Data Mining

Homework 1

Due: 23/3/2014, 23:59.

You must hand in the homeworks electronically and before the due date and time. Check the web page for instructions about collaboration, about being late, and about handing in the homework.

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. We shuffle a standard deck of cards, obtaining a permutation that is uniform over all 52! possible permutations.

1. Define a proper probability space Ω for the above random process. What is the probability of each element in Ω?

2. Find the probability of the following events:

   (a) The first two cards include at least one ace.

   (b) The first five cards include at least one ace.

   (c) The first two cards are a pair of the same rank (they are the same number or both are J, or both are Q, etc.)

   (d) The first five cards are all diamonds.

   (e) The first five cards form a full house (three of one rank and two of another rank).

3. (Optional) Develop some small programs in Python to perform simulations to check your answers.

Problem 2. Prove that if events E and F are independent, then also their complements $\bar{E}$ and $\bar{F}$ are independent.

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 nodes is connected with an edge with probability p, all pairs being independent from each other. model.

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 the class we saw a question about people getting random jackets. Here we will see a variation. Assume that n people enter a restaurant and they leave their umbrellas at the entrance. In their way out, people leave one by one. By mistake, the first person, in his way out, did not get his own umbrella but got another random one and left. From that point on, each person searches for his own umbrella. If he finds it he takes it and leaves. Otherwise, he takes a random one from the ones left (and leaves). The problem is to compute the probability that the last person gets his own umbrela. The answer is probably not obvious. We will find it in two ways.

1. First we will do simulations. Write a python program that tries different values of n and computes the probability. To compute the probability you will need, for each value of n, to perform several trials and see how many times you succeed. From the results of your simulations make a conjecture for the correct answer.

2. Try to prove your conjecture. Probably the simulations in the first part will help you. (There are at least two ways to solve this problem; one is more straightforward but requires a few calculations, instead the second needs a good argument.)

Problem 5. 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:

$$\texttt{userid}_i \texttt{<TAB>track-id}_{i1} \quad \texttt{track-id}_{i2} \quad ...$$

  where $\texttt{userid}_i$ is the ith user and $\texttt{track-id}_{ij}$ 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 6. 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

   $$\texttt{@screen\_name: \ tweet} \ .$$

   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.