1 High level description of the problem

Input Any $N \times M$ matrix with positive entries, which we shall denote A

Output Permutations σ, π so that the $N \times M$ matrix B defined by $B_{ij} = A_{\sigma(i),\pi(j)}$ appears pleasing to me.

2 Algorithm

First we'll make some notation, then we'll write down the algorithm.

2.1 "Partition loss"

We will make our pleasing permutations by iteratively forming larger and larger partitions of the indices. Let us take a moment to make some notation:

Definition 1. An **ordered set** $(a_1, a_2 \cdots a_m)$ is a list of objects with no duplicates. If A, B are ordered sets, we take the notation that $A \cup B$ is what happens if you stick B on the end of the list A.

Definition 2. Consider a collection of ordered sets, $G = (G_1 \cdots G_n)$. That is, each G_k is itself an ordered set. We say that this collection is a **partition of a set** S if the collection is disjoint and $\bigcup_{k=1}^n G_k = S$.

Now, let us consider a partition that applies over both *rows and columns simultaneously*. One can think of this as a kind of matching between groups of rows and groups of columns. In particular, let

- $R = (R_1 \cdots R_n)$ denote a partitition of $\{1 \cdots N\}$
- $C = (C_1 \cdots G_n)$ denote a partition of $\{1 \cdots M\}$

So each R_k denotes a set of rows and each C_k denotes a set of columns.

For any given matrix A, we shall say the "loss" of this matrix with respect to these partitions is defined as

$$L(A, R, C) = \sum \log \frac{\sum_{i \in R_i \text{ or } j \in C_k} A_{i,j}}{\sum_{i \in R_k \text{ and } j \in C_k} A_{i,j}}$$

If this loss is small, what does it mean? Let's look at the terms in plain english:

- $\sum_{i \in R_i \text{ or } j \in C_k} A_{i,j}$ – all the mass that lies in entries which are **either** in the rows indicates by either the set R_k or the columns indicated by the set C_k
- $\sum_{i \in R_i \text{ and } j \in C_k} A_{i,j}$ – all of the mass that lies in the entries that are in the submatrix formed by looking **only** at the rows R_k and the columns C_k

If the loss is small, it means that *most of the mass* of the matrix lies inside smaller submatrices. Our basic approach will be to find a series of partitionings, at coarser and coarser levels of granularity, so that at each level of granularity the loss is as small as possible.

2.2 Algorithm

1. Use the Hungarian algorithm to find a best-possible bipartite matching between the rows and columns. That is, a collection

   $$\{r_k, c_k\}_{k \in 1 \cdots N \wedge M}$$

   that maximizes $\sum_k A_{r_k, c_k}$. We will use this matching to seed our partitions. Let R, C be defined by

   - $R_k = (r_k)$
   - $C_k = (c_k)$

Notice that R may not be a partition of $\{1\cdots N\}$ and C may not be a partition of $\{1\cdots M\}$. This will give us two collections of disjoint sets.

2. Get rid of any zero groups. If $A_{r_k,c_k}=0$ then we don't really want group k. So let $S=\{k:\ A_{r_k,c_k}=0\}$ and $R\leftarrow\{R_k\}_{k\in S}$ and $R\leftarrow\{C_k\}_{k\in S}$.
3. Add in any rows or columns who are now left out of the matching. Rows and columns can be left out because of the "getting rid of zero groups" thing as well as the fact that if $N\neq M$ not everybody can get matched. We will add these rows and columns in such that the loss stays smallish.
   (a) For each r such that $r\notin\cup_k R$,
      i. For each k', let $\tilde{R}^{(k')}$ be defined by taking R and adding r to the kth partition:
      $$\tilde{R}_k^{(k')}=\begin{cases}R_k\cup(r) & \text{if } k=k'\\ R_k & \text{else}\end{cases}$$
      and
      $$L^{(k')}=L(A,\tilde{R},C)$$
      ii. Let $k^*\leftarrow\arg\min_k L^{(k')}$
      iii. Let $R\leftarrow\tilde{R}^{(k^*)}$
   (b) For each c such that $c\notin\cup_k C$
      i. For each k', let $\tilde{C}^{(k')}$ be defined by taking C and adding c to the kth partition
      $$\tilde{C}_k^{(k')}=\begin{cases}C_k\cup(c) & \text{if } k=k'\\ C_k & \text{else}\end{cases}$$
      and
      $$L^{(k')}=L(A,R,\tilde{C})$$
      ii. Let $k^*\leftarrow\arg\min_k L^{(k')}$
      iii. Let $C\leftarrow\tilde{C}^{(k^*)}$
4. Repeatedly merge partitions until everything is in one big partition. When we merge we keep the order within each partition, so at the end we will have an order that respects all the granularities of partitioning that we have considered. Here's how it goes:
   (a) For each k_1,k_2, let $\tilde{R}^{(k_1,k_2)},\tilde{C}^{(k_1,k_2)}$ be defined by taking the partitions R,C and merging the k_1 set with the k_2 set. That is,
   $$\begin{aligned}A&=R_{k_1}\cup R_{k_2}\\ B&=C_{k_1}\cup C_{k_2}\\ \tilde{R}^{(k_1,k_2)}&=\{R\}_{k\notin\{k_1,k_2\}}\cup A\\ \tilde{C}^{(k_1,k_2)}&=\{C\}_{k\notin\{k_1,k_2\}}\cup B\end{aligned}$$
   and $L^{(k_1,k_2)}=L(A,\tilde{R}^{(k_1,k_2)},\tilde{C}^{(k_1,k_2)})$.
   (b) Let $k_1^*,k_2^*\leftarrow\arg\min_{k_1,k_2}L^{(k_1,k_2)}$.
   (c) Let $C\leftarrow\tilde{C}^{(k_1,k_2)}$ and $R\leftarrow\tilde{R}^{(k_1,k_2)}$

   We just repeat that process forever. We're left with $R=(R_1)$ and $C=(C_1)$. The order within R_1 and C_1 is fairly pleasing.