google map, as they are being collected from Twitter. As title for each point use the

   `@screen_name: tweet` .

   The title is the text displayed when the mouse goes over the point. Note that `pygmaps` doesn't have implemented the title functionality, so you need to do a small modification to the package.