Getting rich and guessing lottery numbers

Rodion "rodde" Efremov

May 4, 2015

Abstract

This document describes a topic proposition for Data Mining Project -course.

1 Introduction

A lottery game is defined by a tuple (Σ, k), where Σ is the set of all possible symbols, and k is the amount of symbols a player may choose for each row. Above we require that $0 < k < |\Sigma|$. If Σ is the set of integers $\{1, 2, \ldots, N\}$, we denote it by slightly different notation: (N, k).

It is important to note, that in this project we consider only winning lottery rows as the actual gaming data is not available to persons and institutions outside Veikkaus.

2 The data set

The data set we aim to use in this project is the set of actual lottery rows since 9 January 1988 in Finnish (Veikkaus) Lottery (total of 1416 rows). Each model object for a row stores k distinct symbols from Σ and a time stamp denoting milliseconds since Unix epoch, which is 00:00:00 UTC, Thursday 1 January 1970.

An important point is that since the dawn of Veikkaus Lottery, both Σ and k changed from time to time. In this project, we will confine ourselves to $N = 39$, $k = 7$, which was the case at least since 1988.

3 Formulating the research question

Suppose we are given the lottery row database D, which is a multiset of k-sequences over Σ and each row mentions a symbol $\sigma \in \Sigma$ at most once. The Algorithm 1 will generate a **frequency graph**, which is an undirected, weighted graph with all edge weights within range $(0, 1]$.

Algorithm 1: BUILD-FREQUENCY-GRAPH(D)

```
1  V = ∅
2  E = ∅
3  w = ∅
4  for r ∈ D do
5      for each number pair u, v in r do
6          V ← V ∪ {u, v}
7          if {u, v} ∉ E then
8              E ← E ∪ {{u, v}}
9              w(u, v) ← 1
10         else
11             w(u, v) ← w(u, v) + 1
12 for each edge e ∈ E do
13     w(u, v) ← w(u, v) · |D|^{-1}
14 return (V, E, w)
```

Now, if we extract a connected tree T from a frequency graph and T has exactly $k - 1$ edges, T represents a lottery row, and its **probability** is the product of weights of its constituent edges. Sometimes we will use terms "tree" and "row" interchangeably.

Hypothesis 1. *Each new winning lottery row attempts to improve a tree with relatively low probability.*

Sketch of possible proof. Otherwise the frequencies in the frequency graph would diverse drastically. □

Basically, we want to mine the frequency graph for least probable trees. This can be achieved by running a minimum spanning tree algorithm on the frequency graph and then choosing, say n, least probable connected trees

with all trees with exactly $k-1$ edges. There is a problem: minimum spanning tree algorithms work by summing edge weights, not multiplying. The workaround is to take logarithms ($w(u,v) \to \ln w(u,v)$) from each frequency in the original frequency graph, run, say, Kruskal's algorithm on it, and after obtaining the minimum spanning tree $\hat{T}$, change all edge weights $w(u,v)$ of $\hat{T}$ to $e^{w(u,v)}$. This allows us to compute minimum spanning trees, in which product is used instead of sums, since

$$w_1 w_2 = e^{\ln w_1} e^{\ln w_2} = e^{\ln w_1 + \ln w_2}.$$

One possible contribution might be proving optimality of the above approach.

Another straightforward algorithm is to generate all $(k-1)$-combinations of edges and choose the least probable. Since for $k = 7$ there may be (and is) as much edges as 741, this approach is not feasible for $k > 3$.

If we sort each lottery row, and sort all rows by lexicographic order, we obtain the list L. Now, the $\mathbf{rank}(R, A)$ of a lottery row R is simply its appearance index in A, if it were inserted in A such that A would stay lexicographically sorted. Since $N = 39$ and $k = 7$, $|L| = \binom{N}{k} = 15380937$, and so $\mathbf{rank}(s) \in [1, 15380937]$. The preliminary question might be how the rank of winning lottery rows evolve in time? Another question of interest is as follow: We begin accumulating winning lottery rows into set A, and for each new row, we compute its **relative** rank **within the list of all winning lottery rows** A. That is, relative rank is defined by

$$\mathbf{rank}_{rel}(R) = \frac{\mathbf{rank}(R, A)}{|A|},$$

where R is a lottery row, A is the set of all accumulated winning lottery rows up until R. Now, we want to find out the time evolution of relative ranks.

The third question to tackle is mining least probable rows, and asserting that they do not vary much. This, also, will provide some concrete results.

The fourth and the most awesomest research topic is to devise a lottery winning algorithm. If it ever leads to at least 4 in the row (the least victory category), it might indicate that our understanding of lottery as a process is making baby steps towards understanding of lottery as a process. Of course, if the process is entirely random (without hidden variables), none of the algorithms are any better than guessing.