Stochastica I

Exercises in Probability and applications

Fall term 2020

Elementary Probability

In this chapter we discuss some problems in probability which are more than two hundred years old, but still provide an easy introduction to the subject. There are various strategies which can be adopted: if one is skilful in classical combinatorics he or she can try to find an exact solution; if this is not the case, and since we are in a computational physics blog, there is the almost universal strategy which employs the art of Monte Carlo simulations. Instead of repeating here the theoretical basis of probability theory, we refer to the lecture notes (in Italian) which can be found on the web site *www.eoinfnpr.it* . In a foreseeable future the lectures will be available in English.

1. Playing poker. For a poker player it is obviously first of all very important to have a clear notion of the probability of the various combinations, *three of a kind*, *full house*, *flush*, *four of a kind*, *straight*, *etc.* Starting from the a priori assumption that *all possible permutation of the deck of cards are equally probable* it's a simple matter of combinatorics to compute various probabilities. For instance: what is the probability of starting with *four of a kind* in the first hand? The answer is simple: let's play with a 32 cards deck, *7-8-9-…-K-A*. There are $\binom{32}{5}$ possible combinations for your hand. Among these, the four equal cards can be chosen in 8 ways and to complete the hand there is the choice among the remaining 28 possibilities. Hence the probability is simply

$$\frac{28 \times 8 \times 120}{32 \times 31 \times 30 \times 29 \times 28} = \frac{1}{31 \times 29} = \frac{1}{(30+1)(30-1)} = 1/899$$

or slightly less than once in a thousand cases.

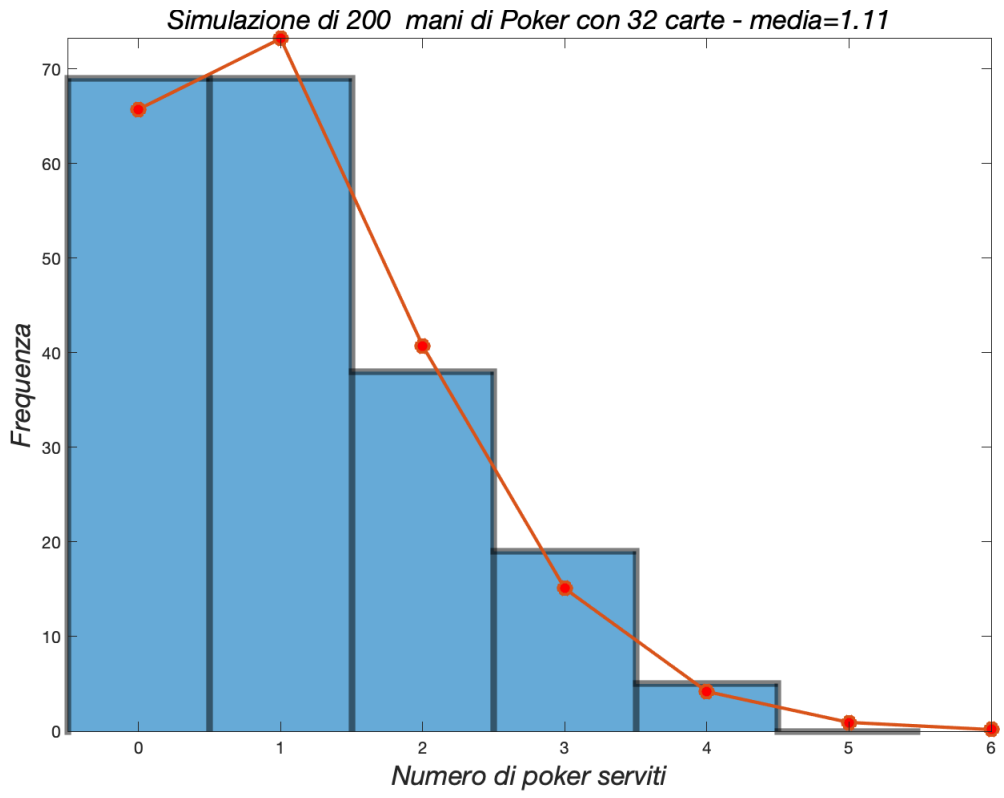
1.1 Now: how can we simulate a card distribution using a MonteCarlo technique? In this specific case we have the exact result, but in other situations it may be not so simple to compute the combinatorics. The easiest way to organise the simulation is using `matlab`. First we have to choose the data format to represent the deck of cards. Ignoring the different suits, we can simply define

```
C = 1:nC          % nC=32 in our previous calculation
D = [C,C,C,C];
```

then we shuffle the deck using *randperm* which is built in and chooses a permutation at random:

```
G = D(randperm(4*nC));
myHand = sort(G(1:5));
```

at this point we have our hand with all cards in increasing order; then if the first card is equal to the fourth or the second is equal to the last one we a have four equal cards, poker!
We set up a loop by repeating the operations a certain number of times, say 1000 (just intuitively we have to give a chance of at least one result in the average); hence the procedure is repeated a certain number of times, i.e. experiments, in order to estimate the statistics: it's important to have a precise idea of fluctuations; no result based on a simulation should be presented without a serious estimate of the statistical error!
The code *poker.m* can be found in the distribution *stochastica.tar* and you can make experiments; it takes less than a second to



run the code with the default values *nC=8, N=1000, Nsamples=200;* typically the result is like $P = (1.12 \pm 0.08)/1000$ consistent with 1000/899 ~ 1.1123. The histogram presenting the statistics of the *Nsamples* experiments shows that it is the Poisson distribution involved!

1.2 Having reached this stage, you can try for yourselves to extend the simulation to cover other possibilities: to do this, the first thing to do is to improve the data structure to include the suit:

♡, ♢, ♣, ♠

The C array can be modified in this way:

```
C = 1:nC; S1=ones(1,nC); G=[C,C,C,C];
 S =[S1,2*S1,3*S1,4*S1];
 Cards = [G;S];
```

A single column of Cards will look like $\begin{pmatrix} 8 \\ 3 \end{pmatrix}$, which is interpreted as 8 of clubs, not as a binomial! Then you should organise the simulation according to the outcome you want to examine. But we suggest more intriguing problems: if you happen to have *four in a hand (f.i.a.h.)* with a value *x*, what is the probability to win against other players who may have a *f.i.a.h.* themselves with a higher value, or even a *straight flush* (royal stairs or five cards of the same suit and all differing by one). This requires analysing all hands (4 or 5 players) for an outcome higher than yours…that's a lot of work, but it can be done.

To continue: other exercises to come: all sorts of dice games, learning your best bets. This knowledge goes back to Galileo and Pascal (!). Also quantum dice are interesting, i.e. dice obeying Bose or Fermi

statistics. Where we cannot do without computers and Monte Carlo simulations is the statistical physics of lattice models, except for a few simple cases. Here the basic fact is that to compute mean values with respect to the Gibbs measure $Z(\beta)^{-1}\exp\{-\beta E\}$ one should evaluate definite integrals on a huge number of integration variables, outside what is normally feasible. In this field an important amount of knowledge both algorithmic and theoretical has accumulated in the years 1950 — today, essentially starting from the pioneering work of Von Neumann, Kac and collaborators (like Metropolis whose name is associated to a general strategy for extracting lattice configurations in equilibrium with the Gibbs distribution) to our days with parallel computing and all sort of clever acceleration tricks. Here in the following months you may find an account of the various programming techniques and algorithms.